Disease Prediction Based On Symptoms Using Large Language Models

A Final Project End Semester Report

submitted by

SAURABH GUPTA (CS20B1056)

in partial fulfilment of requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

Department of Computer Science and Engineering

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, DESIGN AND MANUFACTURING, KANCHEEPURAM

May 2024

DECLARATION OF ORIGINALITY

I, **SAURABH GUPTA**, with Roll No: **CS20B1056** hereby declare that the material presented in the Project Report titled **Disease Prediction Based On Symptoms Using Large Language Models** represents original work carried out by me in the **Department of Computer Science and Engineering** at the Indian Institute of Information Technology, Design and Manufacturing, Kancheepuram.

With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

SAURABH GUPTA

Place: Chennai

Date: 16.06.2024

CERTIFICATE

This is to certify that the report titled **Disease Prediction Based On Symptoms Using Large Language Models**, submitted by **SAURABH GUPTA (CS20B1056)**, to the Indian Institute of Information Technology, Design and Manufacturing Kancheepuram, in partial fulfilment of requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** is a bonafide record of the work done by him/her under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Sadagopan Narasimhan
Project Internal Guide
Associate Professor
Department of Computer Science and Engineering
IIITDM Kancheepuram, Chennai - 600 127

Place: Chennai

Date:

ACKNOWLEDGEMENTS

I take this opportunity to express my sincere gratitude to my respected Project Guide, **Dr. Sadagopan Narasimhan**, for their constant support, invaluable guidance, and mentorship throughout this journey. Their expertise, motivation, and helpful feedback have been essential in guiding this project and expanding my knowledge in the field. Moreover, I'm thankful for the freedom he gave me to delve deep into the problem, which greatly enhanced my skills in software engineering and artificial intelligence.

I am deeply grateful to my Brother, whose unwavering encouragement, insightful discussions, and technical assistance have been a constant source of motivation and inspiration.

To my Mother, I owe a debt of gratitude for her unconditional love and unwavering support. Her belief in my abilities, encouragement during challenging times, and sacrifices have been the cornerstone of my academic journey. This project is a testament to her enduring support and guidance.

I would also like to extend my heartfelt thanks to my friends and close ones for their encouragement, motivation, and unconditional support. Their support, brainstorming sessions, and moral support have provided me with the strength and resilience to overcome obstacles and achieve my goals.

Thank you all for being part of this journey and for your unwavering support.

Sincerely,

Saurabh Gupta

ABSTRACT

In the healthcare domain, creating a disease prediction model based on symptoms addresses critical needs by improving early diagnosis, enabling timely interventions, and optimizing medical resources. This work involves developing a model to accurately identify diseases from symptom descriptions or user queries. The objectives include exploring solutions for fine-tuning large language models (LLMs) on limited computational resources, implementing post-processing of generated outputs, and analyzing results. Key contributions include fine-tuning six models (LLaMA-3 8B Instruct Model, Qwen-2 7B Instruct Model, Mistral 7B Instruct Version 0.2, Phi-3 Medium 4K Instruct Model, Gemma 7B Instruct Model, and Medical LLaMA-3 8B) on a training dataset, integrating post-processing using semantic search by FAISS with Clinical BERT encodings. Achieved remarkable metrics for the **Fine-Tuned LLaMA-3 8B Instruct model**. This model achieved the highest scores across **accuracy (0.968), precision (0.957), recall (0.968), F1-score (0.961), ROUGE scores (ROUGE-1: 0.969, ROUGE-L: 0.968, ROUGE-LSum: 0.969), and BERT scores (BERT-P: 0.994, BERT-R: 0.994, BERT-F1: 0.994)**, making it the most reliable and effective for disease prediction.

KEYWORDS: Healthcare; Disease Prediction; Diagnosis; Natural Language Processing; Large Language Models; Text Classification; Text Generation; Transformers; LLaMA-3; Mistral; Phi-3; Gemma; Qwen-2; FAISS; ClinicalBERT; Semantic-Search

TABLE OF CONTENTS

ACKNOWLEDGEMENTS **i**

ABSTRACT **ii**

LIST OF TABLES **vi**

LIST OF FIGURES **viii**

ABBREVIATIONS **ix**

1 INTRODUCTION **1**

1.1 MOTIVATION 1

1.2 PROBLEM STATEMENT 1

1.3 OBJECTIVES AND SCOPE 1

1.4 CONTRIBUTIONS 2

2 LITERATURE REVIEW **3**

2.1 CONCEPT PAPERS 3

2.2 TECHNICAL LITERATURE SURVEY 3

2.2.1 USE OF AI IN HEALTHCARE 3

2.2.2 USE OF LLMS IN HEALTHCARE 4

2.2.3 PRECONSULTATION AND DIAGNOSIS USING LLMS . 5

2.2.4 CONCLUSION 6

3 METHODOLOGY AND WORK FLOW **7**

3.1 OVERVIEW 7

3.2 PREPARING DATASET FOR TRAINING & TESTING OF LLM MODELS 9

3.2.1 DATA COLLECTION AND CLEANING 9

3.2.2 DATASET MERGING AND TRAIN-TEST SPLITTING . . 11

3.3 LARGE LANGUAGE MODELS (LLMS) 12
3.3.1 TRANSFORMERS 12
3.3.2 TOKENIZER 14
3.3.3 LARGE LANGUAGE MODELS (LLMS) IN DISEASE PREDICTION 16
3.3.4 INFERENCING USING LLMS 17
3.4 QUANTIZATION 17
3.4.1 TECHNIQUES FOR QUANTIZING LLMS 17
3.4.2 PEFT, LORA, QLORA AND LOFTQ 17
3.4.3 LLMS EVALUATED IN THIS PROJECT 18
3.5 FINE-TUNING OF LLMS 20
3.6 POST-PROCESSING OF GENERATED RESPONSES USING FAISS WITH CLINICAL-BERT'S ENCODINGS 21
3.7 PROPOSED SOLUTION 22
3.8 EVALUATION METRICS 23

4 IMPLEMENTATION AND RESULTS 25
4.1 KAGGLE NOTEBOOKS 25
4.1.1 DATA PREPARATION 25
4.1.2 TEST OF PRE-TRAINED MODELS 25
4.1.3 FINE-TUNING AND TESTING OF FINE-TUNED MODELS 25
4.1.4 ANALYSIS 26
4.2 INFERENCING WITH LLMS 26
4.2.1 MODEL LOADING AND QUANTIZATION USING UNSLOTH 26
4.2.2 PROMPT TEMPLATE (INPUT) 27
4.2.3 EXTRACTING RESPONSE 28
4.2.4 INFERENCING ON THE TEST DATASET 29
4.3 MODEL CONFIGURATIONS 30
4.3.1 MODEL LOADING CONFIGURATIONS 30
4.3.2 PEFT CONFIGURATIONS 31
4.3.3 TRAINER CONFIGURATIONS FOR FINE-TUNING . . . 32
4.4 RESULT : EVALUATION OF THE MODELS 33

4.4.1 EVALUATION OF PRE-TRAINED MODELS 34
4.4.2 EVALUATION OF FINE-TUNED MODELS 36
4.4.3 EVALUATION OF FINE-TUNED MODELS WITH POST-PROCESSING OF OUTPUTS USING FAISS WITH CLINICAL BERT'S ENCODINGS 38
4.4.4 GRAPHICAL COMPARISONS 46
4.5 SUMMARY 53
4.5.1 BEST MODELS 53

5 CONCLUSIONS AND FUTURE SCOPE 55
5.1 CONCLUSIONS 55
5.2 FUTURE SCOPE 55

REFERENCES 60

LIST OF TABLES

2.1 ATP's Triage Protocol for Emergency Classification 3

3.1 Performance metrics of various models on different benchmarks[1, 2, 3, 4, 5] 20

4.1 Model Loading Configurations 31

4.2 PEFT Configurations 31

4.3 Training Configurations and Arguments 32

4.4 Pretrained Models: Valid Instruction Following, Accuracy, Precision, Recall, F1-score 34

4.5 Pretrained Models: Valid Instruction Following, ROUGE, and BERT Scores 35

4.6 Fine-tuned Models: Valid Instruction Following, Accuracy, Precision, Recall, F1-score 36

4.7 Fine-tuned Models: Valid Instruction Following, ROUGE, and BERT Scores 37

4.8 Post-processed Fine-tuned Models: Valid Instruction Following, Accuracy, Precision, Recall, F1-score 38

4.9 Post-processed Fine-tuned Models: Valid Instruction Following, ROUGE, and BERT Scores 39

4.10 Summary of total improvements in score from non post-processed pretrained models to post-processed fine-tuned models (average across all metrics) 52

4.11 Best Models Evaluation Scores : Valid Instruction Following, Accuracy, Precision, Recall, F1-score 53

4.12 Best Models Evaluation Scores: Valid Instruction Following (Valid Inst. Follow.), ROUGE, and BERT Scores 54

LIST OF FIGURES

3.1 Project Workflow 7

3.2 Dataset collection from Kaggle and Hugging Face for Symptom-based Disease Classification 10

3.3 Merge Datasets and Split into Train & Test Datasets 11

3.4 Final Train & Test Dataset 11

3.5 Transformer Model Architecture [6] 12

3.6 Types of Tokenizations [7] 14

3.7 An example of LLaMA-2 7B Tokenizer from Hugging Face's Transformer Library 15

3.8 Post-Processing of generated responses using FAISS with Clinical-BERT's Encodings 21

3.9 Proposed solutions with Fine-Tuned LLMs with post-processing . . 22

4.1 Model Loading and Quantization using Unsloth 26

4.2 Prompt Template (Input) for Inferencing 27

4.3 Inferencing and Response Extraction 28

4.4 Formatting the Test Dataset in proper Prompt Format 29

4.5 Inferencing on the LLMs using Test Dataset for Disease Prediction . 30

4.6 Confusion Matrix for Qwen-2 7B Instruct Outputs 40

4.7 Confusion Matrix for Gemma 7B Instruct Outputs 41

4.8 Confusion Matrix for Mistral 7B Instruct Outputs 42

4.9 Confusion Matrix for Phi-3 Medium 4K Instruct Outputs 43

4.10 Confusion Matrix for Medical LLaMA-3 8B Outputs 44

4.11 Confusion Matrix for LLaMA-3 8B Instruct Outputs 45

4.12 Accuracy Score for each Model 46

4.13 Precision Score for each Model 46

4.14 Recall Score for each Model 47

4.15 F1 Score for each Model 47

4.16 Accuracy Trends across Models 48
4.17 F1 Score Trends across Models 48
4.18 Classification Metrics Radar Chart : Qwen-2 7B Instruct 49
4.19 Text-to-Text Metrics Radar Chart : Qwen-2 7B Instruct 49
4.20 Classification Metrics Radar Chart : Mistral 7B Instruct 50
4.21 Text-to-Text Metrics Radar Chart : Mistral 7B Instruct 50
4.22 Classification Metrics Radar Chart : LLaMA-3 8B Instruct 51
4.23 Text-to-Text Metrics Radar Chart : LLaMA-3 8B Instruct 51

ABBREVIATIONS

LLM	Large Language Model
FAISS	Facebook AI Similarity Search
BERT	Bidirectional Encoder Representations from Transformers
GPT	Generative Pre-trained Transformer
LLaMA	Large Language Model Meta AI,
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
LoRA	Low-Rank Adaptation
PEFT	Parameter-Efficient Fine-Tuning
QLoRA	Quantized Low-Rank Adaptation
LoftQ	LoRA-Fine-Tuning-Aware Quantization
MVP	Minimum Viable Product
SOS	Save Our Souls (Emergency Distress Signal)
API	Application Programming Interface
CSV	Comma-Separated Values
HIPAA	Health Insurance Portability and Accountability Act
FHIR	Fast Healthcare Interoperability Resources

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

In the healthcare domain, creating a disease prediction model based on symptoms addresses several critical needs. This is because it can significantly improve early diagnosis, enabling timely intervention and potentially better patient outcomes. Along with this, it helps optimize the use of medical resources by guiding patients to the appropriate specialties, thereby reducing unnecessary consultations and tests.

1.2 PROBLEM STATEMENT

Developing a disease prediction model which can accurately identify disease based on the detailed description of symptoms or user query. The model should not only just **predict diseases based on symptoms** but should also be able to perform **text summarization** and **multi-turn conversation (question answering)**.

1.3 OBJECTIVES AND SCOPE

- Exploring solutions for fine-Tuning of LLM models especially on limited computer resources and afterwords fine-tuning them on the training dataset.
- Exploring solutions for post-processing of generated outputs.
- Testing fined-tuned models and post-processing the outputs. Analysing the results after post-processing.
- Testing Model for three tasks - **disease prediction**, **text summarization** and **multi-turn conversation (question answering)**.

1.4 CONTRIBUTIONS

- Fine-tuned all six models **LLaMA-3 8B Instruct Model**, **Qwen-2 7B Instruct Model**, **Mistral 7B Instruct Version 0.2**, **Phi-3 Medium 4K Instruct Model**, **Gemma 7B Instruct Model** and **Medical LLaMA-3 8B** on the Training Dataset.
- Integrated post-processing method for improved disease prediction using similarity-search[8] by FAISS[9] with Clinical BERT's encodings [10] and performed analytics on post-processed outputs.
- Achieved incredible metrics scores for the **Fine-Tuned LLaMA-3 8B Instruct** model with post-processed ouputs. It achieves the highest scores across **accuracy (0.968), precision (0.957), recall (0.968), F1-score (0.961), ROUGE scores (ROUGE-1 : 0.969; ROUGE-L : 0.968, ROUGE-LSum : 0.969), and BERT scores (BERT-P: 0.994; BERT-R : 0.994; BERT-F1:0.994)**, making it the most reliable and effective model for the task at hand.
- Successfully tested top LLM - **Fine-Tuned LLaMA-3 8B Instruct** for three tasks:- **disease prediction**, **text summarization** and **multi-turn conversation (question answering)**.

CHAPTER 2

LITERATURE REVIEW

2.1 CONCEPT PAPERS

- **Emergency Medical Service (EMS) in India: A Concept Paper [11]** : The concept paper highlights the differences in Emergency Medical Services (EMS) around the world. It compares the doctor-led EMS models in Europe with the paramedic-led systems in North America, the UK, and Australia. In India, EMS faces problems like poor infrastructure, lack of training, and unclear legal rules. A complete plan is needed to improve ambulance services, call centers, health facilities, training, education, legal guidelines, and management.

- **ATP for managing busy emergency services [12]** : India has a lot of traffic of emergency cases but lacks a proper ED triage system. The All India Institute of Medical Sciences (AIIMS) created the AIIMS Triage Protocol (ATP) to overcome this. As shown in Table 2.1, it uses color codes (Red, Yellow, Green) for immediate, semi-urgent, and minor cases. This system improves care by aligning triage decisions with resources, making patient identification and management easier.

Table 2.1: ATP's Triage Protocol for Emergency Classification

ATP category	Urgency Level	Types of Emergency
Red	Very Urgent	Immediate life-threatening situations to those with potentially life-threatening or situational urgency.
Yellow	Urgent	Requiring multiple resources despite non-dangerous vital signs / Requiring only one resource / Potentially serious
Green	Standard	No resource needed / Normal Problem

-

2.2 TECHNICAL LITERATURE SURVEY

2.2.1 USE OF AI IN HEALTHCARE

- **Review paper on the Role of Machine Learning in Healthcare [13]**
  This review paper provides a summary of the applications of machine learning re-

lated algos and techniques in healthcare, that indicate their potential in disease diagnosis, treatment optimization, and healthcare management. It discusses various machine learning algos and their performance in different healthcare domains.

- **Predicting Hospital Readmissions with Machine Learning Models[14]**
  This paper focused on hospital readmission prediction - probability of a patient to get readmissioned. Also this paper explores the use of machine learning techniques to identify patients at risk of readmission. It discusses the use of patient data, such as demographics, medical history, and clinical variables, to develop predictive models for readmission risk assessment tasks - thus providing a early detection system for patients in need.

- **Analyzing the Distribution of Urgency in Accident and Emergency Departments[15]**
  This study analyzed the distribution of urgency related to an accident and workload faced by the emergency department. It examined the patterns of patient conditions and the distribution of urgency levels. It provides insights into the distribution of patient urgency and its impact on resource allocation and capacity planning.

- **Further Analysis of Data from the 1980 Patient Urgency Study [16]**
  This paper presents further analyses the data from the 1980 Patient Urgency Study. It investigated about different factors influencing patient urgency in emergency department settings.It looked at the links between patients' conditions, their complaints, and urgency levels. This helps us understand how emergency departments work and how patients are sorted.

2.2.2 USE OF LLMS IN HEALTHCARE

The field of natural language processing (NLP) has gone through excellent progress with the development of large language models (LLMs) which are trained using deep learning concets on large amount **textual data**. These models may have parameters in billions [6]. Initially introduced by Google in 2017 with the "Transformer" architecture and its subsequent models like BERT [17], GPT-3 [18, 19], and GPT-4 [20] have achieved excellent metrics across numerous NLP tasks.

These LLMs can be categorized based on their pre-training architecture (encoder-only, encoder-decoder, or decoder-only), the tasks they are trained on, and the datasets used [21]. Advances in computational resources have enabled these models to possess "zero-shot" and "few-shot" learning capabilities. These qualities allow them to perform tasks with very less or no specific training [22].

In healthcare, NLP powered by LLMs has made significant changes, like, support-

ing tasks such as pre-consultation, diagnosis, and patient management [23, 24, 25]. Specialized biomedical LLMs like **SCIBERT**, **BioBERT**, **PubMedBERT**, & **ClinicalBERT** have been developed to solve these domain-specific tasks [26, 27, 28, 29, 30] . These models use the learnings from large corpuses from healthcare literature to enhance their performance in medical tasks.

Conversational LLMs specially made for healthcare, such as **Med-PaLM** and **ChatDoctor**, are improving interaction with patients and also help in medical consultation work [31, 32]. These models are fine-tuned to handle medical queries, provide information to users, and offer support through interactive dialogue mechanism.

Some research work also focused on optimizing LLMs for specialized healthcare tasks through fine-tuning and prompt-tuning methods [33, 34, 35]. LLMs (like ChatGPT) can also be used for analysing urgency [36] in times of heavy load on emergency management system. This approach aims to maximize the utility of LLMs in healthcare domain, thereby enhancing patient care and operational efficiency.

Thus, all these indicate the importance of integrating LLMs in healthcare in enhancing medical decision-making, patient interaction, and operational efficiency.

2.2.3 PRECONSULTATION AND DIAGNOSIS USING LLMS

Preconsultation

Analysing Urgency by Enhanced Patient Triaging LLMs helps in triaging cases of emergency by predicting disease categories based on symptoms. They can also helps in recommending specific medical procedures and specialist for appointments, which in turn can optimize medical resource allocation [37].

Integration with Telehealth Services for Low-End Devices in Remote Locations Integrated with tele-health services. LLMs can help in real-time self-assessment tools via SMS, to assess urgency for remote populations. This can facilitate timely access to healthcare [25].

Improving Diagnostic Predictions LLMs improve diagnostic predictions by incorporating patient-specific data such as existing health problems, risk factors, and history of medication.

Diagnosis

Smooth Operations for Patient Consultations LLMs streamline patient consultations by creating concise summaries of medical histories, including critical health conditions (comorbidities) and past treatments [24, 38, 39].

Support Systems for Decision Making LLMs can also act as decision support systems for identifying appropriate radiological investigations, reducing unnecessary pathology which helps in optimizing resource utilization [38].

Enhancing Understanding of the Diagnosis LLMs integrated into computer-aided diagnosis systems can enhance interpretability by enabling healthcare professionals to query the logic behind diagnostic outputs which can potentially help in discovering new disease insights [38].

2.2.4 CONCLUSION

After doing a detailed literature review it can be concluded that there are many chatbots and llm models to interact with patient for question-answering and consultations. However, there are very less solutions for **symptoms based disease classification/prediction**, which is important in preconsultation of patients, especially during times of heavy traffic of emergency cases. Thus, in this project various models and solutions will be explored for disease classification/prediction based on the symptoms.

CHAPTER 3

METHODOLOGY AND WORK FLOW

3.1 OVERVIEW

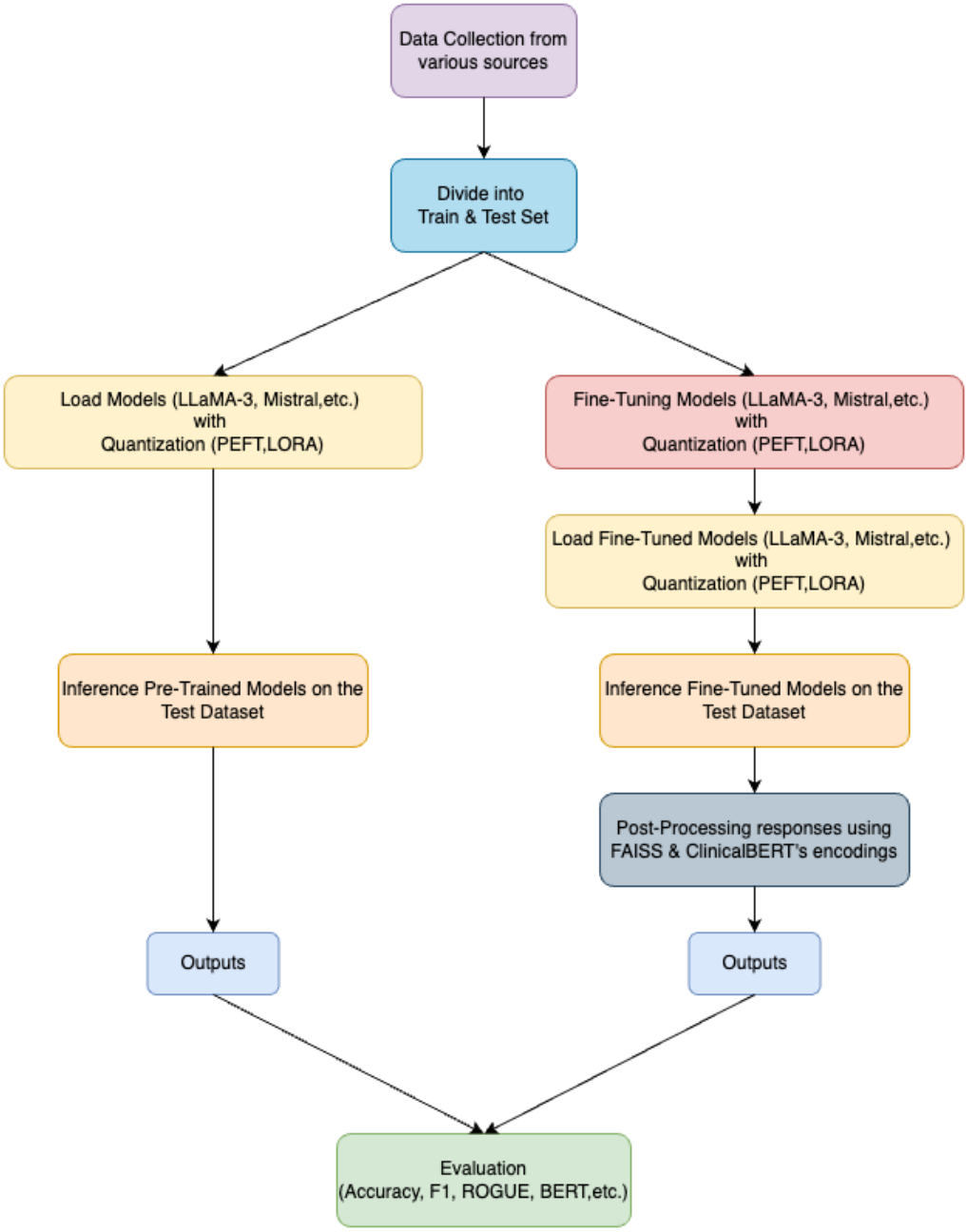


Figure 3.1: Project Workflow

Overview of the workflow:

1. **Data Collection** : Collecting data from sources like Kaggle, Huggingface based on the authenticity and wide user acceptance. These datasets have information about diseases and related symptoms or user query. Thus, a great source for working with symptoms based disease prediction. The final merged dataset with unique rows contains two attributs text (symotoms or user query) which is the input and disease (corresponding disease) which is the label or output.

2. **Splitting Dataset** : Dividing the Merged Dataset into Train and Test Datasets with a ratio of 4:1. The Test Dataset is used for model evaluation while Train Dataset will be used for training. The datasets can be found at this Kaggle Notebook - Data Preparation - Symptom Disease Classification.

3. **Loading Large Language Models (LLMs) with quantization [40, 41, 42, 43]** : Loading the state-of-the art LLMs using unsloth library with PEFT [40] & LoftQ [43] configurations to load the model in a quantized way, such that the size of the models is reduced while maintaining the model's performance with increased speed. LLMs evaluated in this project are **LLaMA-3 8B Instruct Model**, **Qwen-2 7B Instruct Model**, **Mistral 7B Instruct Version 0.2**, **Phi-3 Medium 4K Instruct Model**, **Gemma 7B Instruct Model** and **Medical LLaMA-3 8B** - a Pre-Trained Model on Medical Chatbot Dataset. Details of these models are discussed in later sections.

4. **Inference Pre-Trained Models on the Test Dataset** : The Model are tested on the Test Dataset via Inferencing. These models are loaded through a special library Unsloth. In, inferencing, model takes a prompt and gives a completed prompt called the response. The prompt contains three parts **Instruction** - the instruction given to the model to perform specific task, Input - the actual input, and **Response** - generated response. Use of instruction based prompting is called Prompt Engineering [44] The generated Response is then extracted via string processing and saved as the output.

5. **Fine-Tuning Models** : Fine-tuning the above base models using Unsloth library with the prepared training dataset.

6. **Inference Fine-Tuned Models on the Test Dataset** : Loading the fine-tuned models and inferencing them in the similar way as inferencing the pre-trained models.

7. **Post-Processing of responses using FAISS with ClinicalBERT's Encoder** : The generated response from the fine-tuned models is first processed to get the output within the label domain. After this results are saved for evaluation.

8. **Evaluation** : The evaluation of the generated outputs for both pre-trained and fine-tuned (with post-processing) models are done one the basis of classification metrics like accuracy, precision, recall, f1-score. One more type of evaluation is also done, i.e., text to text evaluation metrics like ROGUE and BERT scores [45, 46].

3.2 PREPARING DATASET FOR TRAINING & TESTING OF LLM MODELS

3.2.1 DATA COLLECTION AND CLEANING

Following Datsets have been gathered from Kaggle and Huggingface for Symptom-based Disease Classification:

- Datset-1 | disease-and-symptoms-dataset | Kaggle
  https://www.kaggle.com/datasets/choongqianzheng/disease-and-symptoms-dataset?select=DiseaseAndSymptoms.csv
- Dataset-2 | duxprajapati/symptom-disease-dataset | Hugging Face
  https://huggingface.co/datasets/duxprajapati/symptom-disease-dataset
- Dataset-3 | disease-prediction-based-on-symptoms | Kaggle
  https://www.kaggle.com/datasets/pasindueranga/disease-prediction-based-on-symptoms
- Dataset-4 | symptom2disease | Kaggle
  https://www.kaggle.com/datasets/niyarrbarman/symptom2disease?select=Symptom2Disease.csv

DiseaseAndSymptoms.csv (632.2 kB)

Detail Compact Column 10 of 18 columns

Disease Disease name	Symptom_1 Symptom of the disease	Symptom_2 Symptom of the disease	Symptom_3 Symptom of the disease	Symptom_4 Symptom of the disease	Sympto Symptom
41 unique values	vomiting 17% fatigue 14% Other (3408) 69%	vomiting 18% fatigue 8% Other (3648) 74%	fatigue 15% high_fever 7% Other (3870) 79%	high_fever 8% [null] 7% Other (4194) 85%	[null] headache Other (33
Fungal infection	itching	skin_rash	nodal_skin_eruptions	dischromic _patches	
Fungal infection	skin_rash	nodal_skin_eruptions	dischromic _patches		
Fungal infection	itching	nodal_skin_eruptions	dischromic _patches		
Fungal infection	itching	skin_rash	dischromic _patches		

Dataset-1 | Kaggle

Dataset Viewer Auto-converted to Parquet API View in Dataset Viewer

Split (2)
train · 5.63k rows

Search this dataset

text string · lengths 28 18.7k	label int64 0 1.08k
I have been having migraines and headaches. I can't sleep. My whole body is shaking and shivering. I feel dizzy sometimes.	308
I have asthma and I get wheezing and breathing problems. I also have fevers, headaches, and I feel tired all the time.	35
Signs and symptoms of primary ovarian insufficiency are similar to those of menopause or estrogen deficiency. They include: Irregular or skipped periods, which might be present for years or…	798

Dataset-2 | Hugging Face

dataset.csv (16 kB)

Detail Compact Column 5 of 5 co

disease	symptoms	cures	doctor	risk level
pneumonia 3% alzheimer's disease 3% Other (93) 94%	memory loss,confu... 4% fever,cough,shortn... 3% Other (92) 93%	antibiotics 9% surgery,chemother... 7% Other (83) 84%	family doctor,urge... 11% family doctor,pulm... 6% Other (82) 83%	varies 28% high (70%) 16% Other (55) 56%
flu	fever,cough,sore throat,runny or stuffy nose,muscle aches,headache,fatigue	over-the-counter medications,rest,fluids	family doctor,urgent care	low (0.1%
bronchitis	cough,mucus production,shortness of breath,chest pain	antibiotics,over-the-counter medications,rest,fluids	family doctor,pulmonologist	low (0.5%

Dataset-3 | Kaggle

Symptom2Disease.csv (229.85 kB)

Detail Compact Column

# index	label label	text text
0 299	24 unique values	1153 unique values
0	Psoriasis	I have been experiencing a skin rash on my arms, legs, and torso for the past few weeks. It is red, ...
1	Psoriasis	My skin has been peeling, especially on my knees, elbows,

Dataset-4 | Kaggle

Figure 3.2: Dataset collection from Kaggle and Hugging Face for Symptom-based Disease Classification

3.2.2 DATASET MERGING AND TRAIN-TEST SPLITTING

Merging all the datasets (Figure : 3.3) to get a final dataset with symptoms and corresponding disease. After, this the merged data is further divided judiciously into Train and Test Datasets Figure : 3.4 in the proper ratio of 4:1.

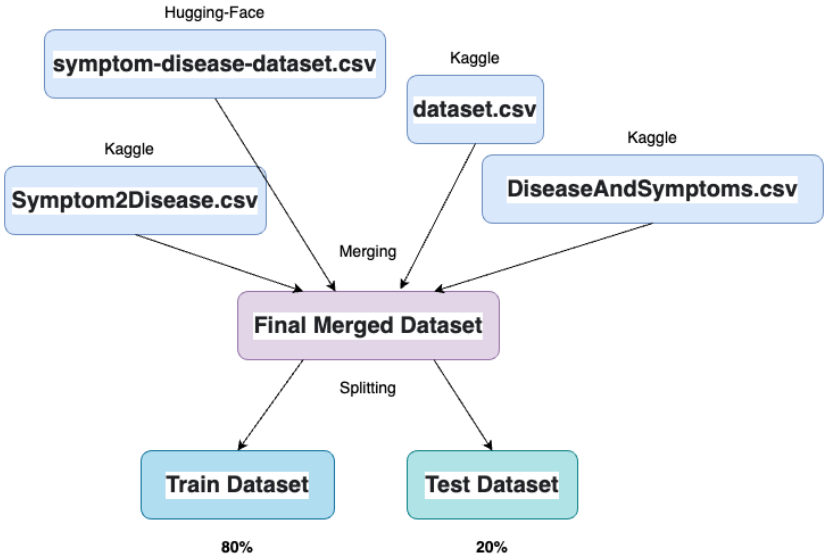


Figure 3.3: Merge Datasets and Split into Train & Test Datasets

train.csv (1.28 MB)

text	disease
vomiting,headache,nausea,loss_of_balance,unsteadiness	(vertigo) Paroymsal Positional Vertigo
headache,nausea,spinning_movements,loss_of_balance,unsteadiness	(vertigo) Paroymsal Positional Vertigo
vomiting,nausea,spinning_movements,loss_of_balance,unsteadiness	(vertigo) Paroymsal Positional Vertigo
vomiting,headache,nausea,spinning_movements,loss_of_balance,	(vertigo) Paroymsal Positional Vertigo

Final Train Dataset

test_primary.csv (124.61 kB)

text	disease
vomiting,headache,nausea,spinning_movements,loss_of_balance	(vertigo) Paroymsal Positional Vertigo
vomiting,headache,nausea,spinning_movements,unsteadiness	(vertigo) Paroymsal Positional Vertigo
vomiting, nausea, spinning_movements, loss_of_balance, unsteadiness	(vertigo) Paroymsal Positional Vertigo
When I awoke this morning, I saw a severe rash across	Acne

Final Test Dataset

Figure 3.4: Final Train & Test Dataset

3.3 LARGE LANGUAGE MODELS (LLMS)

3.3.1 TRANSFORMERS

Transformers [6] are fundamental basic architecture in large language models (LLMs) that have completely changed the level of natural language processing (NLP) by enabling more effective and efficient understanding of context within text data. The architecture is designed to handle data in sequential manner, which make it perfectly suitable for tasks involving language and time series analysis.

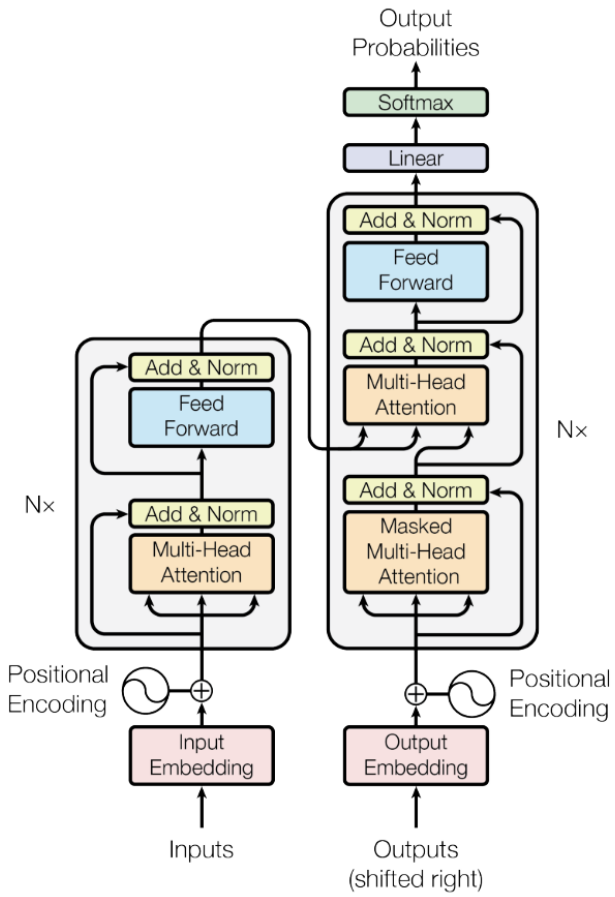


Figure 3.5: Transformer Model Architecture [6]

Architecture of Transformer

- **Attention Mechanism**: Transformers use an attention mechanism enabling the model to evaluate the significance of individual words in a sentence, irrespective of their position. Thus, enhancing contextual comprehension beyond the level of conventional models.

- **Encoder-Decoder Structure**: Initially, transformers were designed with an encoder for processing input sequences and a decoder for generating output sequences. However, several LLMs such as BERT and GPT utilize either the encoder or decoder component exclusively. Encoder is the part which takes the initial sequence the pass on the encoded information. Then decoder, generates the output using this encoded information and completes use this new generated information as the next input. Thus, the cycle continues and a desired output is generated.

Important Components

- **Self-Attention**: This mechanism allows the model to evaluate all positions within the input sequence concurrently, helping in identifying the correlations (context) between different parts effectively.
- **Feed-forward Neural Networks**: These add non-linearity to the model. It is applied to each part separately.
- **Positional Encoding**: Transformers lack inherent positional understanding, so, positional encodings are incorporated into input embeddings. This provides the model with crucial information regarding the sequential placement of words or tokens within a sentence.

What benefits they provide to Large Language Models (LLMs)

Pretraining of LLMs

- LLMs are typically pretrained on large corpora of text using transformer architectures. This involves training the model to predict missing words in sentences, called, **masked language modeling** or the next word in a sentence called, **causal language modeling**.
- **Examples**: BERT (Bidirectional Encoder Representations from Transformers) [17] uses a masked language modeling technique, whereas GPT Models (Generative Pretrained Transformer) [19] use causal language modeling for text generation.

Fine-Tuning of LLMs

- LLMs can be fine-tuned [47] after pretraining, on specific tasks or domains. It allows them to adapt to particular applications, like for example, text classification, sentiment analysis, question answering, machine translation, and so on.

3.3.2 TOKENIZER

A tokenizer [7] serves as a preprocessing tool designed to transform text into tokens. Tokens are fundamental units of text that models can interpret and handle very easily. Depending on the chosen tokenization method (Figure 3.6), these tokens may represent words, subwords, characters, or other. Here, are the types of tokenization methods used in NLP.

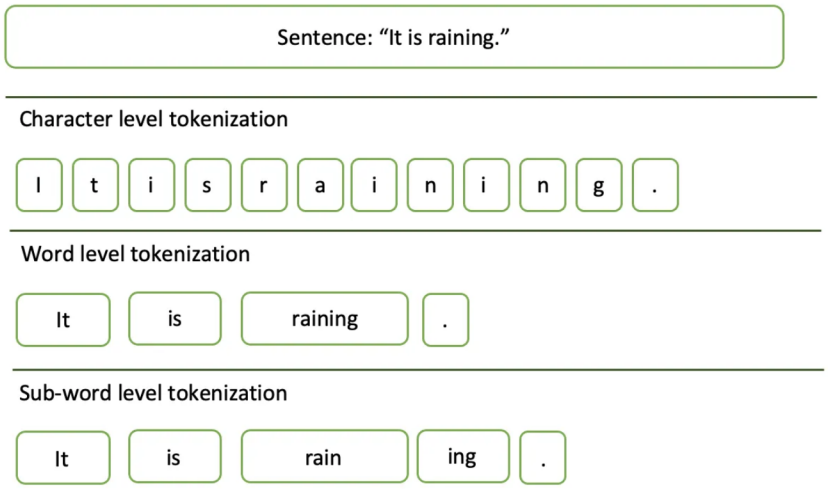


Figure 3.6: Types of Tokenizations [7]

Types of Tokenizers

Word-Level Tokenizer:

- **Description**: Splits text into individual words.
- **Example**: "Tokenizers are important." → ["Tokenizers", "are", "important", "."]

Character-Level Tokenizer:

- **Description**: Splits text into individual characters.
- **Example**: "Tokenizers" → ["T", "o", "k", "e", "n", "i", "z", "e", "r", "s"]

Subword Tokenizer (Byte Pair Encoding (BPE), WordPiece, SentencePiece):

- **Description**: Splits text into subword units, balancing between words and characters.

- **Example**: "Tokenizers are important." → ["Token", "izers", "are", "im", "port", "ant", "."]

Sentence Tokenizer:

- **Description**: Splits text into sentences.
- **Example**: "Tokenizers are important. They help preprocess text." → ["Tokenizers are important.", "They help preprocess text."]

Use of Tokenization in LLMs

Text Preprocessing

- Tokenizers prepare raw text data as input for LLM models. They convert text into numerical representations (token IDs) that the model can process easily.

Code Example

main.py

```python
from transformers import AutoTokenizer

# Load the LLaMA-2 tokenizer
tokenizer = AutoTokenizer.from_pretrained("meta-llama/Llama-2-7b")

# Example input text
text = "Explain Hypertension?"

# Tokenize the input text
tokens = tokenizer(text)

# Print the tokenized output
print(tokens)
```

Figure 3.7: An example of LLaMA-2 7B Tokenizer from Hugging Face's Transformer Library

Management of Vocabulary

- Tokenizers define a **fixed vocabulary** of tokens that the model can understand and produce outputs in the same format. Each token is mapped to a unique identifier (token IDs).

3.3.3 LARGE LANGUAGE MODELS (LLMS) IN DISEASE PREDICTION

Large Language Models (LLMs) [48] are advanced artificial intelligence systems developed for comprehending and generating textual data in human language. These are built on transformer architectures (discussed earlier). LLMs like GPT [19] and BERT [17, 23] have completely changed natural language processing by leveraging vast amounts of text data for training. These models excel at tasks ranging from language translation & sentiment analysis to summarization & question-answering. They have shown their excellent ability to comprehend context and generate related text. LLMs operate by breaking down text into tokens, and then they process them through layers of self-attention in addition with feedforward neural networks. After this they generate responses based on learned patterns and structures in language data. Their versatility makes them so important across various applications. From enhancing search engines and automating content creation to powering chatbots - LLMs are now used everywhere.

The workflow of LLMs involves initial **training on extensive datasets** (also known as pre-training) followed by **fine-tuning for specific tasks** - enabling them to adapt to different domains and refine their performance. Through **tokenization** and **encoding** processes, LLMs convert raw text inputs into numerical representations that capture semantic meaning and also the syntactic structure. This pre-processing step ensures that the models can handle a highly diverse linguistic nuances and a lot of complexities. Thus, allowing them to comprehend and generate text with a level of proficiency same as to that of humans. As research and development in LLMs continue to progress, these models are expected to play an increasingly important role in advancing natural language understanding and applications across industries, promising further breakthroughs in AI-driven - language processing technologies.

3.3.4 INFERENCING USING LLMS

Text-Classification Tasks through LLMs

Large Language Models (LLMs) are extensively used in text classification and **prediction tasks** like **sentiment analysis**, **fake news detection**, etc. These can also be for disease detection based on symptoms. This can be done by analyzing textual descriptions of symptoms, LLMs can classify and predict potential diseases with high accuracy. These models use their understanding of context and semantic relationships to interpret symptoms effectively. Thus, aiding in early diagnosis and treatment planning.

3.4 QUANTIZATION

Quantization in LLMs (Large Language Models) involves reducing the precision of the model's weights and activations. This technique is crucial for minimizing the memory usage and computational needs of the model. Thus, making it feasible for deployment on resource-constrained devices or for optimizing the computational efficiency during inference.

3.4.1 TECHNIQUES FOR QUANTIZING LLMS

- **Integer Quantization (IQ):** Represents weights and activations as integers.
- **Ternary Quantization (TQ):** Represents weights and activations as ternary values (-1, 0, or 1).
- **Binary Quantization (BQ):** Represents weights and activations as binary values (0 or 1).
- **Post-Training Quantization (PTQ):** Quantisation after training.
- **Quantization-Aware Training (QAT):** Moodel training with quantized weights and activations.

3.4.2 PEFT, LORA, QLORA AND LOFTQ

- **PEFT (Parameter-Efficient Fine-Tuning) [40]:** It optimizes large pretrained language models (PLMs) by reducing parameters and memory usage during adap-

tation. It maintains task performance similar to full fine-tuning, making PLMs feasible for resource-limited deployments.

- **LoRA (Low-Rank Adaptation) [41]:** LoRA involves the freezing of pre-trained model weights and introduces trainable rank decomposition matrices, drastically reducing trainable parameters and GPU memory usage while outperforming traditional fine-tuning on various large language models.
- **QLoRA (Quantized LORA)[42]:** QLoRA represents an efficient fine-tuning approach for large models, achieving memory efficiency that allows fitting a 65-billion-parameter model on a single 48GB GPU. It uses 4-bit quantization and LoRA, maintaining high performance and achieving 99.3% of ChatGPT's performance. It uses innovations like 4-bit NormalFloat, double quantization, and paged optimizers.
- **LoftQ (LoRA-Fine-Tuning-aware Quantization) [43]** : LoftQ is a framework for quantizing large language models (LLMs). It improves LoRA fine-tuning by finding optimal low-rank initialization, reducing performance gaps between quantized and full-precision models. LoftQ outperforms existing methods, especially in 2-bit and mixed precision regimes

LORA ADAPTORS [49]

LORA adaptors are used to enhance the accuracy of quantized Large Language Models (LLMs) by linearizing **attention weights**. This handles the impact of quantization errors, thereby enhancing the model's capacity to generalize to new, unseen data. These compact, trainable modules are integrated into the attention mechanism of a transformer-based LLM using merging. During the quantization process, they are trained alongside the rest of the model to dynamically adjust attention weights, effectively compensating for quantization-induced errors.

3.4.3 LLMS EVALUATED IN THIS PROJECT

LLAMA-3 8B INSTRUCT [1]

LLama-3 8B Instruct is an 8 billion parameter model from the LLama family, designed for instruction-based learning. It excels at generating detailed and structured responses by following specific user instructions, that's why known as Instruct model. Utilizing the transformer architecture, LLama-3 8B Instruct is well-suited for applications

requiring complex text understanding and generation, such as detailed report generation, educational tools, etc. These models undergo pre-training on a variety of datasets, including C4 (15%), GitHub (4.5%), Stack Exchange (2%), Gutenberg and Books3 (4.5%), English CommonCrawl (67%), ArXiv (2.5%), and Wikipedia (4.5%).

QWEN-2 7B INSTRUCT [5]

Qwen-2 7B Instruct, developed by Alibaba Cloud, is a 7 billion parameter model fine-tuned for instruction-following tasks, again to generate detailed and structured responses by following specific user instructions. It leverages advanced NLP techniques within the transformer framework to accurately interpret and generate text based on user instructions. This model is ideal for task automation, instruction-based data entry, and workflow management systems, balancing computational efficiency with robust performance.

MISTRAL 7B INSTRUCT V0.2 [2]

Mistral 7B Instruct V0.2 is a 7 billion parameter model from the Mistral series, focusing on enhanced instruction-based tasks. This version includes improvements for generating relatable and structured responses, making it effective for interactive virtual assistants, content generation following specific guidelines, and automated documentation. It uses state-of-the-art pretraining and fine-tuning techniques within the transformer architecture.

PHI-3 MEDIUM 4K INSTRUCT [3]

Phi-3 Medium 4K Instruct is a medium-sized model in the Phi series with 4,000 parameters, optimized for instruction-based tasks. It is designed to deliver precise and accurate outputs based on given instructions, leveraging the transformer architecture for effective NLP handling. Suitable applications include educational content creation, step-by-step guides, and structured data generation, offering a balance between efficiency and performance.

GEMMA 7B INSTRUCT [4]

Gemma 7B Instruct is a 7 billion parameter model from the GEMMA family, tailored for instruction-based tasks with an emphasis on multimodal capabilities. This model excels in generating precise and coherent text by following specific instructions, integrating various data types such as text, images, and audio. It is ideal for multimodal content creation, detailed instruction-following applications, and cross-modal data integration, utilizing the robust transformer architecture for high performance.

MEDICAL LLAMA 3 8B

A fine-tuned variant of LLaMA-3 8B by ruslanmv at Hugging Face [ruslanmv/Medical-Llama3-8B], fine-tuned on [Dataset by ruslanmv at Hugging Face]. Thus, will serve as a standard for model evaluation in this project.

Table 3.1: Performance metrics of various models on different benchmarks[1, 2, 3, 4, 5]

Model	**MMLU**	**ARC**	**TruthfulQA**	**HellaSwag**	**GSM8K**
LLaMA-3 8B Instruct	77.5	45.2	59.3	79.0	55.1
Qwen-2 7B Instruct	78.4	44.7	60.1	78.9	54.3
Mistral 7B Instruct v0.2	77.0	45.0	58.9	78.5	54.0
Phi-3 Medium 4K Instruct	76.8	44.5	59.0	78.8	54.1
Gemma 7B Instruct	79.0	45.5	60.2	79.3	54.8

3.5 FINE-TUNING OF LLMS

Fine-Tunings is based on the concept of Transfer learning [50] which involves taking a model trained on one task (usually with a large dataset) and applying it to a different but related task. This approach uses the pre-trained model as a starting point, leveraging the general knowledge it has gained to perform well on new tasks with limited data. In Fine-tuning a pre-trained model is taken and slight adjustments are made to it using a smaller, task-specific dataset. This helps in customising the functioning of model for specific tasks and also helps in increasing the performance of the model.

In this project, the models **LLaMA-3 8B Instruct Model**, **Qwen-2 7B Instruct**

Model, **Mistral 7B Instruct Version 0.2**, **Phi-3 Medium 4K Instruct Model**, **Gemma 7B Instruct Model** and **Medical LLaMA-3 8B** are fine-tuned on the training dataset and then tested on Test Dataset but before that the generated response are first post-processed using FAISS with ClinicalBERT's encodings.

3.6 POST-PROCESSING OF GENERATED RESPONSES USING FAISS WITH CLINICAL-BERT'S ENCODINGS

FAISS (Facebook AI Similarity Search) [9] is a library developed by Facebook AI Research to efficiently search for similar vectors. It is highly optimized for performance, enabling fast and scalable similarity searches in large datasets. FAISS.

ClinicalBERT [10] is a variant of BERT [17] (Bidirectional Encoder Representations from Transformers) specifically fine-tuned on clinical and biomedical text data. It generates dense vector embeddings that capture the semantic nuances of medical language.

In this post-processing solution (Figure : 3.8), FAISS and ClinicalBERT are used together to enhance the disease prediction model. The process involves several steps:

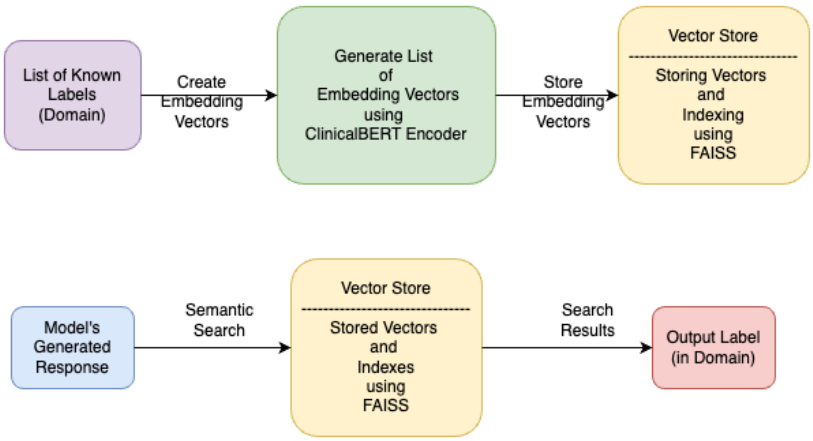


Figure 3.8: Post-Processing of generated responses using FAISS with Clinical-BERT's Encodings

1. **Encoding Disease Labels**: All known disease labels are first encoded into dense vectors using ClinicalBERT. These vectors are then indexed and stored in FAISS for efficient similarity search.
2. **Symptom Input and Initial Prediction**: Symptoms are provided to the respective fine-tuned models, which generate a predicted disease response.
3. **Semantic Search for Post-Processing**: The generated disease response is encoded using ClinicalBERT and searched against the pre-encoded disease vectors stored in FAISS. FAISS retrieves the most semantically similar or equal disease label.
4. **Output**: The retrieved disease label from FAISS is output as the final predicted disease, ensuring that the model's response is semantically accurate and aligned with known disease labels.

Hence, in this way the most semantic similar disease term is produced as output, which is then evaluated for results.

3.7 PROPOSED SOLUTION

Based on the above discussions following solution is proposed for disease prediction task.

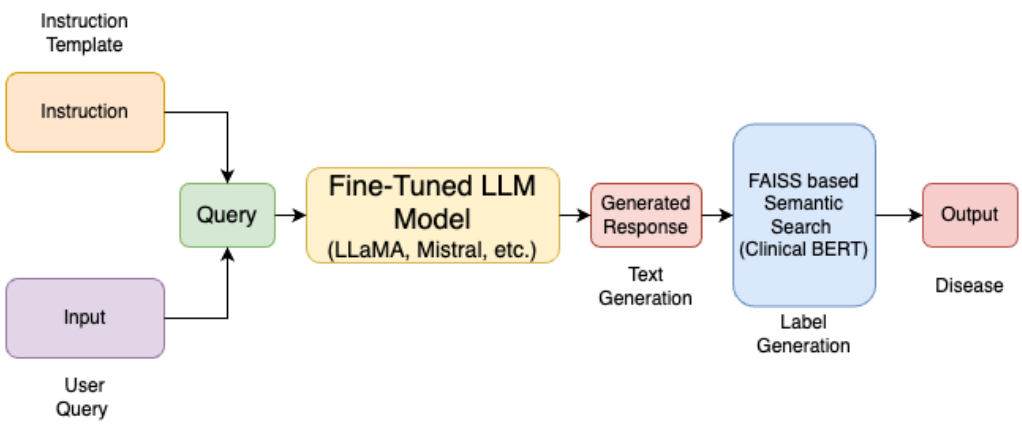


Figure 3.9: Proposed solutions with Fine-Tuned LLMs with post-processing

3.8 EVALUATION METRICS

Evaluation metrics used to assess model prediction and classification performance fall into two categories: classification metrics and text-to-text evaluation metrics. Each metric is briefly defined, accompanied by its mathematical formulation.

A. CLASSIFICATION METRICS

- **Precision**: It is defined as the ratio of correctly predicted positive observations to all predicted positive labels:

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

  where TP denotes True Positives and FP denotes False Positives.

- **Recall**: It is defined as the ratio of correctly predicted positive observations to all actual positive labels:

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

  where FN represents False Negatives.

- **F1-Score**: It is defined as the harmonic mean of Precision and Recall, which balances false positives and false negatives:

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

- **Accuracy**: It is defined as the ratio of correctly predicted observations to all observations:

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

  where TN denotes True Negatives.

B. TEXT-TO-TEXT EVALUATION METRICS

- **ROUGE** (Recall-Oriented Understudy for Gisting Evaluation) has 4 types of measures:
  1. **ROUGE-1**: Measures the overlap of unigrams (single words):

$$\text{ROUGE-1} = \frac{\text{Number of overlapping unigrams}}{\text{Total number of unigrams in the reference}}$$

  2. **ROUGE-2**: Measures the overlap of bigrams (consecutive word pairs):

$$\text{ROUGE-2} = \frac{\text{Number of overlapping bigrams}}{\text{Total number of bigrams in the reference}}$$

3. **ROUGE-L**: Measures the longest common subsequence (LCS) between generated and reference text:

$$\text{ROUGE-L} = \frac{\text{Length of LCS}}{\text{Total number of words in the reference}}$$

4. **ROUGE-LSum**: Measures the LCS overlap in the summary:

$$\text{ROUGE-LSum} = \frac{\text{LCS length in summary}}{\text{Total number of words in the reference summary}}$$

- **BERT-based Scores**:
  1. **BERT Precision**: Evaluates the precision of embeddings between generated and reference texts. Let X be the set of tokens in the generated text and Y be the set of tokens in the reference text. Then,

$$P = \frac{1}{|X|} \sum_{x \in X} \max_{y \in Y} \text{cosine_sim}(x, y)$$

  2. **BERT Recall**: Assesses the recall of embeddings between generated and reference texts.

$$R = \frac{1}{|Y|} \sum_{y \in Y} \max_{x \in X} \text{cosine_sim}(y, x)$$

  3. **BERT F1-Score**: The harmonic mean of BERT Precision and BERT Recall:

$$F1 = \frac{2 \cdot P \cdot R}{P + R}$$

CHAPTER 4

IMPLEMENTATION AND RESULTS

4.1 KAGGLE NOTEBOOKS

The Kaggle Notebooks links for all the Project Codes are given below (click to view) :

4.1.1 DATA PREPARATION

1. Data Preparation - Symptom Disease Classification

4.1.2 TEST OF PRE-TRAINED MODELS

1. Inferencing on Pre-Trained Medical LLaMA-3 8B Model
2. Inferencing on Pre-Trained Mistral 7B Instruct Version 0.2 Model
3. Inferencing on Pre-Trained Gemma 7B Instruct Model
4. Inferencing on Pre-Trained Phi-3 Medium 4K Instruct Model
5. Inferencing on Pre-Trained Qwen-2 7B Instruct Model
6. Inferencing on Pre-Trained LLaMA-3 8B Instruct Model

4.1.3 FINE-TUNING AND TESTING OF FINE-TUNED MODELS

1. Fine-Tuning & Inferencing on Medical LLaMA-3 8B Model
2. Fine-Tuning & Inferencing on Mistral 7B Instruct Version 0.2 Model
3. Fine-Tuning & Inferencing on Gemma 7B Instruct Model
4. Fine-Tuning & Inferencing on Phi-3 Medium 4K Instruct Model
5. Fine-Tuning & Inferencing on Qwen-2 7B Instruct Model
6. Fine-Tuning & Inferencing on LLaMA-3 8B Instruct Model

4.1.4 ANALYSIS

1. Result Analysis - Raw Outputs
2. Result Analysis - Post-Processing using FAISS [9] and ClinicalBERT's Encodings [10]

4.2 INFERENCING WITH LLMS

4.2.1 MODEL LOADING AND QUANTIZATION USING UNSLOTH

main.py Save Run

```python
#Inferencing on Pre-trained Models on Kaggle
from kaggle_secrets import UserSecretsClient
user_secrets = UserSecretsClient()
hf_token = user_secrets.get_secret("HF-TOKEN")

model_id = 'gemma-7b-instruct'
model_type = 'pre-trained'
model_name = "unsloth/gemma-7b-it-bnb-4bit"

from unsloth import FastLanguageModel
import torch

max_seq_length = 2048
dtype = None # None for auto detection. Float16 for Tesla T4, V100, Bfloat16 for Ampere+
load_in_4bit = True

model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = model_name, # model path or repo
    max_seq_length = max_seq_length,
    dtype = dtype,
    load_in_4bit = load_in_4bit,
    token = hf_token,
)

model = FastLanguageModel.get_peft_model(
    model,
    r = 16,
    target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
                      "gate_proj", "up_proj", "down_proj",],
    lora_alpha = 16,
    lora_dropout = 0, # Supports any, but = 0 is optimized
    bias = "none",    # Supports any, but = "none" is optimized
    use_gradient_checkpointing = "unsloth", # 4x longer contexts auto supported!
    random_state = 3407,
    use_rslora = False,  # rank stabilized LoRA
    loftq_config = None, # And LoftQ
)
# Setting the model for inference
FastLanguageModel.for_inference(model) # Enable native 2x faster inference
EOS_TOKEN = tokenizer.eos_token # Must add EOS_TOKEN

```

Figure 4.1: Model Loading and Quantization using Unsloth

4.2.2 PROMPT TEMPLATE (INPUT)

main.py Save Run

```python

# Inference Template
alpaca_prompt = """Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

### Instruction:
{}

### Input:
{}

### Response:
{}"""

def formatting_prompts_func(examples):
    instructions = examples["instruction"]
    inputs       = examples["input"]
    outputs      = examples["output"]
    texts = []
    for instruction, input, output in zip(instructions, inputs, outputs):
        # Must add EOS_TOKEN, otherwise generation will go on forever!
        text = alpaca_prompt.format(instruction, input, output) + EOS_TOKEN
        texts.append(text)
    return { "text" : texts, }
pass

# generating response and cleaning it
def get_raw_response(instruction, input_text):
    inputs = tokenizer(
        [
            alpaca_prompt.format(
                f"{instruction}", # instruction
                f"{input_text}", # input
                "", # output - blank for generation!
            )
        ], return_tensors="pt").to("cuda")

    outputs = model.generate(**inputs, max_new_tokens=16, use_cache=True)
    raw_response = tokenizer.batch_decode(outputs, skip_special_tokens=True)

    return raw_response[0]
```

Figure 4.2: Prompt Template (Input) for Inferencing

4.2.3 EXTRACTING RESPONSE

main.py

```python
# generating response and cleaning it
def get_raw_response(instruction, input_text):
    inputs = tokenizer(
        [
            alpaca_prompt.format(
                f"{instruction}", # instruction
                f"{input_text}", # input
                "", # output - blank for generation!
            )
        ], return_tensors="pt").to("cuda")

    outputs = model.generate(**inputs, max_new_tokens=16, use_cache=True)
    raw_response = tokenizer.batch_decode(outputs, skip_special_tokens=True)

    return raw_response[0]

def get_response(instruction, input_text):
    raw_response = get_raw_response(instruction=instruction, input_text=input_text)
    # Extract Response from raw_response and return it as a string
    response = raw_response.split("### Response:\n")[1].strip()
    return response

# demo
instruction = ("You're an Advanced Medical Chatbot - Chatbot Dr. MCX, so assume yourself as a Healthcare Professional "
                "whose job is to listen to the Patient's query or symptoms and predict the underlying Health Problem the patient is suffering. "
                "Carefully listen to the Patient's query or symptoms and output only the Identified Health Problem in the following format: "
                "Identified Health Problem : [Health Problem Identified by you].")
input_text = "Patient's query or symptoms : skin rash, blackheads, scarring"

print(get_response(instruction=instruction, input_text=input_text))
#Output :: Identified Health Problem : Skin Infection

```

Figure 4.3: Inferencing and Response Extraction

4.2.4 INFERENCING ON THE TEST DATASET

Formatting the Test Dataset in proper Prompt Format

main.py Save Run

```python
# Transforming the Test Dataset into proper Inference Format

import pandas as pd
import os

p_test_df = pd.read_csv('/kaggle/input/data-preparation-symptom-disease-classification/test_primary.c
    )
p_test_df = p_test_df[['text','disease']]
# Perform transformations on p_test_df to convert it into Instruction-Input-Output Format
p_test_df['text'] = "Patient's query or symptoms : " + p_test_df['text']
p_test_df['disease'] = "Identified Health Problem : " + p_test_df['disease']

# Rename columns
p_test_df = p_test_df.rename(columns={'text': 'input', 'disease': 'output'})

# Add new column 'instruction'
instruction = ("You're an Advanced Medical Chatbot - Chatbot Dr. MCX, so assume yourself as a Healthc
    Professional "
               "whose job is to listen to the Patient's query or symptoms and predict the underlying
                   Health Problem the patient is suffering. "
               "Carefully listen to the Patient's query or symptoms and output only the Identified
                   Health Problem in the following format: "
               "Identified Health Problem : [Health Problem Identified by you].")

p_test_df['instruction'] = instruction
p_test_df = p_test_df[['output','input','instruction']]


# Create the directory if it doesn't exist
output_dir = '/kaggle/working/test_dataset'
os.makedirs(output_dir, exist_ok=True)

# Define the file path
output_file = os.path.join(output_dir, 'transformed_test_primary.csv')

# Save the DataFrame to CSV
p_test_df.to_csv(output_file, index=False)

# Display the file path
print("Saved transformed test dataset to:", output_file)
```

Figure 4.4: Formatting the Test Dataset in proper Prompt Format

Inferencing on the LLMs using Test Dataset for Disease Prediction

main.py

```python
# Transforming the Test Dataset into proper Inference Format

import pandas as pd
import os

p_test_df = pd.read_csv('/kaggle/input/data-preparation-symptom-disease-classification/test_primary.c
    )
p_test_df = p_test_df[['text','disease']]
# Perform transformations on p_test_df to convert it into Instruction-Input-Output Format
p_test_df['text'] = "Patient's query or symptoms : " + p_test_df['text']
p_test_df['disease'] = "Identified Health Problem : " + p_test_df['disease']

# Rename columns
p_test_df = p_test_df.rename(columns={'text': 'input', 'disease': 'output'})

# Add new column 'instruction'
instruction = ("You're an Advanced Medical Chatbot - Chatbot Dr. MCX, so assume yourself as a Healthc
    Professional "
               "whose job is to listen to the Patient's query or symptoms and predict the underlying
                   Health Problem the patient is suffering. "
               "Carefully listen to the Patient's query or symptoms and output only the Identified
                   Health Problem in the following format: "
               "Identified Health Problem : [Health Problem Identified by you].")

p_test_df['instruction'] = instruction
p_test_df = p_test_df[['output','input','instruction']]


# Create the directory if it doesn't exist
output_dir = '/kaggle/working/test_dataset'
os.makedirs(output_dir, exist_ok=True)

# Define the file path
output_file = os.path.join(output_dir, 'transformed_test_primary.csv')

# Save the DataFrame to CSV
p_test_df.to_csv(output_file, index=False)

# Display the file path
print("Saved transformed test dataset to:", output_file)
```

Figure 4.5: Inferencing on the LLMs using Test Dataset for Disease Prediction

4.3 MODEL CONFIGURATIONS

4.3.1 MODEL LOADING CONFIGURATIONS

The following are the configurations used to load base models:

Table 4.1: Model Loading Configurations

Parameter	**Value**
max_seq_length	2048
dtype	None
load_in_4bit	True

The key steps and parameters involved in loading the model are as follows:

- `max_seq_length`: Defines the maximum sequence length the model will handle, set to 2048 tokens.
- `dtype`: Specifies the data type for model parameters. If set to `None`, the data type is automatically detected. Specific values like `Float16` or `Bfloat16` can be used for specific hardware optimizations.
- `load_in_4bit`: Boolean value indicating whether to load the model weights in 4-bit precision to save memory.

4.3.2 PEFT CONFIGURATIONS

The following configurations were used to transform the loaded model with Parameter-Efficient Fine-Tuning (PEFT) techniques:

Table 4.2: PEFT Configurations

Parameter	**Value**
r	16
target_modules	{"q_proj", "k_proj", "v_proj", "o_proj", "gate_proj", "up_proj", "down_proj"}
lora_alpha	16
lora_dropout	0
bias	none
use_gradient_checkpointing	unsloth
random_state	3407
use_rslora	False
loftq_config	None

The key parameters and their roles in getting the PEFT [40] model are as follows:

- `model`: The base model loaded previously.
- `r`: Specifies the rank of the LoRA (Low-Rank Adaptation) layers, set to 16.
- `target_modules`: List of specific model layers to apply LoRA, such as projections (`q_proj`, `k_proj`, `v_proj`, etc.).
- `lora_alpha`: Scaling factor for LoRA layers, set to 16.
- `lora_dropout`: Dropout rate for LoRA layers; 0 indicates no dropout.
- `bias`: Configures how biases are handled in the model; `"none"` means no biases are used.
- `use_gradient_checkpointing`: Optimization technique to reduce memory usage during training. `"unsloth"` indicates a specific implementation supporting longer contexts.
- `random_state`: Sets the random seed for reproducibility, set to 3407.
- `use_rslora`: Option to use Rank Stabilized LoRA, set to `False`.
- `loftq_config`: Configuration for LoftQ if used (Low-rank factorized quantization), set to `None`.

4.3.3 TRAINER CONFIGURATIONS FOR FINE-TUNING

The following configurations were used to set up the Trainer for fine tuning [47] for 2 epochs:

Table 4.3: Training Configurations and Arguments

Parameter	**Value**
num_epochs	2
total_training_examples	{len(train_dataset)}
batch_size	2
gradient_accumulation_steps	4
max_steps	3
per_device_train_batch_size	2
warmup_steps	5
learning_rate	2e-4
logging_steps	1
optim	adamw_8bit
weight_decay	0.01
lr_scheduler_type	linear
seed	3407

- **model**: The language model to be fine-tuned. **Value**: `model`
- **tokenizer**: Tokenizer for preprocessing text data.
- **train_dataset**: Dataset used for training.
- **eval_dataset**: Dataset used for evaluation during training.
- **dataset_text_field**: Field in the dataset containing text data. **Value**: `"text"`
- **max_seq_length**: Maximum sequence length for inputs. **Value**: `max_seq_length = 2048`
- **dataset_num_proc**: Number of processes for dataset preprocessing. **Value**: `2`
- **packing**: Whether to pack multiple sequences into one batch. **Value**: `False`
- **per_device_train_batch_size**: Batch size per device (e.g., per GPU). **Value**: `2`
- **gradient_accumulation_steps**: Steps to accumulate gradients before a backward pass. **Value**: `4`
- **warmup_steps**: Steps for learning rate warmup. **Value**: `5`
- **max_steps**: Total number of training steps. **Value**: `max_steps = epochs + 1 = 3`
- **learning_rate**: Learning rate during training. **Value**: `2e-4`
- **fp16**: Use 16-bit (mixed) precision training. **Value**: `False if bf16 is available`
- **bf16**: Use bfloat16 precision training.
- **logging_steps**: Frequency of logging during training. **Value**: `1`
- **optim**: Optimizer type. **Value**: `"adamw_8bit"`
- **weight_decay**: Weight decay for L2 regularization. **Value**: `0.01`
- **lr_scheduler_type**: Learning rate scheduler type. **Value**: `"linear"`
- **seed**: Random seed for initialization. **Value**: `3407`

4.4 RESULT : EVALUATION OF THE MODELS

NOTE :

1. **Valid Instruct Following** is the score which refers to the number of test cases for which the model has followed the provided instruction out of total test cases. A model is said to be following an instruction if it generates the response in the expected format as per the instruction.

2. R-1, R-2, R-L and R-LSm refers to ROUGE-1, ROUGE-2, ROUGE-L and ROUGE-LSUM scores respectively[45].
3. BRT-P, BRT-R and BRT-F1 refers to BERT-Precision, BERT-Recall, and BERT-F1 Scores respectively[46].

4.4.1 EVALUATION OF PRE-TRAINED MODELS

CLASSIFICATION METRICS

Table 4.4: Pretrained Models: Valid Instruction Following, Accuracy, Precision, Recall, F1-score

Model Name	**Valid Instruction Following**	**Accuracy**	**Precision**	**Recall**	**F1-score**
Qwen-2 7B Instruct	**0.997**	0.05	0.253	0.05	0.064
Gemma 7B Instruct	0.134	0.057	0.191	0.057	0.074
Mistral 7B Instruct V0.2	0.982	0.057	0.192	0.057	0.065
Phi-3 Medium 4K Instruct	0.416	**0.154**	**0.446**	**0.154**	**0.179**
Medical LLaMA-3 8B	0.013	0.139	0.354	0.139	0.157
LLaMA-3 8B Instruct	0.751	0.107	0.237	0.107	0.123

TEXT TO TEXT EVALUATION METRICS

Table 4.5: Pretrained Models: Valid Instruction Following, ROUGE, and BERT Scores

Model Name	**Valid Instruction Following**	**R-1**	**R-2**	**R-L**	**R-LSm**	**BRT-P**	**BRT-R**	**BRT-F1**
Qwen-2 7B Instruct	**0.997**	0.2	0.074	0.2	0.199	0.828	0.858	0.842
Gemma 7B Instruct	0.134	0.162	0.066	0.163	0.162	0.838	0.842	0.84
Mistral 7B Instruct V0.2	0.982	0.154	0.065	0.155	0.155	0.835	0.842	0.838
Phi-3 Medium 4K Instruct	0.416	**0.298**	**0.08**	**0.298**	**0.297**	0.853	**0.872**	**0.862**
Medical LLaMA-3 8B	0.013	0.175	0.07	0.174	0.174	**0.855**	0.822	0.838
LLaMA-3 8B Instruct	0.751	0.237	0.071	0.238	0.238	0.839	0.869	0.853

CONCLUSIONS

- **Top Models - Valid Instruct Following** : **Qwen-2 7B Instruct** is the top performing pre-trained model in instruction following task with a score of **0.997**.
- **Top Models - Classification Metrics** : **Phi-3 Medium 4K Instruct** is the top performing pre-trained model according to all Classification Metrics.
- **Top Models - Text-to-Text Evaluation Metrics** : **Phi-3 Medium 4K Instruct** is the top performing pre-trained model according to all Text-to-Text Evaluation Metrics except BERT Precision, for which **Medical LLaMA-3 8B** performed best.
- **Best Model** : **Phi-3 Medium 4K Instruct** is the best pre-trained model as per the majority of the metrics.

4.4.2 EVALUATION OF FINE-TUNED MODELS

CLASSIFICATION METRICS

Table 4.6: Fine-tuned Models: Valid Instruction Following, Accuracy, Precision, Recall, F1-score

Model Name	**Valid Instruction Following**	**Accuracy**	**Precision**	**Recall**	**F1-score**
Qwen-2 7B Instruct	0.998	0.738	0.931	0.738	0.778
Gemma 7B Instruct	0.974	0.078	0.288	0.078	0.101
Mistral 7B Instruct V0.2	0.437	0.383	0.482	0.383	0.414
Phi-3 Medium 4K Instruct	0.968	0.015	0.081	0.015	0.02
Medical LLaMA-3 8B	0.995	0.057	0.204	0.057	0.069
LLaMA-3 8B Instruct	**1.0**	**0.961**	**0.955**	**0.961**	**0.957**

TEXT TO TEXT EVALUATION METRICS

Table 4.7: Fine-tuned Models: Valid Instruction Following, ROUGE, and BERT Scores

Model Name	**Valid Instruction Following**	**R-1**	**R-2**	**R-L**	**R-LSm**	**BRT-P**	**BRT-R**	**BRT-F1**
Qwen-2 7B Instruct	0.998	0.87	0.412	0.87	0.871	0.962	0.978	0.97
Gemma 7B Instruct	0.974	0.461	0.201	0.462	0.461	0.86	0.933	0.895
Mistral 7B Instruct V0.2	0.437	0.715	0.365	0.715	0.715	0.921	0.963	0.941
Phi-3 Medium 4K Instruct	0.968	0.482	0.222	0.482	0.481	0.859	0.942	0.898
Medical LLaMA-3 8B	0.995	0.487	0.223	0.487	0.487	0.862	0.943	0.9
LLaMA-3 8B Instruct	**1.0**	**0.966**	**0.449**	**0.966**	**0.966**	**0.994**	**0.994**	**0.994**

CONCLUSIONS

- **Top Models - Valid Instruct Following** : **LLaMA-3 8B Instruct** is the top performing fine-tuned model in instruction following task with a perfect score of **1**.
- **Top Models - Classification Metrics** : **LLaMA-3 8B Instruct** is the top performing fine-tuned model according to all Classification Metrics.
- **Top Models - Text-to-Text Evaluation Metrics** : **LLaMA-3 8B Instruct** is the top performing fine-tuned model according to all Text-to-Text Evaluation Metrics.
- **Best Model** : **LLaMA-3 8B Instruct** is the best fine-tuned model as per all the metrics.
- The high scores of **LLaMA-3 8B Instruct** after fine-tuning shows the success of fine-tuning in increasing accuracy of this model. It also demonstrates the incred-

ible capabilities of this model, to learn, understand and produce highly accurate outputs for domain specific tasks after proper fine-tuning.

4.4.3 EVALUATION OF FINE-TUNED MODELS WITH POST-PROCESSING OF OUTPUTS USING FAISS WITH CLINICAL BERT'S ENCODINGS

CLASSIFICATION METRICS

Table 4.8: Post-processed Fine-tuned Models: Valid Instruction Following, Accuracy, Precision, Recall, F1-score

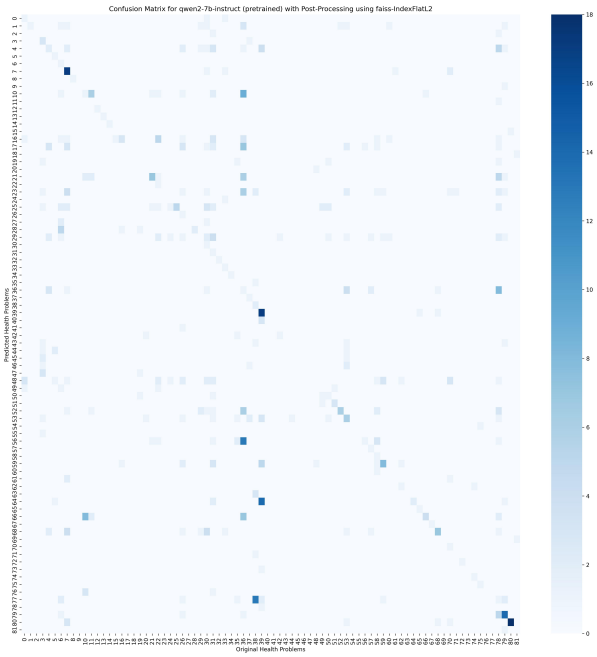
Model Name	**Valid Instruction Following**	**Accuracy**	**Precision**	**Recall**	**F1-score**
Qwen-2 7B Instruct	0.998	0.843	0.916	0.843	0.848
Gemma 7B Instruct	0.974	0.657	0.725	0.657	0.659
Mistral 7B Instruct V0.2	0.437	0.735	0.787	0.735	0.737
Phi-3 Medium 4K Instruct	0.968	0.673	0.783	0.673	0.682
Medical LLaMA-3 8B	0.995	0.756	0.863	0.756	0.766
LLaMA-3 8B Instruct	**1.0**	**0.968**	**0.957**	**0.968**	**0.961**

TEXT TO TEXT EVALUATION METRICS

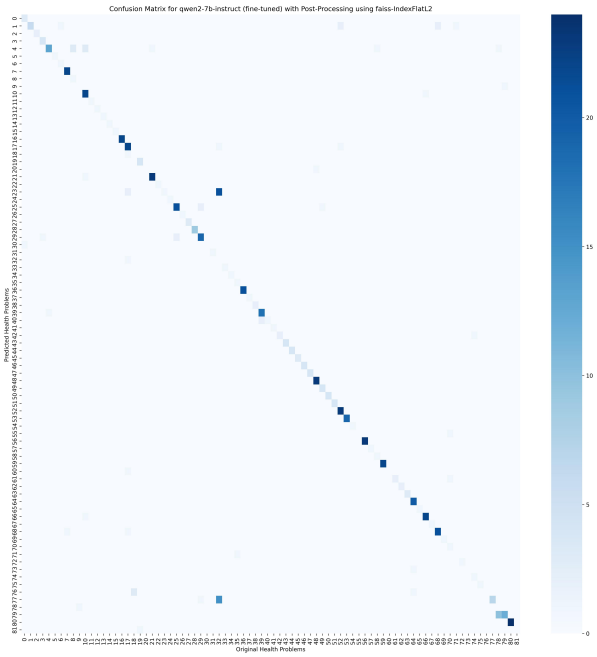
Table 4.9: Post-processed Fine-tuned Models: Valid Instruction Following, ROUGE, and BERT Scores

Model Name	**Valid Instruction Following**	**R-1**	**R-2**	**R-L**	**R-LSm**	**BRT-P**	**BRT-R**	**BRT-F1**
Qwen-2 7B Instruct	0.998	0.845	0.421	0.843	0.843	0.977	0.972	0.974
Gemma 7B Instruct	0.974	0.679	0.319	0.676	0.677	0.935	0.944	0.939
Mistral 7B Instruct V0.2	0.437	0.756	0.356	0.756	0.756	0.962	0.958	0.959
Phi-3 Medium 4K Instruct	0.968	0.688	0.35	0.685	0.688	0.946	0.947	0.946
Medical LLaMA-3 8B	0.995	0.795	0.359	0.794	0.794	0.957	0.967	0.961
LLaMA-3 8B Instruct	**1.0**	**0.969**	**0.448**	**0.968**	**0.969**	**0.994**	**0.994**	**0.994**

CONFUSION MATRICES FOR EACH MODEL

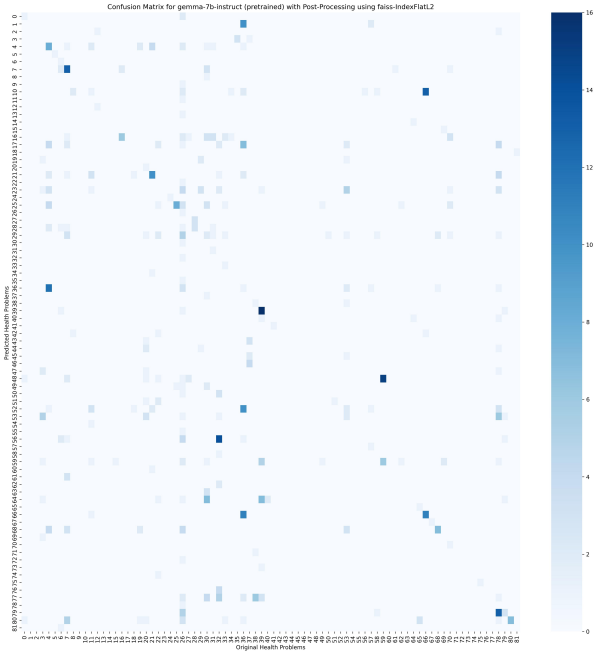


(a) Outputs from Pre-Trained Qwen-2 7B Instruct

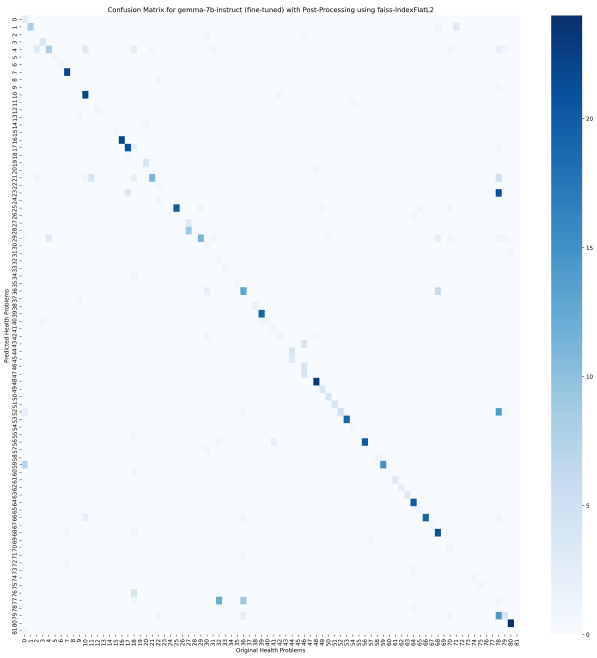


(b) Outputs from Fine-Tuned Qwen-2 7B Instruct

Figure 4.6: Confusion Matrix for Qwen-2 7B Instruct Outputs

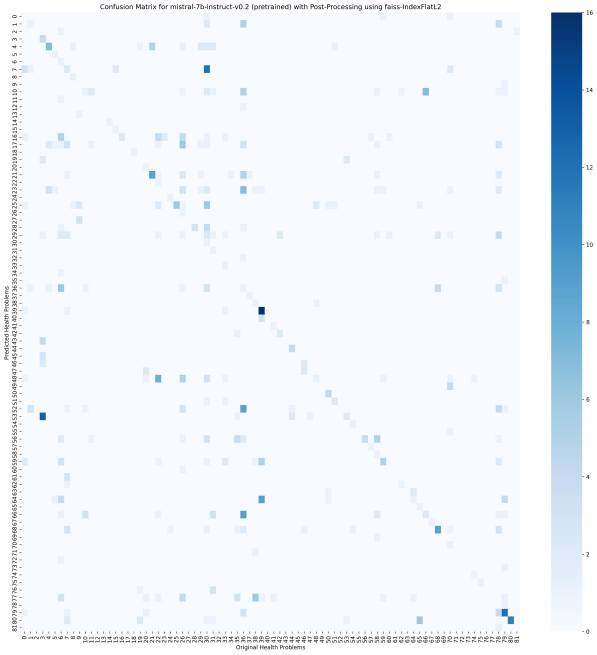


(a) Outputs from Pre-Trained Gemma 7B Instruct

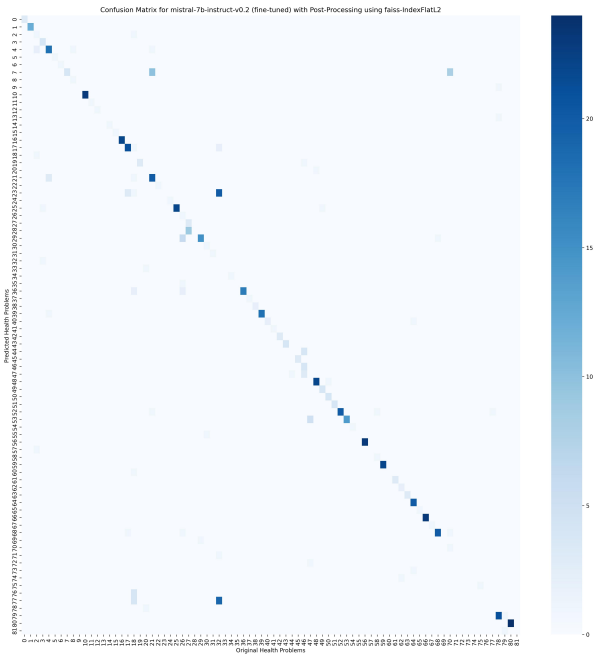


(b) Outputs from Fine-Tuned Gemma 7B Instruct

Figure 4.7: Confusion Matrix for Gemma 7B Instruct Outputs

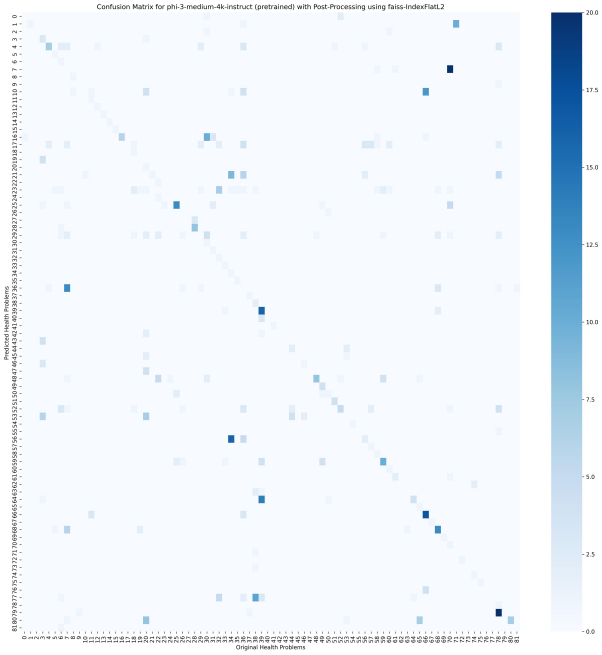


(a) Outputs from Pre-Trained Mistral 7B Instruct

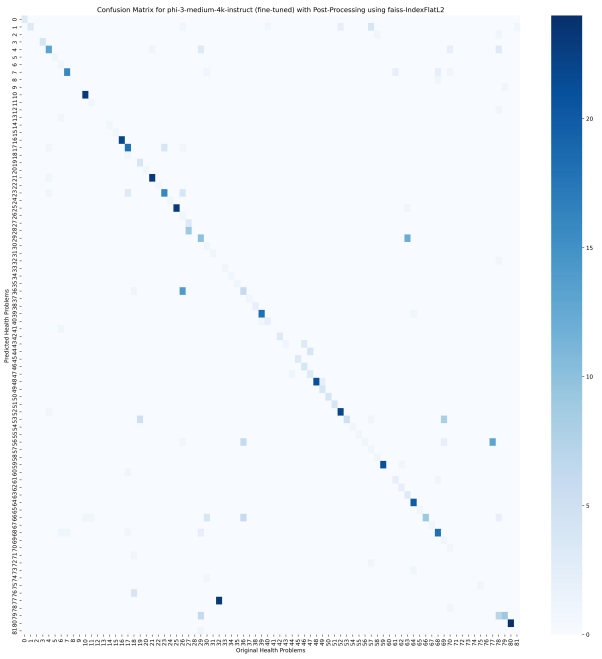


(b) Outputs from Fine-Tuned Mistral 7B Instruct

Figure 4.8: Confusion Matrix for Mistral 7B Instruct Outputs

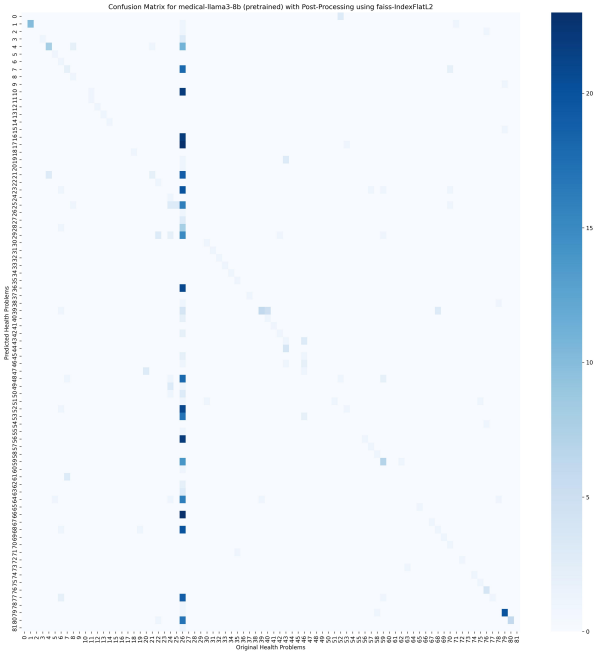


(a) Outputs from Pre-Trained Phi-3 Medium 4K Instruct

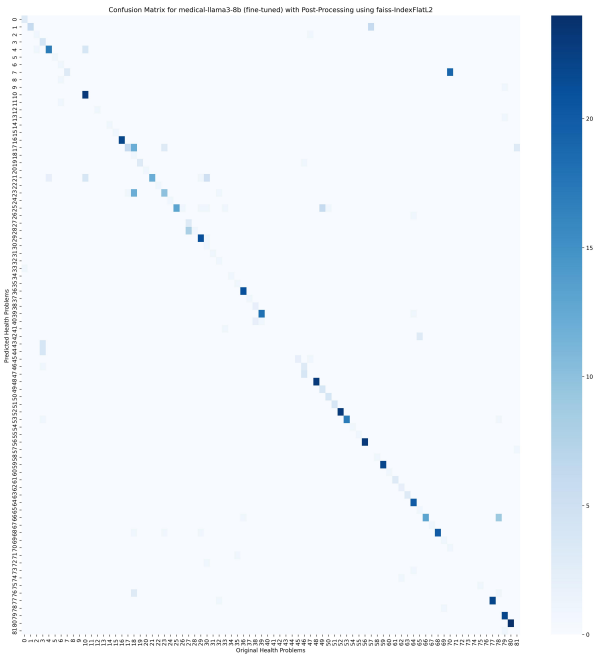


(b) Outputs from Fine-Tuned Phi-3 Medium 4K Instruct

Figure 4.9: Confusion Matrix for Phi-3 Medium 4K Instruct Outputs

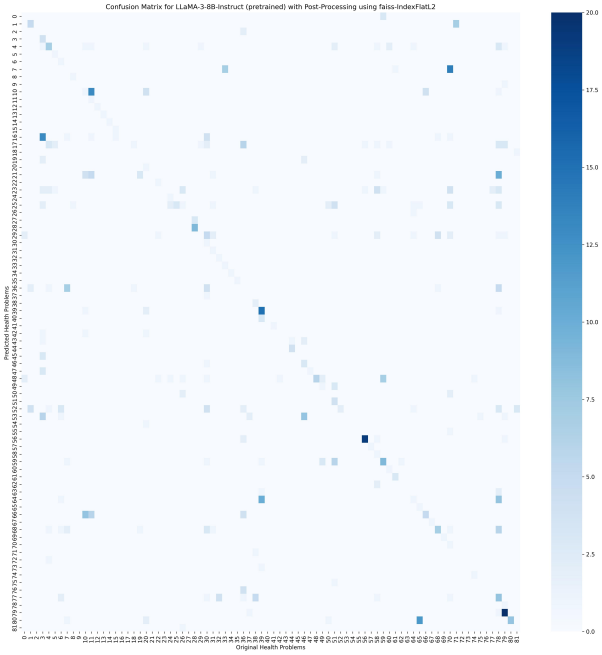


(a) Outputs from Pre-Trained Medical LLaMA-3 8B

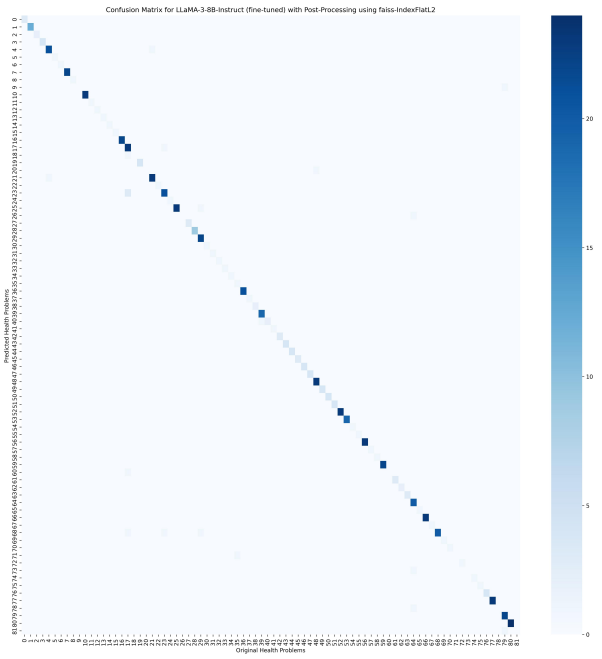


(b) Outputs from Fine-Tuned Medical LLaMA-3 8B

Figure 4.10: Confusion Matrix for Medical LLaMA-3 8B Outputs



(a) Outputs from Pre-Trained LLaMA-3 8B Instruct



(b) Outputs from Fine-Tuned LLaMA-3 8B Instruct

Figure 4.11: Confusion Matrix for LLaMA-3 8B Instruct Outputs

4.4.4 GRAPHICAL COMPARISONS

CLASSIFICATION METRICS COMPARISON USING BAR CHART

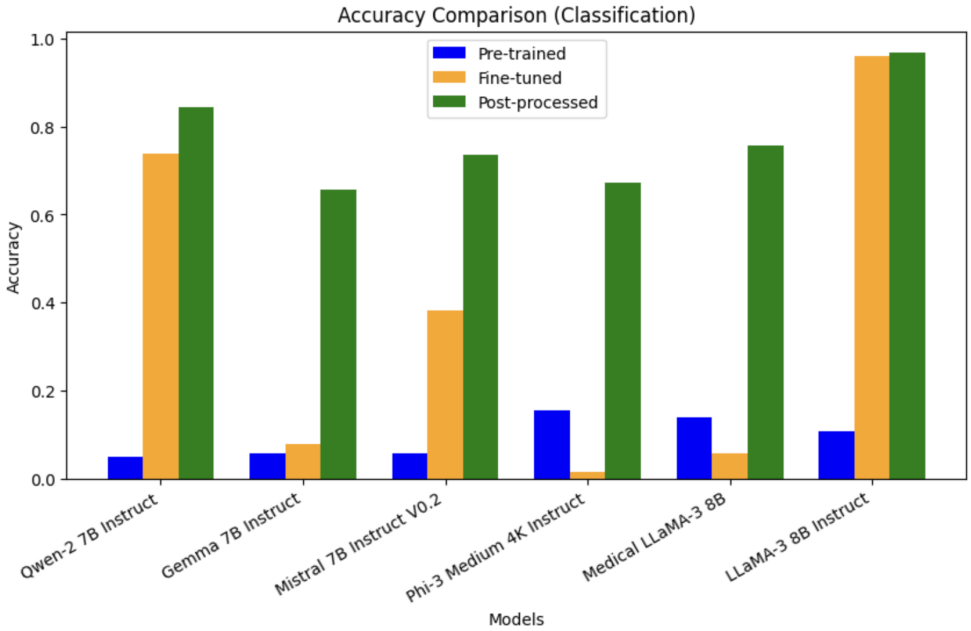


Figure 4.12: Accuracy Score for each Model

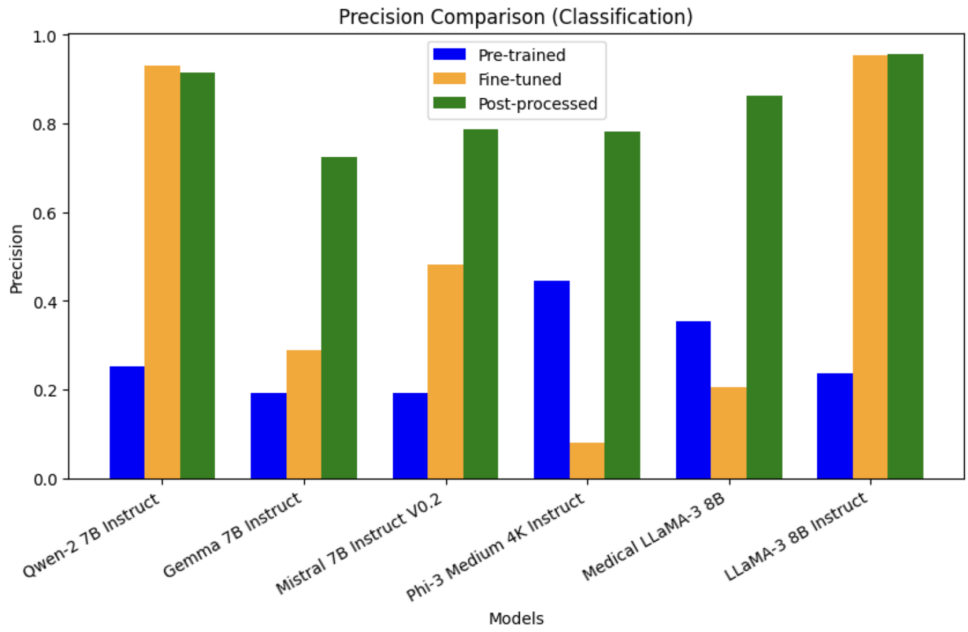


Figure 4.13: Precision Score for each Model

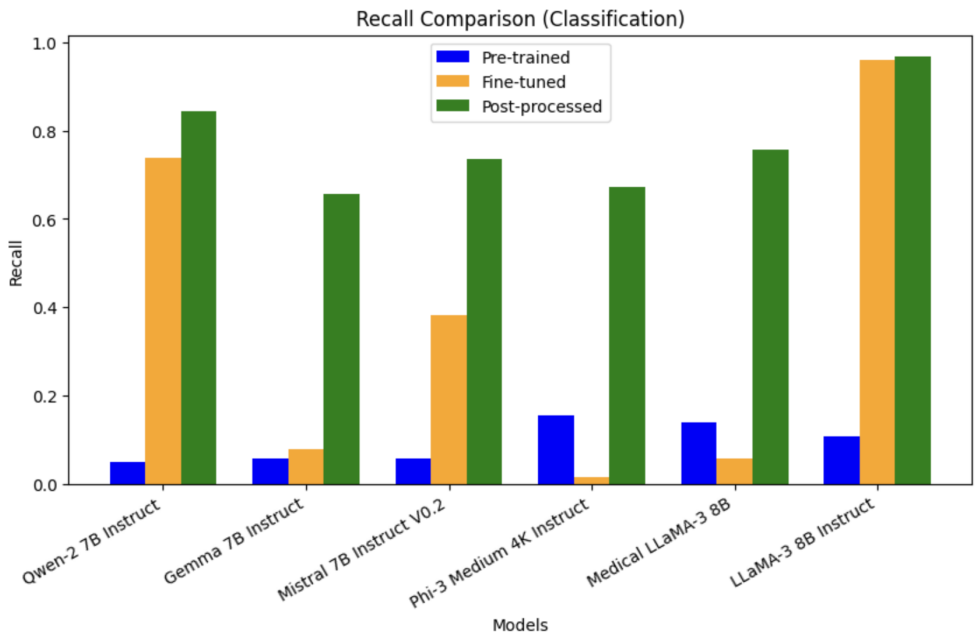


Figure 4.14: Recall Score for each Model

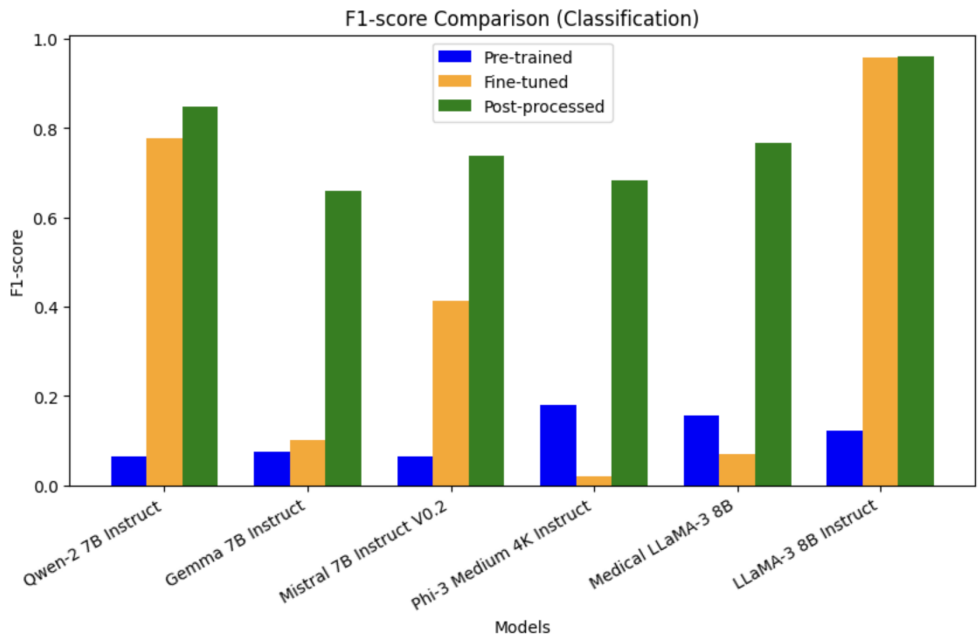


Figure 4.15: F1 Score for each Model

ACCURACY & F1-SCORE TRENDS ACROSS MODELS

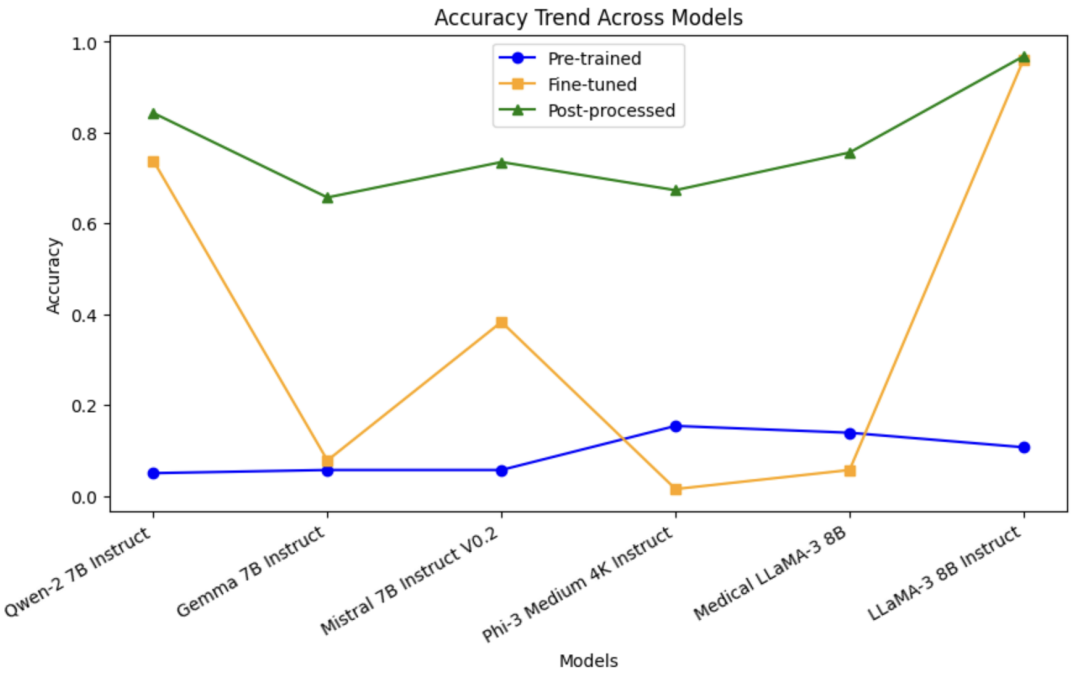


Figure 4.16: Accuracy Trends across Models

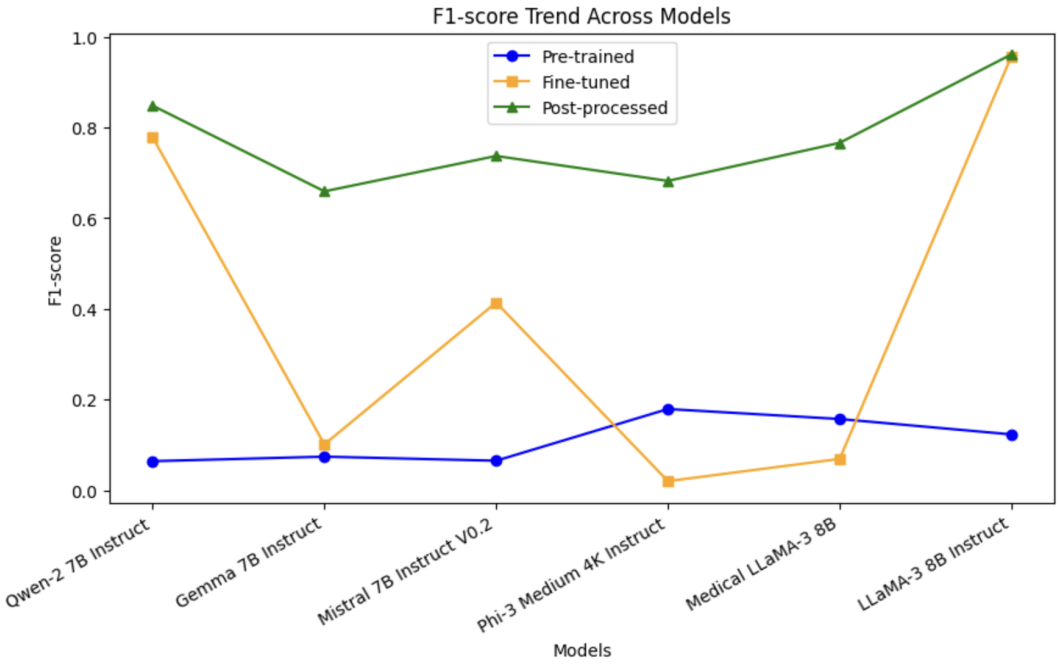


Figure 4.17: F1 Score Trends across Models

RADAR CHART FOR EVALUATION METRICS OF TOP 3 MODELS

QWEN-2 7B INSTRUCT

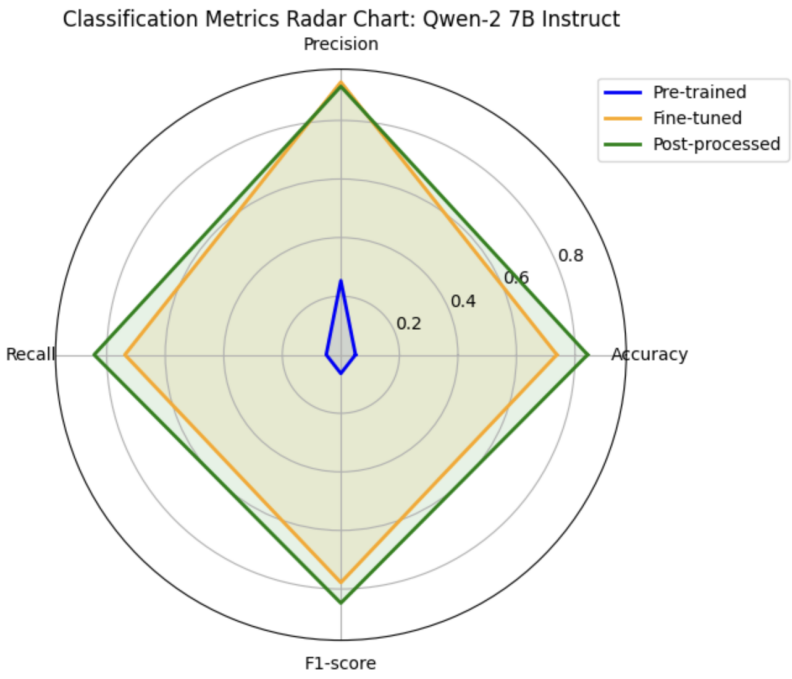


Figure 4.18: Classification Metrics Radar Chart : Qwen-2 7B Instruct

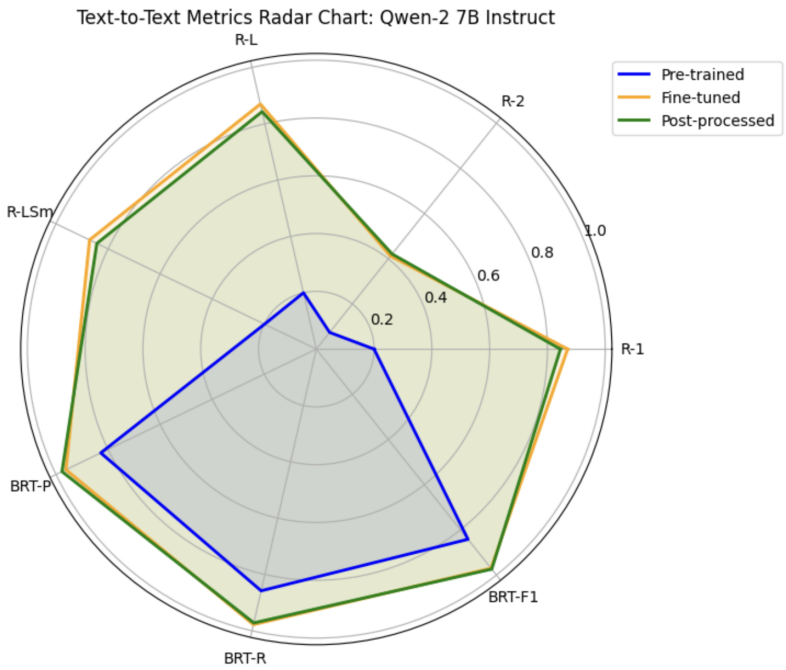


Figure 4.19: Text-to-Text Metrics Radar Chart : Qwen-2 7B Instruct

MISTRAL 7B INSTRUCT

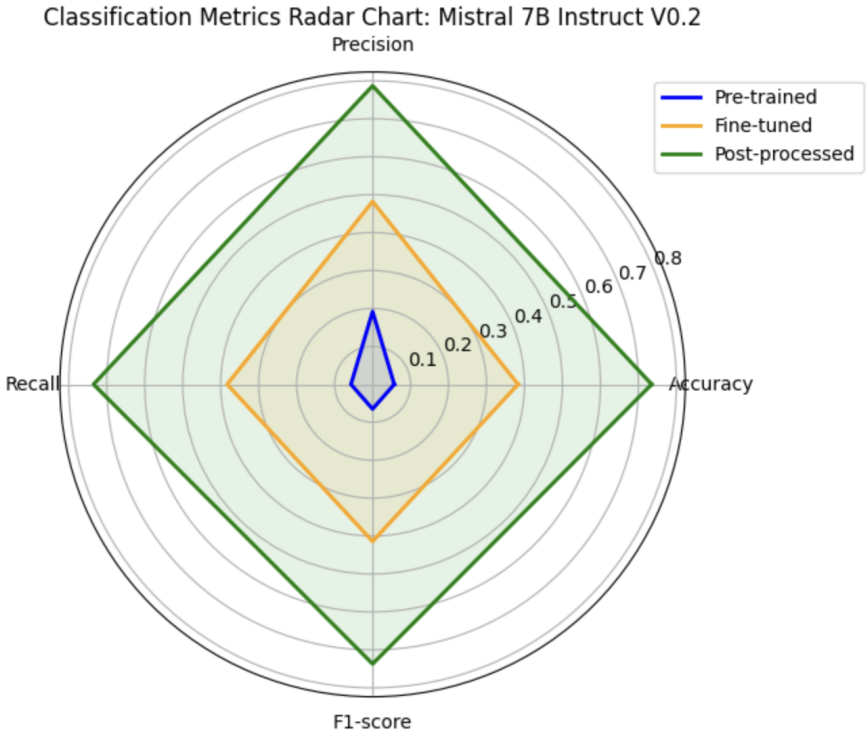


Figure 4.20: Classification Metrics Radar Chart : Mistral 7B Instruct

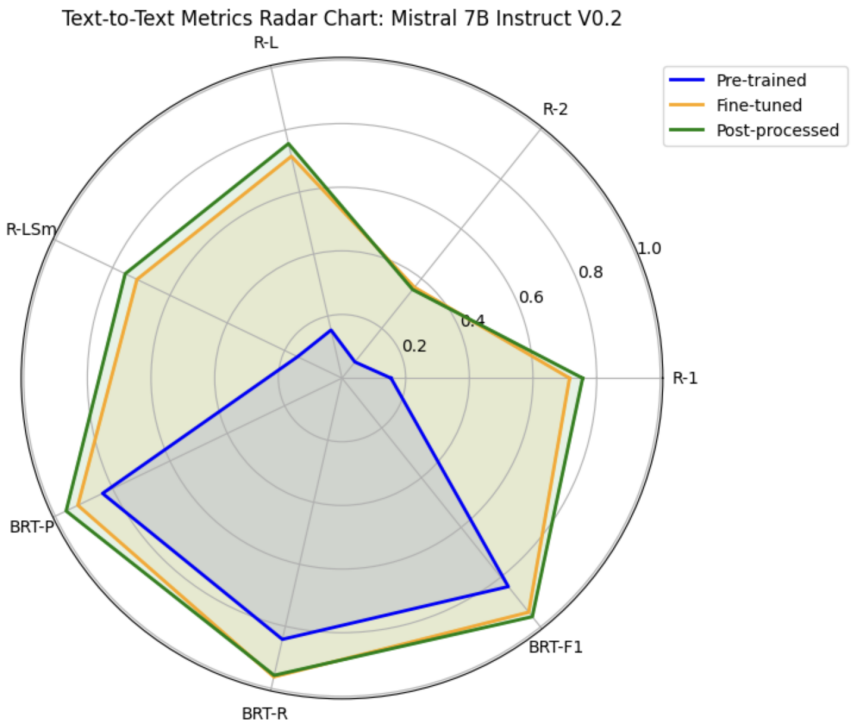


Figure 4.21: Text-to-Text Metrics Radar Chart : Mistral 7B Instruct

LLAMA-3 8B INSTRUCT

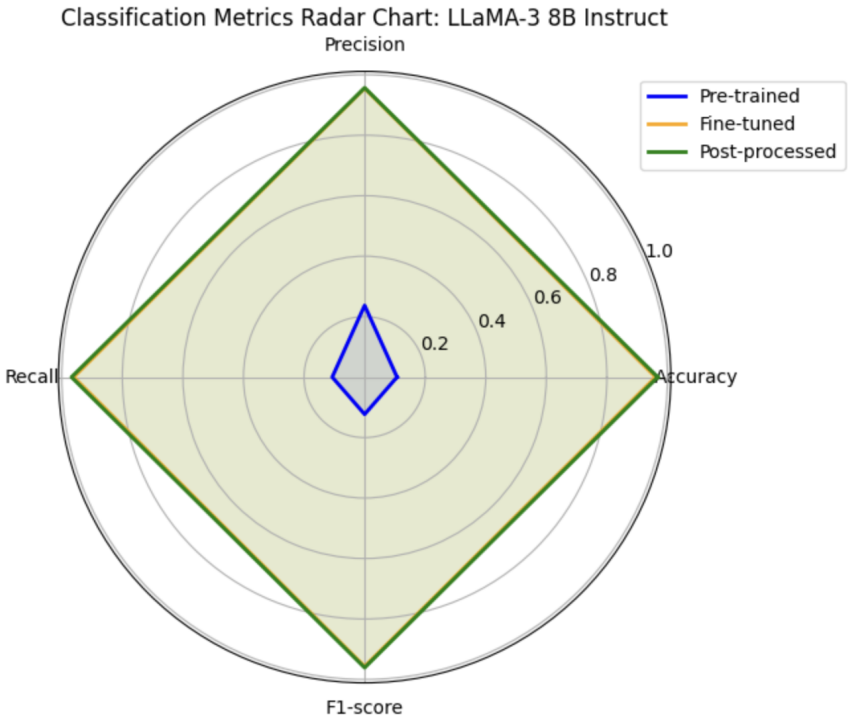


Figure 4.22: Classification Metrics Radar Chart : LLaMA-3 8B Instruct

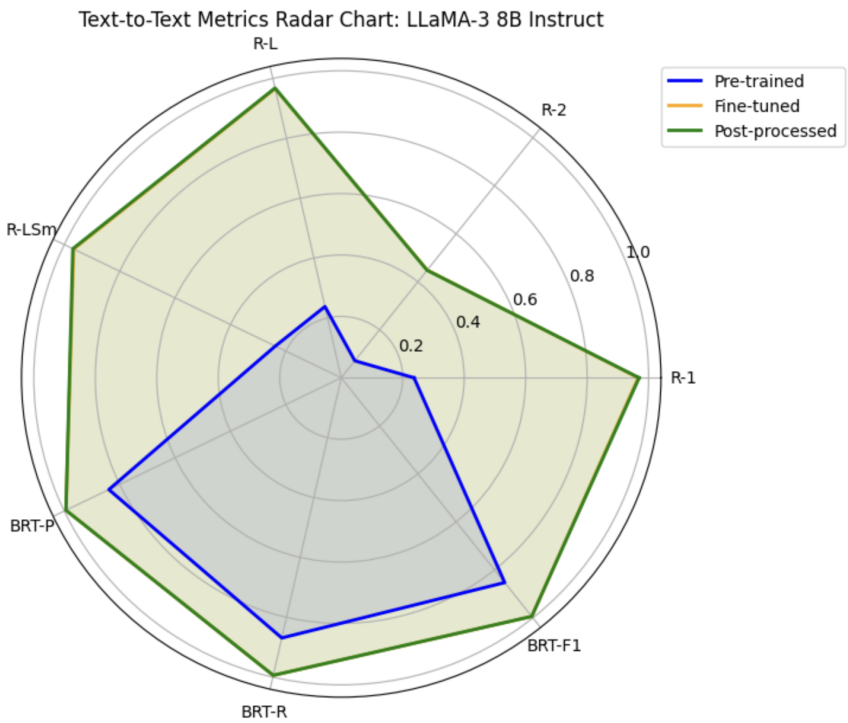


Figure 4.23: Text-to-Text Metrics Radar Chart : LLaMA-3 8B Instruct

Table 4.10: Summary of total improvements in score from non post-processed pre-trained models to post-processed fine-tuned models (average across all metrics)

Model	**Total Improvement (Average across Metrics)**
Qwen-2 7B Instruct	+0.644
Gemma 7B Instruct	+0.519
Mistral 7B Instruct V0.2	+0.606
Phi-3 Medium 4K Instruct	+0.427
Medical LLaMA-3 8B	+0.562
LLaMA-3 8B Instruct	**+0.759**

CONCLUSIONS

- **Top Models - Valid Instruct Following** : **LLaMA-3 8B Instruct** is the top performing fine-tuned model with post-processing in instruction following task with a perfect score of **1**.
- **Top Models - Classification Metrics** : **LLaMA-3 8B Instruct** is the top performing fine-tuned model with post-processing according to all Classification Metrics.
- **Top Models - Text-to-Text Evaluation Metrics** : **LLaMA-3 8B Instruct** is the top performing fine-tuned model with post-processing according to all Text-to-Text Evaluation Metrics.
- **Best Model** : **LLaMA-3 8B Instruct** is the best fine-tuned model with post-processing as per all the metrics.
- The further improved high scores (**+0.759**) of **LLaMA-3 8B Instruct** after further post-processing of fine-tuned model outputs shows the efficacy of post-processing along with fine-tuning in increasing accuracy of this model. Thus the post-processing using **FAISS** [9] **semantic-search** of relatable-labels [8] with **Clinical BERT's encodings** [10].

4.5 SUMMARY

4.5.1 BEST MODELS

LLaMA-3-8B-Instruct : Based on the evaluation metrics, LLaMA-3-8B-Instruct stands out as the best overall model after post-processing using **FAISS** [9] **semantic-search** of relatable-labels [8] with **Clinical BERT's encodings** [10]. It achieves the highest scores across accuracy, precision,recall, F1-score, ROUGE scores (except for ROUGE-2 by just a difference of 0.001), and BERT scores, making it the most reliable and effective model for the task at hand.

Here, is the summary for the best models in each category [Pre-trained (PT), Fine-Tuned (FT) and Fine-Tuned + Post-processing (FT + PP)] :-

CLASSIFICATION METRICS

Table 4.11: Best Models Evaluation Scores : Valid Instruction Following, Accuracy, Precision, Recall, F1-score

Model Name	**Valid Instruction Following**	**Accuracy**	**Precision**	**Recall**	**F1-score**
Phi-3 Medium 4K Instruct (PT)	0.416	0.154	0.446	0.154	0.179
LLaMA-3 8B Instruct (PT)	0.751	0.107	0.237	0.107	0.123
LLaMA-3 8B Instruct (FT)	1.0	0.961	0.955	0.961	0.957
LLaMA-3 8B Instruct (FT + PP)	**1.0**	**0.968**	**0.957**	**0.968**	**0.961**

TEXT-TO-TEXT EVALUATION METRICS

Table 4.12: Best Models Evaluation Scores: Valid Instruction Following (Valid Inst. Follow.), ROUGE, and BERT Scores

Model Name	**Valid Inst. Follow.**	**R-1**	**R-2**	**R-L**	**R-LSm**	**BRT-P**	**BRT-R**	**BRT-F1**
Phi-3 Medium 4K Instruct (PT)	0.416	0.298	0.08	0.298	0.297	0.853	0.872	0.862
Medical LLaMA-3 8B (PT)	0.013	0.175	0.07	0.174	0.174	0.855	0.822	0.838
LLaMA-3 8B Instruct (PT)	0.751	0.237	0.071	0.238	0.238	0.839	0.869	0.853
LLaMA-3 8B Instruct (FT)	1.0	0.966	**0.449**	0.966	0.966	0.994	0.994	0.994
LLaMA-3 8B Instruct (FT + PP)	**1.0**	**0.969**	0.448	**0.968**	**0.969**	**0.994**	**0.994**	**0.994**

CHAPTER 5

CONCLUSIONS AND FUTURE SCOPE

5.1 CONCLUSIONS

- Disease prediction model with high accuracy are in high demand in medical domain and is a great topic of research.
- Large Language Models can perform well in these task involving predictions on the basis of text, apart from the traditional Machine Learning Models.
- **Fine-Tuned LLaMA-3 8B with post-processed ouputs using FAISS and Clinical-BERT's encoding** turns out to be the best model. It achieves the highest scores across accuracy, precision,recall, F1-score, ROUGE scores (except for ROUGE-2 by just a difference of 0.001), and BERT scores, making it the most reliable and effective model for the task at hand.
- The experiments in this project demonstrate that fine-tuning along with post-processing of fine-tuned model outputs shows the efficacy of fine-tuning & post-processing in improving performances of models in classification tasks, which is also evident from the improved high scores (+0.759) of LLaMA-3 8B Instruct after fine-tuning and post-processing.

5.2 FUTURE SCOPE

- **Sequential Training** can be done on these fine-tuned models for other tasks benefitting from the knowledge gained by these models in the training.
- New models can be used for providing the encodings for tokenization in semantic-search with FAISS [9]. It will help in testing the effectiveness of those models in post-processing tasks.

REFERENCES

[1] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023.

[2] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7b," 2023.

[3] M. Abdin, S. A. Jacobs, A. A. Awan, J. Aneja, A. Awadallah, H. Awadalla, N. Bach, A. Bahree, A. Bakhtiari, J. Bao, and et al., "Phi-3 technical report: A highly capable language model locally on your phone," 2024.

[4] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, and et al., "Gemma: Open models based on gemini research and technology," 2024.

[5] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang, B. Hui, L. Ji, M. Li, J. Lin, R. Lin, D. Liu, G. Liu, C. Lu, K. Lu, J. Ma, R. Men, X. Ren, X. Ren, C. Tan, S. Tan, J. Tu, P. Wang, S. Wang, W. Wang, S. Wu, B. Xu, J. Xu, A. Yang, H. Yang, J. Yang, S. Yang, Y. Yao, B. Yu, H. Yuan, Z. Yuan, J. Zhang, X. Zhang, Y. Zhang, Z. Zhang, C. Zhou, J. Zhou, X. Zhou, and T. Zhu, "Qwen technical report," 2023.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[7] M. Ali, M. Fromm, K. Thellmann, R. Rutmann, M. Lübbering, J. Leveling, K. Klug, J. Ebert, N. Doll, J. S. Buschhoff, C. Jain, A. A. Weber, L. Jurkschat, H. Abdelwahab, C. John, P. O. Suarez, M. Ostendorff, S. Weinbach, R. Sifa, S. Kesselheim, and N. Flores-Herr, "Tokenizer choice for llm training: Negligible or crucial?" 2024.

[8] D. Chandrasekaran and V. Mago, "Evolution of semantic similarity—a survey," *ACM Computing Surveys*, vol. 54, no. 2, p. 1–37, Feb. 2021. [Online]. Available: http://dx.doi.org/10.1145/3440755

[9] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, "The faiss library," 2024.

[10] K. Huang, J. Altosaar, and R. Ranganath, "Clinicalbert: Modeling clinical notes and predicting hospital readmission," 2020.

[11] National Health Systems Resource Centre, "Emergency medical service (ems) in india: A concept paper," https://nhsrcindia.org/sites/default/files/2021-02/Emergency_Medical_Service_in_India_Concept_Paper.pdf.

[12] A. Sahu, S. Bhoi, P. Aggarwal, J. Nayer, L. murmu, M. Ekka, A. Kumar, S. Galwankar, and T. Sinha, "Aiims triage protocol - atp of a busy ed," *Journal of Emergencies Trauma and Shock*, 01 2019. [Online]. Available: https://www.researchgate.net/publication/336902793_AIIMS_Triage_Protocol_-_ATP_of_a_busy_ED

[13] K. Shailaja, B. Seetharamulu, and M. A. Jabbar, "Machine learning in healthcare: A review," in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 2018, pp. 910–914.

[14] S. Alajmani and H. Elazhary, "Hospital readmission prediction using machine learning techniques," *International Journal of Advanced Computer Science and Applications*, vol. 10, 01 2019.

[15] G. Fitzgerald, C. Robertson, K. Little, and A. Dove, "The urgency distribution of an accident and emergency department's workload," *Archives of emergency medicine*, vol. 3, pp. 225–30, 01 1987.

[16] J. J. Guterman, J. B. Franaszek, D. Murdy, and M. Gifford, "The 1980 patient urgency study: Further analysis of the data," *Annals of Emergency Medicine*, vol. 14, no. 12, pp. 1191–1198, 1985. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0196064485810283

[17] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: https://arxiv.org/abs/1810.04805

[18] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[19] K. S. Kalyan, "A survey of gpt-3 family large language models including chatgpt and gpt-4," *Natural Language Processing Journal*, vol. 6, p. 100048, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2949719123000456

[20] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat *et al.*, "Gpt-4 technical report," 2024.

[21] X. Amatriain, A. Sankar, J. Bing, P. K. Bodigutla, T. J. Hazen, and M. Kazi, "Transformer models: an introduction and catalog," 2024.

[22] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," 2020.

[23] Y. He, Z. Zhu, Y. Zhang, Q. Chen, and J. Caverlee, "Infusing disease knowledge into bert for health question answering, medical inference and disease name recognition," 2020.

[24] C. Li, Y. Zhang, Y. Weng, B. Wang, and Z. Li, "Natural language processing applications for computer-aided diagnosis in oncology," *Diagnostics*, vol. 13, no. 2, 2023. [Online]. Available: https://www.mdpi.com/2075-4418/13/2/286

[25] N. A. I. Omoregbe, I. O. Ndaman, S. Misra, O. O. Abayomi-Alli, and R. Damaševičius, "Text messaging-based medical diagnosis using natural language processing and fuzzy logic," *Journal of Healthcare Engineering*, vol. 2020, no. 1, p. 8839524, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1155/2020/8839524

[26] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "BioBERT: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 09 2019. [Online]. Available: https://doi.org/10.1093/bioinformatics/btz682

[27] E. Alsentzer, J. R. Murphy, W. Boag, W.-H. Weng, D. Jin, T. Naumann, and M. B. A. McDermott, "Publicly available clinical bert embeddings," 2019.

[28] I. Beltagy, K. Lo, and A. Cohan, "SciBERT: A pretrained language model for scientific text," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3615–3620. [Online]. Available: https://aclanthology.org/D19-1371

[29] Y. Gu, R. Tinn, H. Cheng, M. Lucas, N. Usuyama, X. Liu, T. Naumann, J. Gao, and H. Poon, "Domain-specific language model pretraining for biomedical natural language processing," *ACM Trans. Comput. Healthcare*, vol. 3, no. 1, oct 2021. [Online]. Available: https://doi.org/10.1145/3458754

[30] A. E. Johnson, T. J. Pollard, L. Shen, L.-w. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Anthony Celi, and R. G. Mark, "MIMIC-III, a freely accessible critical care database," *Scientific Data*, vol. 3, no. 1, p. 160035, 2016, published online 24 May 2016. [Online]. Available: https://doi.org/10.1038/sdata.2016.35

[31] A. Gilson, C. W. Safranek, T. Huang, V. Socrates, L. Chi, R. A. Taylor, and D. Chartash, "How does chatgpt perform on the united states medical licensing examination? the implications of large language models for medical education and knowledge assessment," *JMIR Med Educ*, vol. 9, p. e45312, Feb 2023. [Online]. Available: https://mededu.jmir.org/2023/1/e45312

[32] Y. Li, Z. Li, K. Zhang, R. Dan, S. Jiang, and Y. Zhang, "Chatdoctor: A medical chat model fine-tuned on a large language model meta-ai (llama) using medical domain knowledge," 2023.

[33] A. J. Thirunavukarasu, R. Hassan, S. Mahmood, R. Sanghera, K. Barzangi, M. El Mukashfi, and S. Shah, "Trialling a large language model (chatgpt) in general practice with the applied knowledge test: Observational study demonstrating opportunities and limitations in primary care," *JMIR Med Educ*, vol. 9, p. e46599, Apr 2023. [Online]. Available: https://mededu.jmir.org/2023/1/e46599

[34] C. Xu, D. Guo, N. Duan, and J. McAuley, "Baize: An open-source chat model with parameter-efficient tuning on self-chat data," 2023.

[35] A. Ben Abacha and D. Demner-Fushman, "A question-entailment approach to question answering," *BMC Bioinformatics*, vol. 20, no. 1, p. 511, 10 2019. [Online]. Available: https://doi.org/10.1186/s12859-019-3119-4

[36] Z. Xue, C. Xu, and X. Xu, "Application of chatgpt in natural disaster prevention and reduction," *Natural Hazards Research*, vol. 3, no. 3, pp. 556–562, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666592123000744

[37] J. Wang, G. Zhang, W. Wang, K. Zhang, and Y. Sheng, "Cloud-based intelligent self-diagnosis and department recommendation service using chinese medical bert," *Journal of Cloud Computing*, vol. 10, no. 1, p. 4, 01 2021. [Online]. Available: https://doi.org/10.1186/s13677-020-00218-2

[38] Y. Shen, L. Heacock, J. Elias, K. Hentel, B. Reig, G. Shih, and L. Moy, "Chatgpt and other large language models are double-edged swords," *Radiology*, vol. 307, 01 2023.

[39] X. Yang, A. Chen, N. PourNejatian, H. C. Shin, K. E. Smith, C. Parisien, C. Compas, C. Martin, A. B. Costa, M. G. Flores, Y. Zhang, T. Magoc, C. A. Harle, G. Lipori, D. A. Mitchell, W. R. Hogan, E. A. Shenkman, J. Bian, and Y. Wu, "A large language model for electronic health records," *npj Digital Medicine*, vol. 5, no. 1, p. 194, 12 2022. [Online]. Available: https://doi.org/10.1038/s41746-022-00742-2

[40] L. Xu, H. Xie, S.-Z. J. Qin, X. Tao, and F. L. Wang, "Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment," 2023.

[41] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021.

[42] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient fine-tuning of quantized llms," 2023.

[43] Y. Li, Y. Yu, C. Liang, P. He, N. Karampatziakis, W. Chen, and T. Zhao, "Loftq: Lora-fine-tuning-aware quantization for large language models," 2023.

[44] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, "A systematic survey of prompt engineering in large language models: Techniques and applications," 2024.

[45] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," 01 2004, p. 10.

[46] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," 2020.

[47] C. Jeong, "Domain-specialized llm: Financial fine-tuning and utilization method using mistral 7b," *Journal of Intelligence and Information Systems*, vol. 30, no. 1, p. 93–120, Mar. 2024. [Online]. Available: http://dx.doi.org/10.13088/jiis.2024.30.1.093

[48] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian, "A comprehensive overview of large language models," 2024. [Online]. Available: https://arxiv.org/abs/2307.06435

[49] Y. Sheng, S. Cao, D. Li, C. Hooper, N. Lee, S. Yang, C. Chou, B. Zhu, L. Zheng, K. Keutzer, J. E. Gonzalez, and I. Stoica, "S-lora: Serving thousands of concurrent lora adapters," 2024.

[50] N. Kazi and I. Kahanda, "Enhancing transfer learning of llms through fine- tuning on task - related corpora for automated short-answer grading," in *2023 International Conference on Machine Learning and Applications (ICMLA)*, 2023, pp. 1687–1691.

Important Notice

This document has been **digitally signed** using **DigiLocker eSign** by **Saurabh Gupta** on **March 31, 2025**.

The digital signature confirms the authenticity and originality of this work. Unauthorized copying, modification, or distribution of any part of this document is strictly prohibited.

Any attempt to alter this document will invalidate the digital signature. Verification can be done through the embedded signature.