(DESIGN OF GENAI SOLUTION FOR DOMAIN SPECIFIC APPLICATION)

AIMLCZG628T DISSERTATION

Submitted in partial fulfillment of the requirements of the
MTech Data Science and Engineering Degree programme

by

(MANOJ KUMAR)

(BITS ID NO. 2022AA05373)

under the supervision of

(DIBYAJYOTI GOSWAMI)

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
Pilani (Rajasthan) INDIA

MAY, 2024

ACKNOWLEDGEMENT

First and foremost, I am deeply grateful to my M.Tech. Supervisor, Mr. Dibyajyoti Goswami, for his invaluable advice and steadfast support throughout my studies. I am also thankful to him for serving as the course mentor and for his insightful guidance on technical matters, without which this project would have been an incredibly challenging task. I extend my sincere appreciation to my team at the Military College of Telecommunication and Engineering, Mhow, for their continuous encouragement and support, which played a crucial role in the completion of my studies.

Lastly, I express my heartfelt gratitude to my parents for their relentless emotional support and unwavering encouragement throughout this journey. Their love, faith, and constant reassurance have been my guiding light, making this accomplishment possible.

ABSTRACT

Recent releases of pre-trained large language models (LLMs) have gained considerable traction, yet research on fine-tuning and employing domain-specific LLMs remains scarce. The swift progress in artificial intelligence (AI) and natural language processing (NLP) has driven the creation of advanced large language models (LLMs) capable of handling a wide range of tasks, such as generating text, summarizing content, and answering questions. These advancement have been propelled by innovation in transformer architecture and extensive training on diverse datasets. This dissertation aims to design, implement, and evaluate a private question-answering (Q&A) model tailored to a specific domain, leveraging the capabilities of LLMs while addressing the unique challenges associated with privacy and data security.

The research begins with a comprehensive literature review to understand the historical context, evolution, and technological advancements of LLMs. This phase involves analyzing key papers and identifying trends and gaps in current research, with a particular focus on transformer architecture, training methodologies, and the ethical considerations surrounding the deployment of LLMs. This foundation sets the stage for the technical exploration and requirement analysis necessary for developing a domain-specific Q&A model.

In the subsequent phase, the dissertation delves into the technical intricacies of transformer-based models. A detailed examination of their strengths and limitations is conducted, with an emphasis on understanding the computational requirements and ethical implications of deploying such models in real-world applications. This phase includes a thorough requirement analysis to identify the specific needs for developing a private Q&A model, including data sources, computational resources, and the technical infrastructure necessary for model customization and deployment.

The design and implementation phase outlines the strategic approach for training the Q&A model. This involves a multi-step process starting with data collection and pre-processing, followed by data augmentation to enhance the model's performance. The core of this phase is the implementation of a robust training pipeline, which includes model customization, hyper parameter tuning, and optimization techniques tailored to the specific domain. The goal is to develop a prototype that demonstrates the feasibility and potential of the proposed model.

Evaluation and testing are critical to validate the effectiveness of the Q&A model. This phase establishes rigorous evaluation metrics and benchmarks to assess the model's accuracy, efficiency, and reliability. The dissertation compares the performance of the developed model against existing public models to highlight its strengths and areas for improvement. Detailed testing ensures that the model meets the defined standards and can be reliably deployed in the target domain.

Given the increasing concerns over data privacy and security, the dissertation addresses these challenges by implementing robust security measures. This phase involves identifying potential privacy risks, developing strategies to mitigate them, and ensuring compliance with relevant regulatory frameworks. The goal is to protect sensitive data while maintaining the model's performance and utility.

The final phase of the dissertation involves comprehensive documentation of the development process, including design decisions, challenges faced, and solutions implemented. The documentation

serves as a valuable resource for future research and development in the field of domain-specific Q&A models. The dissertation concludes with the preparation of the final document and presentation materials for dissemination.

In summary, this dissertation provides a structured approach to developing a private Q&A system using large language model, addressing both technical and ethical considerations. This will enhance the knowledge in NLP and AI, offer practical insights and solutions for deploying LLMs in specialized domains while ensuring data privacy and security.

TABLE CONTENT

LIST OF ABBREVIATION

LLM	large language models
NLP	Natural Language Processing
RNN	Recurrent Neural Networks
LSTM	Long Short-Term Memory
BERT	Bidirectional Encoder Representations From Transformers
GloVe	Global Vector for Word Representation
GPT	Generative Pre-trained Transformer
GDPR	General Data Protection Regulation
Q&A	Question-Answering
UI/UX	User Interface/User Experience
SQuAD	Stanford Question Answering Dataset
LFS	Large File Storage
LoRA	Low Rank Adaptation
PEFT	Parameter-Efficient Fine-Tuning
NLTK	Natural Language Toolkit
PyMuPDF	Python library for data extraction, analysis, conversion & manipulation of PDF
TPU	Tensor Processing Unit
GPU	Graphic Processing Unit
CPU	Central Processing Unit
RAG	Retrieval Augmented Generation
GUI	Graphical User Interface

List of Figures

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Background

1.1.1 Evolution of Large Language Model

The emergence of large language models (LLMs) marks a pivotal advancement in artificial intelligence (AI) and natural language processing (NLP). Early NLP approaches, such as bag-of-words and TF-IDF, laid the groundwork for language modeling but were limited in their capacity to grasp the nuanced complexities of human language. The advent of neural networks, especially recurrent neural networks (RNNs) and long short-term memory (LSTM) networks, significantly enhanced the ability to capture contextual information across sequences of words. Despite these improvements, RNNs and LSTMs still faced challenges related to managing long-term dependencies and scaling effectively.

The advent of transformer architecture as introduced by [1], revolutionized NLP by addressing these limitations. Transformers leverage self-attention mechanisms to weigh the importance of different words in a sentence, allowing for better handling of sequential analysis and parallelization during training. This innovation paved the way for the creation of sophisticated LLMs like GPT (Generative Pre-trained Transformer) series by OpenAI, BERT (Bidirectional Encoder Representations from Transformers) by Google, and their successors. These models demonstrated unprecedented capabilities in text generation, summarization, translation, and question answering, largely due to their training on vast and diverse datasets.

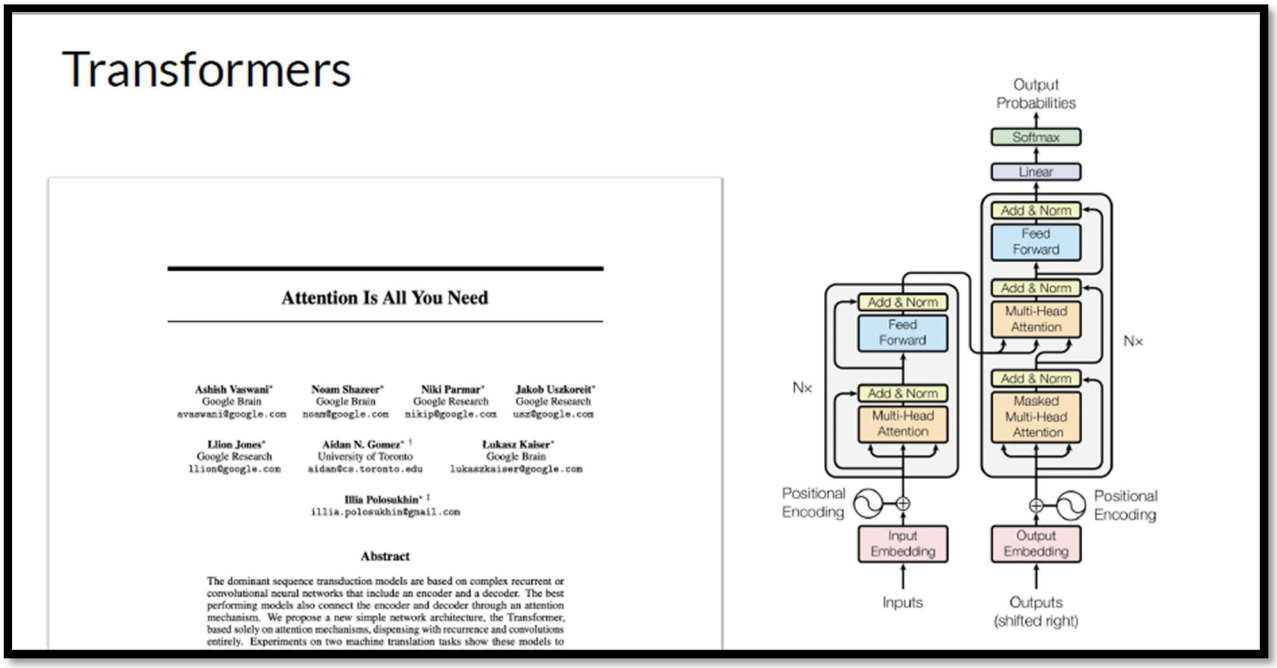


Figure 1.1.1-1 Transformer Architecture

1.1.2 The Rise of Domain-Specific LLMs

Despite the general success of LLMs, their application in specific domains often reveals limitations in performance and relevance. General-purpose LLMs, while versatile, may not possess the nuanced understanding required for specialized fields such as Telecommunication ,medicine, law, finance or specific task that are related to organization we can say publically non availability of data. This gap has led to growing interest in fine-tuning pre-trained model or fine tune LLMs for domain-specific applications. Fine-tuning involves training an existing LLM smaller dataset, leveraging hardware tailored to the organization's need. This process enhance the model performance in that particular area of interest and align it more closely with the organization's requirement.

Research on domain-specific LLMs is still emerging, with studies exploring their potential to improve accuracy and reliability in specialized tasks. For instance, BioBERT has been fine-tuned for biomedical text mining, and LegalBERT for legal document processing. Recently, Meta AI introduced the LLaMA model, which has brought significant advancements in Natural Language Processing (NLP) over text. These models have shown promising results, but there remains a considerable need for further research and development to fully harness the capabilities of LLMs across various domains.

However, public model like Llama3 are trained on public data but the organization has their internal requirement that may not be the part of pretrained model, second the sharing of information over internet is not safe. These all constraint bound a organization that required a model that can maintain the confidentiality and privacy of data. There are three approaches we can adopt for same implementation:-

- Fine-tuning – Continue train public model with our own data.
- Prompt Engineering – Understand the model input prompt to influence the model's output.
- Retrieval-Augmented Generation (RAG) pipeline – Used to enhance model with our data efficiently and securely.

1.1.3 Privacy and Ethical Considerations

As the deployment of domain-specific Large Language Model (LLMs) become more prevalent, addressing privacy and ethical consideration is crucial. LLMs require vast amounts of data for training, raising questions about the handling of sensitive and personal information. Ensuring data privacy is crucial, especially in domains like healthcare and finance where data breaches can have severe consequences. Researchers and practitioners must implement robust security measures and adhere to regulatory frameworks to protect user data. Ensuring the privacy and security of organization data is paramount.

1.1.3.1 Privacy Concerns

i. **Data Anonymization**: To protect organization privacy, data used for training and fine-tuning Large Language Model (LLM) should be anonymized. Removing personally

identifiable information (PII) from dataset help us to prevent unintended disclosure of sensitive data.

ii. **Data Security**: Robust security measures must be in place to safeguard data throughout its lifecycle. This includes secure storage, encryption, and access controls to prevent unauthorized access and data breaches.

iii. **Compliance with Regulations**: Organization must comply with data protection regulation such as GDPR and which impose strict guidelines on how data should be collected, processed, and stored.

1.1.3.2 Ethical Considerations

i. **Bias and Fairness**: Large Language Model (LLM) can inadvertently perpetuate biases present in the training data, leading to unfair outcomes. Continuous monitoring and mitigation strategies, such as bias audits and fairness metrics, are necessary to ensure equitable treatment across different demographic groups.

ii. **Impact on Employment**: The adoption of LLMs in specific domains can have implications for employment, potentially displacing certain job roles. Organizations should consider strategies for workforce transition, such as retraining programs and support for affected employees.

1.1.4 Objectives of the Dissertation

This dissertation aims to address the challenges and opportunities presented by domain-specific LLMs. By designing, implementing, and evaluating a private question-answering (Q&A) model tailored to a specific domain, this research seeks to advance the understanding of fine-tuning LLMs for specialized applications. The objectives include:

i. Conducting a comprehensive literature review to contextualize the evolution and advancements of LLMs.

ii. Analyzing the strengths and limitations of transformer-based models, particularly in the context of computational requirements and ethical implications.

iii. Developing a strategic approach for training a domain-specific Q&A model, encompassing data collection, pre-processing, and augmentation.

iv. Implementing a robust training pipeline with model customization and optimization tailored to the specific domain.

v. Establishing rigorous evaluation metrics to validate the model's performance against existing public models.

vi. Ensuring data privacy and security through the development of robust security measures and compliance with regulatory standards.

Through this structured approach, the dissertation aims to contribute to the growing body of knowledge in NLP and AI, offering practical insights and solutions for deploying LLMs in specialized domains while ensuring privacy and security of our data.

1.2 Problem statement and related work in this field

1.2.1 Problem Statement:

The rapid progress in artificial intelligence (AI) and natural language processing (NLP) has resulted in the creation of large language models (LLMs) that can handle a diverse array of tasks. Although models like GPT-3, BERT, and their successors have shown remarkable abilities, their effectiveness in domain-specific contexts often falls short. General-purpose LLMs lack the depth of understanding necessary for specialized areas, leading to issues with accuracy, relevance, and reliability when applied to fields such as medicine, law, and finance.

Furthermore, the deployment of LLMs in these domains raises significant concerns regarding data privacy and security. LLMs require extensive training on large datasets, which often contain sensitive information. It is very crucial to maintain the confidentiality and integrity of data related to organization where data privacy matter.

This dissertation addresses these challenges by designing, implementing, and evaluating a private question-answering (Q&A) model tailored to a specific domain. The primary research questions are:

i. How can LLMs be effectively fine-tuned to improve their performance in domain-specific applications?
ii. What are the technical and computational requirements for developing a private domain-specific Q&A model?
iii. How can privacy and data security be ensured during the training and deployment of such models?

1.2.2 Related Work

1.2.2.1 Transformer-Based Model Evaluation

The debut of transformer technology, as discussed in [1], represented a significant milestone in the field of Natural Language Processing (NLP). By employing self-attention mechanisms, transformers are able to handle input data concurrently. This not only enhances processing efficiency but also improves the model's ability to recognize distant relationships within text, laying the groundwork for numerous advanced large language models (LLMs) such as GPT and BERT, among others.

Developed by OpenAI, the Generative Pre-trained Transformer (GPT) series, especially the GPT-3 with its vast 175 billion parameters, demonstrates impressive abilities in both generating and understanding text. Despite its versatility in handling diverse tasks with minimal specific training, GPT-3 often encounters limitations when applied to more niche fields due to its broad training scope.

On the other hand, Google’s BERT (Bidirectional Encoder Representations from Transformers) prioritizes the bidirectional training approach. This method significantly enriches the model's contextual comprehension, proving particularly effective in specialized applications like question answering and recognizing named entities. Nevertheless, adapting BERT to meet the unique demands of certain fields continues to pose substantial challenges.

1.3 Aim & Objective

1.3.1 Aim

The central objective of this dissertation is to develop and assess a question-answering (Q&A) model tailored to specific domains, leveraging the capabilities of large language models (LLMs). This study emphasizes the critical importance of privacy and data security in the deployment of LLMs. It aims to improve these models' efficacy and practical utility in specialized areas, ensuring they adhere to ethical guidelines and protect confidential data.

1.3.2 Objective

The objectives of my project are as follows:

i. **Literature Review and Background Study**

(a) Conduct a comprehensive review of existing literature on LLM models and their applications.

(b) Analyze the evolution of language models and the technological advancements

ii. **Understanding on Technical Foundations.**

(a) Explore the architectural principles of deploying LLMs, including the transformer architecture and training processes.

(b) Examine the strengths and limitations of current models, focusing on aspects like performance, scalability, and ethical considerations specifically related to organization.

iii. **Requirement Analysis.**

(a) Identify the specific requirements for developing a private GPT tailored to a particular domain or application.

(b) Determine the necessary data sources, computational resources, and technical infrastructure needed for training and deployment.

(c) Study requirement of maintaining a private repository of data to run models on a private cloud ensuring CIA.

iv. **Designing and Implementation.**

(a) Develop a detailed plan for the design and implementation of the private GPT model.

(b) Outline the steps for data collection, pre-processing, and augmentation to ensure a high-quality training dataset.

(c) Implement the training pipeline, including model architecture customization, hyper parameter tuning, and optimization techniques.

v. **Evaluation and Testing.**

(a) Establish evaluation metrics and benchmarks to assess the performance of the private GPT model.

(b) Conduct rigorous testing to ensure the model meets the desired accuracy, efficiency, and reliability standards.

(c) Compare the performance of the private GPT with existing public models to highlight improvements or unique features.

vi. **Privacy and Security Considerations.**

(a) Address the privacy and security challenges associated with developing and deploying a private GPT.

(b) Establish protocols to safeguard confidential information and adhere to pertinent regulations and the organization's standard operating procedures.

vii. **Ethical Implications.**

(a) Explore the ethical implications of deploying a private GPT, including potential biases, misuse, and impact on users.

(b) Propose strategies to mitigate ethical concerns and promote responsible use of the technology.

viii. **Scalability and Maintenance.**

(a) Bring out strategies for scaling the private GPT model to handle large-scale data and user interactions for different us cases.

(b) Develop a maintenance plan to ensure the model remains up-to-date and continues to perform effectively over time.

ix. **Documentation and Dissemination.**

(a) Document the entire development process, including design decisions, challenges faced, and solutions implemented.

(b) Publish a paper if novel work is carried out and achieved during the implementation.

1.4 Scope of Dissertation

The scope of this dissertation encompasses the development, implementation, evaluation, and documentation of an offline AI generative question-answering system. This system will leverage advanced language models and retrieval mechanisms to provide accurate and relevant answers based on a corpus of PDF documents

i. **Development:**

(a) Extract and pre-process text from PDF documents.

(b) Train a language model on the processed text to understand and generate relevant answers.

(c) Implement an efficient information retrieval mechanism to locate pertinent information within the documents.

ii. **Implementation:**

(a) Develop a comprehensive plan for designing and deploying the offline question-answering system.

(b) Customize the model architecture, fine-tune hyper parameters, and optimize the training process.

(c) Ensure the system can function independently of internet connectivity, relying solely on the local document corpus.

iii. **Evaluation:**

(a) Establish evaluation metrics and benchmarks to assess the system's accuracy, efficiency, and overall performance.

(b) Conduct rigorous testing to ensure the system meets desired standards and compare its performance with existing online systems.

(c) Evaluate the system's ability to handle various types of queries and its effectiveness in different scenarios.

iv. **Documentation:**

(a) Document the entire development process, including design decisions, challenges encountered, and solutions implemented.

(b) Provide detailed documentation of the system's architecture, data processing methods, and training procedures.

(c) Publish findings and insights gained from the project, potentially contributing novel work to the field of offline AI question-answering systems.

By addressing these aspects, the dissertation aims to create a robust and reliable offline question-answering system that can operate effectively in environments with limited or no internet access. The system will be a valuable tool for applications in remote locations, secure facilities, and situations where internet connectivity is not feasible.

1.4.1 In scope

The scope of dissertation includes the following key elements and activities:

i. **Corpus Preparation**

- Collection and pre-processing of plain text /PDF documents relevant to the chosen domain.
- Extraction of text from PDF documents using appropriate tools and techniques.
- Cleaning and formatting the extracted text for consistency and usability in model training.

ii. **Model Development**

- Selection and customization of a pre-trained large language model (LLM) suitable for the Q&A task.
- Fine-tuning the selected LLM on the preprocessed text corpus to enhance domain-specific understanding and performance.
- Implementation of an information retrieval mechanism to support accurate and efficient answer generation.

iii. **System Design and Implementation**

- Designing the architecture of the offline Q&A system to ensure robustness and scalability.
- Implementing the Q&A system, ensuring it functions independently of internet connectivity and relies solely on the local document corpus.
- Fine-tuning hyper parameters and optimizing the training process for improved performance.

iv. **Privacy and Security Measures**

- Integrating advanced privacy-preserving techniques, such as federated learning and differential privacy, to protect sensitive information during model training and deployment.
- Ensuring compliance with relevant regulatory frameworks and establishing best practices for data privacy and security.

v. **Evaluation and Testing**

- Establishing evaluation metrics and benchmarks to rigorously assess the system's accuracy, efficiency, and overall performance.
- Conducting comprehensive testing to validate the system's effectiveness in handling various types of queries and operational scenarios.
- Comparing the performance of the offline Q&A system with existing online systems to highlight improvements and areas for further development.

vi. **Documentation and Dissemination**

- Documenting the development process, including detailed descriptions of design decisions, challenges encountered, and solutions implemented.
- Providing thorough documentation of the system's architecture, data processing methods, and training procedures.
- Publishing the findings and insights from the project to contribute to the field of offline AI question-answering systems and support future research.

1.4.2 Out Scope

While this dissertation aims to develop a robust offline AI generative question-answering (Q&A) system, certain aspects and activities fall outside its scope. These include:

i. **Online and Real-Time Data Processing**

- The system is designed without incorporating real-time data processing or the management of live data streams.
- Integration with live data sources or continuous updates from online repositories is not considered within this project's scope.

ii. **Cross-Domain Model Generalization**

- The focus is on creating a domain-specific Q&A system, and cross-domain model generalization or adaptation to multiple unrelated domains is not covered.
- The research will not explore general-purpose Q&A models applicable to a wide range of topics.

iii. **Natural Language Understanding Beyond the Q&A Task**

- While the system aims to perform well in question-answering tasks, it will not extend to other natural language understanding tasks such as sentiment analysis, machine translation, or text summarization.
- The project will not delve into multi-task learning or the development of models for diverse NLP tasks beyond Q&A.

iv. **Advanced Model Interpretability and Explain ability**

- The dissertation will not focus on developing advanced interpretability or explain ability features for the model.
- Detailed exploration of model transparency, interpretability techniques, or user-friendly explanation interfaces is beyond the scope.

v. **User Interface and Experience Design**

- The primary focus is on the backend development of the Q&A system; comprehensive user interface (UI) and user experience (UX) design is not included.
- While basic functionalities for interacting with the system will be provided, advanced UI/UX considerations and end-user application development are out of scope.

vi. **Large-Scale Deployment and Maintenance**

- The research will develop a prototype offline Q&A system, but large-scale deployment strategies, ongoing maintenance, and operational support are not part of the dissertation.
- The dissertation does not cover aspects such as scaling the system for enterprise use, continuous model updates, or long-term system maintenance.

vii. **Extensive Comparative Analysis with Multiple Models**

- While the dissertation will benchmark the developed system against a few existing models, it will not conduct an extensive comparative analysis with a broad range of other models or systems.
- Conducting extensive performance evaluations across various cutting-edge models and systems exceeds the scope of this project.

viii. **Regulatory and Legal Compliance Across Multiple Jurisdictions**

- While the dissertation will consider privacy and security measures, detailed exploration of regulatory and legal compliance across various jurisdictions is not included.
- The focus will be on general best practices for data privacy rather than in-depth legal analysis or compliance strategies for specific regions.

1.4.3 Reason for defining the Scope

i. **Clarity and Focus**

- **Establish Boundaries**: Clearly defining what is in and out of scope helps establish the boundaries of my research. This prevents scope creep and ensures that the project stays focused on its primary objectives.
- **Guides Research Direction**: It directs the research efforts and resources towards specific goals, ensuring that the work is purposeful and aligned with the intended outcomes.

ii. **Feasibility and Manageability**

- **Set Realistic Expectations**: By outlining what will and won't be covered, you set realistic expectations for the scope and scale of the project, making it more manageable within the available time and resources.
- **Avoid Overextension**: Helps in avoiding overextension by identifying the limits of what can be accomplished, thus maintaining the project's feasibility and sustainability.

iii. **Resource Allocation**

- **Efficient Use of Resources**: Defining the scope allows for the efficient allocation of resources, including time, budget, and effort, ensuring that they are utilized effectively to achieve the research goals.

- **Focused Research Activities**: It enables prioritization of key activities and tasks, ensuring that critical aspects of the research are addressed without unnecessary duplication or diversion of resources.

iv. **Stakeholder Communication**

- **Clear Communication**: Provides a clear basis for communicating with stakeholders, including advisors, funding bodies, and potential collaborators, about what the research will cover and its intended contributions.
- **Expectation Management**: Helps in managing expectations by providing a detailed outline of the research's scope, objectives, and deliverables, reducing the likelihood of misunderstandings or misaligned expectations.

v. **Quality and Depth of Research**

- **Enhances Depth and Quality**: By narrowing the focus, the research can delve deeper into specific areas, leading to a more thorough and detailed exploration of the chosen topics.
- **Facilitates Rigorous Analysis**: Allows for the development of more targeted research methodologies, ensuring that the analysis is rigorous, relevant, and comprehensive within the defined scope.

vi. **Documentation and Reporting**

- **Structured Documentation**: Provides a framework for documenting the research process, findings, and conclusions, ensuring that the dissertation is well-organized and coherent.
- **Basis for Future Work**: Establishes a clear foundation for future research, providing a documented rationale for the scope and boundaries, which can be useful for subsequent studies or developments in the field.

vii. **Compliance and Ethical Considerations**

- **Guides Ethical Standards**: Ensures that the research complies with ethical standards and guidelines, particularly regarding privacy, data security, and participant consent.
- **Regulatory Adherence**: Helps in adhering to regulatory and legal requirements relevant to the research, ensuring that the study is conducted within the appropriate legal and ethical frameworks.

viii. Significance Study

This study holds significant potential to contribute to the advancement of artificial intelligence (AI) and natural language processing (NLP) by tackling critical challenges associated with the creation of domain-specific question-answering (Q&A) systems. The main points underscoring the importance of this research are as follows:

a) **Advancement in Domain-Specific AI Solutions**

This research contributes to the development of tailored AI solutions that enhance the performance and relevance of large language models (LLMs) in specialized fields. By focusing on domain-specific applications, the study aims to bridge the gap between general-purpose models and the nuanced requirements of specific domains such as medicine, law, and finance.

b) **Enhancing Data Privacy and Security**

In an era of increasing concerns over data privacy and security, this study emphasizes the integration of robust privacy-preserving techniques. The implementation of federated learning, differential privacy, and secure multi-party computation ensures that sensitive information is protected during model training and deployment, aligning with regulatory standards and ethical guidelines.

c) **Development of a Practical Offline Q&A System**

By developing an offline Q&A system, the study addresses the need for reliable AI tools in environments with limited or no internet connectivity. This has significant implications for applications in remote locations, secure facilities, and situations where internet access is unreliable or non-existent.

d) **Contribution to the Academic and Practical Understanding**

The dissertation provides a detailed exploration of the technical, ethical, and practical aspects of developing a domain-specific Q&A model. It aims to fill existing research gaps by offering insights into model customization, training optimization, and privacy-enhancing techniques, thereby contributing valuable knowledge to the fields of AI and NLP.

e) **Foundation for Future Research and Development**

The findings and methodologies developed in this study serve as a foundation for future research in offline AI systems and domain-specific NLP applications. By documenting the development process, challenges, and solutions, the study paves the way for further innovations and enhancements in this area.

1.5 Structure of Study

The dissertation is structured to provide a comprehensive overview of the research process, findings, and contributions. The structure is as follows:

1.5.1 Chapter 1: Introduction

- **Overview of the Study**: Introduction to the dissertation topic, background, problem statement, aim, objectives, scope, and significance.

- **Outline of the Dissertation**: Overview of the chapters and their contents to guide the reader through the study.

1.5.2 Chapter 2: Literature Review

- **Historical Context and Evolution**: Review of the development and advancements in large language models (LLMs) and their applications in NLP.
- **Current Trends and Gaps**: Analysis of recent research, identifying trends, challenges, and gaps in the development of domain-specific Q&A systems.
- **Privacy and Security Considerations**: Examination of existing privacy-preserving techniques and ethical considerations in AI model deployment.

1.5.3 Chapter 3: Methodology

- **Research Design**: Detailed description of the research design, including the approach, methodology, and tools used for developing the Q&A system.
- **Data Collection and Pre-processing**: Methods for collecting and preparing the corpus of PDF documents for model training.
- **Model Development and Training**: Overview of the model architecture, training process, hyper parameter tuning, and optimization techniques.

1.5.4 Chapter 4: System Design and Implementation

- **System Architecture**: Detailed design of the offline Q&A system, including its components and interaction flow.
- **Implementation Details**: Description of the implementation process, focusing on text extraction, information retrieval, and model integration.
- **Privacy and Security Implementation**: Strategies and technologies employed to ensure data privacy and security.

1.5.5 Chapter 5: Evaluation and Results

- **Evaluation Metrics and Benchmarks**: Definition of metrics and benchmarks used to assess the system's performance.
- **Testing Methodology**: Description of the testing procedures, including scenario-based testing and comparison with existing systems.
- **Results Analysis**: Presentation and analysis of the evaluation results, highlighting the system's strengths and areas for improvement.

1.5.6 Chapter 6: RAG with Fine Tune LLM and GUI

- Introduction , Architecture and Implementation
- Development of GUI

1.5.7 Chapter 7: Documentation and Dissemination

- **Document the development process**: Analysis of the results in relation to research objectives and existing literature.
- **Prepare the finalize dissertation**: Compare the system's performance to other model and highlights the contribution.
- **Challenges and Solutions**: Reflection on the challenges faced during the research and describe the solution used to overcome them.

1.5.8 References

Citations: Comprehensive list of all scholarly references and sources cited throughout the dissertation.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The emergence of large language models (LLMs) has greatly impacted the fields of natural language processing (NLP) and artificial intelligence (AI). Among the latest advancements, LLAMA 3 stands out as a cutting-edge model specifically designed to handle domain-specific tasks while addressing privacy concerns. This section examines the existing literature on large language models, with an emphasis on LLAMA 3, its predecessors, and related privacy-preserving methods. The review underscores significant progress in the field, identifies existing gaps, and lays the groundwork for the development of a domain-specific question-answering (Q&A) system utilizing LLAMA 3.

2.2 Literature Review of Proposed Models and Algorithms

2.2.1 Evolution of Large Language Models

The progression of LLMs has been marked by several notable models, each contributing to the field's evolution:

- **Word2Vec and GloVe**: Early models such as Word2Vec [2]and GloVe [3] introduced the concept of word embedding, capturing semantic relationships through dense vector representations.
- **Sequence-to-Sequence Models**: The development of sequence-to-sequence models [4] utilizing RNNs and LSTMs enabled significant improvements in machine translation and text generation.
- **Transformers and Attention Mechanism**: The Transformer architecture [1] revolutionized NLP by allowing for parallelization and efficiently handling long-range dependencies through the attention mechanism.

2.2.2 Breakthroughs with GPT, BERT, and LLAMA

- **BERT**[5]: Bidirectional Encoder Representations from Transformers (BERT) set new benchmarks by pre-training on large text corpora and fine-tuning on specific tasks, significantly improving various NLP applications.

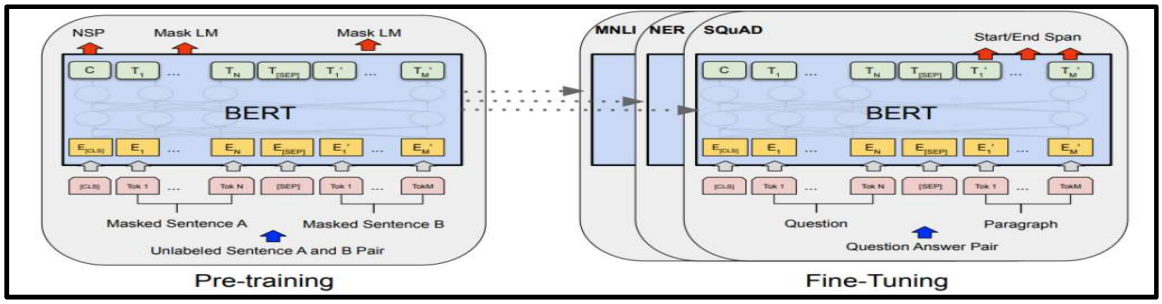


Figure 2.2-1 BERT architecture and pipeline for pretraining and fine-tuning.

- **GPT Series** [6]: The Generative Pre-trained Transformer (GPT) series, especially GPT-3, demonstrated the power of large-scale unsupervised learning, showcasing remarkable capabilities in text generation and question-answering.
- **LLAMA Series (LLAMA 1 and 2)**: The LLAMA series introduced enhancements in model architecture and training efficiency, focusing on practical applications in various domains. LLAMA 2 [7]emphasized improvements in performance and adaptability to specific tasks.
- **LLAMA 3**: LLAMA 3 represents a significant leap in LLM capabilities, with an emphasis on domain-specific customization, efficiency, and privacy. Its architecture incorporates advanced techniques to enhance model performance while addressing the challenges of data privacy and security.

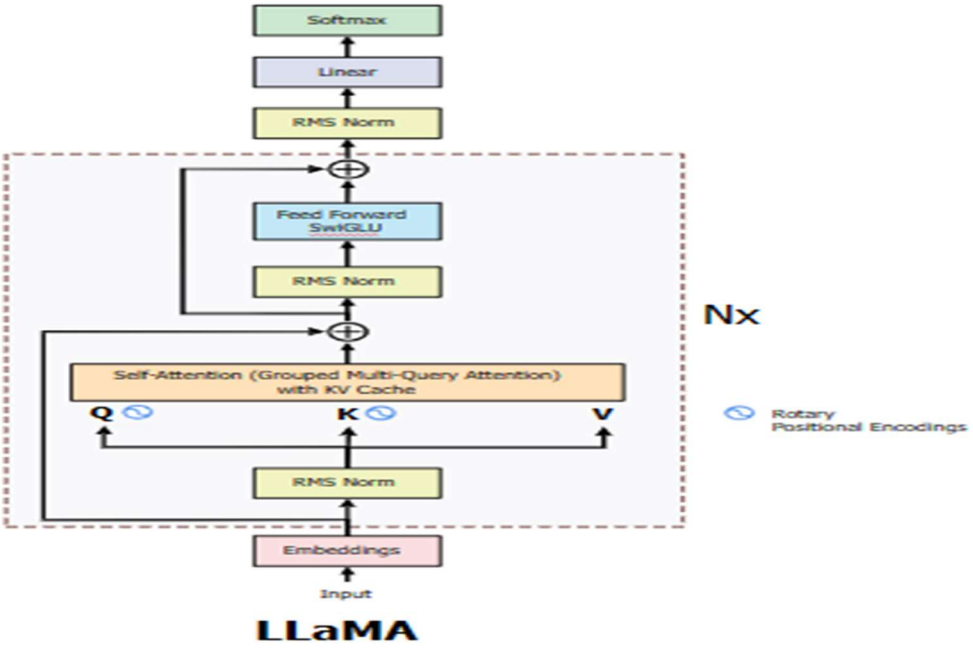


Figure 2.2-2 LLAMA Architecture

Note: The architecture focused on the Inferencing the model not for training and we can say that Decoder Block with Cross Attention is not being covered and KV Cache will not be used for Training Phase of the model.

2.2.3 Advances in Domain-Specific Models

- **Domain Adaptation and Fine-Tuning**: Research on domain adaptation [8] has demonstrated the effectiveness of fine-tuning pre-trained models on domain-specific data. Models like SciBERT [9]and BioBERT [10]have shown significant improvements in scientific and biomedical NLP tasks.
- **Specialized Architectures**: Developing specialized architectures for domain-specific applications has been a key focus. Studies on legal Q&A systems [11]and medical Q&A systems [12]highlight the importance of tailoring models to meet the unique requirements of specific fields.

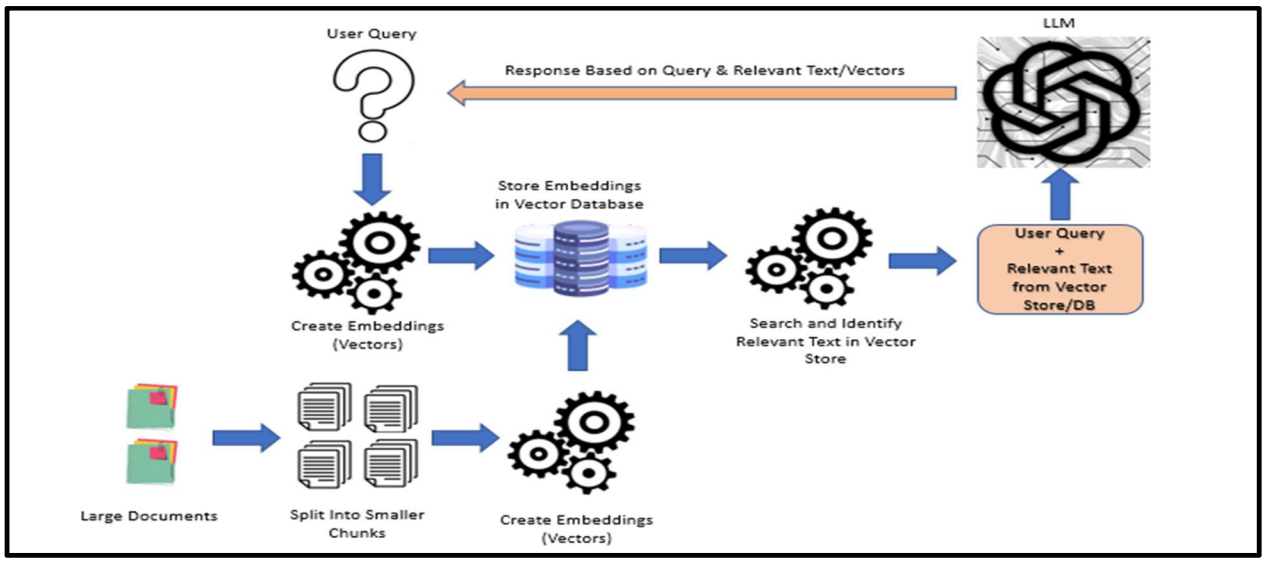


Figure 2.2-3 Domain Specific Q&A Application

2.2.4 Privacy-Preserving Techniques in NLP

- **Differential Privacy**: Differential privacy [13]has become a cornerstone for ensuring privacy in data analysis. Techniques for integrating differential privacy into neural network training [14]have been explored to protect sensitive information.
- **Federated Learning**: Federated learning [15] enables decentralized model training, enhancing privacy by keeping data local. This approach has been investigated for various NLP tasks, facilitating collaborative learning without data centralization.
- **Homomorphic Encryption and Secure Multi-party Computation**: Advanced cryptographic methods, including homomorphic encryption and secure multi-party computation, have been utilized to enable secure model training. These techniques play a crucial role in preserving data privacy by allowing computations to be performed on encrypted data without exposing sensitive information during the training process.

2.2.5 State-of-the-Art Question-Answering Systems

- **SQuAD and Its Variants**: The Stanford Question Answering Dataset (SQuAD) [16] and its subsequent versions serve as benchmarks for evaluating Q&A systems. Models like BERT and RoBERTa have achieved state-of-the-art performance on these datasets.
- **Domain-Specific Q&A Models**: Recent advancements in domain-specific Q&A systems [17]emphasize the importance of customizing models for particular fields. These models leverage specialized data and techniques to improve accuracy and relevance.

2.3 Summary

This literature review highlights the rapid advancements in LLMs and the significant strides made with models like LLAMA 3/Gemma Model. Despite the progress, challenges remain in adapting these models for domain-specific tasks, particularly in offline settings with stringent privacy requirements. The integration of privacy-preserving techniques and the development of robust Q&A systems tailored to specific domains are crucial areas for future research. This study aims to address these

gaps by leveraging LLAMA 3 to develop a state-of-the-art offline Q&A system with a strong emphasis on privacy and data security.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

This research methodology section describes the structured approach taken to develop an offline question-answering (Q&A) system using LLAMA 3. It covers the methods employed to understand the organizational problem, the processes involved in selecting and preparing data, the steps for transforming data for model training, and the identification of the necessary hardware and software resources for effective implementation. The goal is to present a detailed framework for the design, implementation, and evaluation of the Q&A system, ensuring its efficiency and effectiveness within a domain-specific context.

3.2 Research Methodology

3.2.1 Organization Understanding

- **Problem Definition**: The primary objective is to develop an offline Q&A system tailored to a specific domain. This involves identifying the key functional requirements, including the types of queries the system should handle and the nature of the responses expected. Key considerations include understanding user needs, domain-specific terminology, and the privacy concerns associated with handling sensitive data. The data source is being created as a primary source to prepare the offline Question Answer for organization.
- **Scope and Constraints**: The system will be designed to operate offline, meaning it must be fully functional without internet connectivity. Constraints include ensuring that the system remains efficient and responsive despite operating on potentially limited computational resources. Privacy constraints necessitate careful handling of data to prevent unauthorized access or leakage. RAG pipeline is being introduce with fine-tuned large language model (LLM) to enhance the optimality of hardware and the consideration of response time on user queries.

3.3 Prepare Input Dataset

Preparing a dataset to train Language Model (LLM) involves several key steps.

3.3.1 Define the Task and Goals:

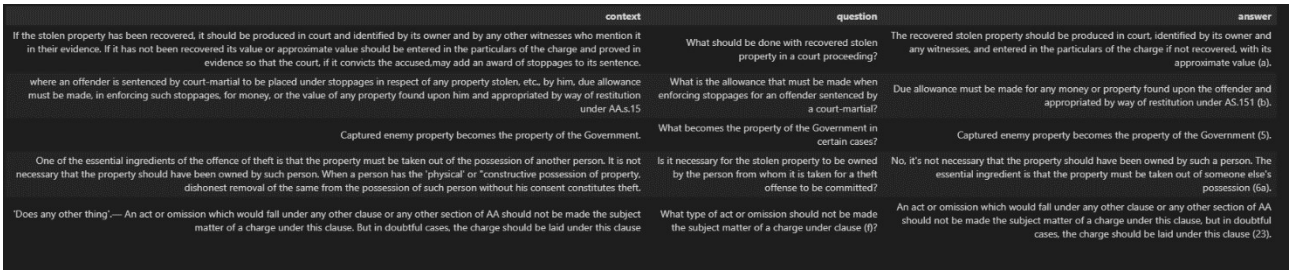
- Determine the specific task(s) that we want our LLM to perform (e.g., text generation, sentiment analysis, question answering on specific domain using fine tune and utilizing of RAG pipeline).
- Define the goals of training dataset, such as the diversity of language, domain relevance, and data quality.

3.3.2 Data Collection:

- Gather a diverse and representative collection of text data relevant to task and goals.
- Sources can include websites, books, articles, social media, forums, and domain-specific documents. This table is just showing the particular example of dataset but for specific use case it can be differ.

S No	Question	Answer	Context	Team Member	Milestone
1.	Project x	Development of a new AI model for customer support.	Ongoing	[‘A’,’B’,’C’]	[‘Research’,’Design’, ’Development’]
2.	Project y	Redesign of the company website for better user experience.	Completed	[‘X’,’Y’,’Z’]	[‘Research’,Design’, ’Launch’]

Table 3.3.2-1 Data Format

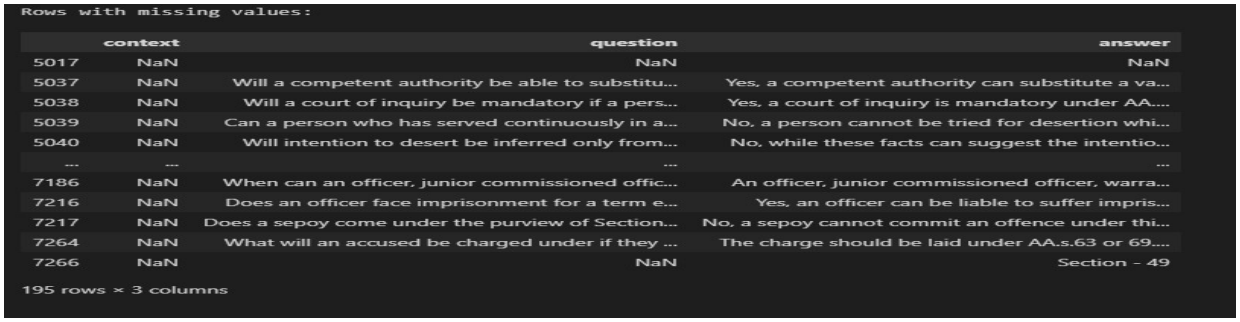


context	question	answer
If the stolen property has been recovered, it should be produced in court and identified by its owner and by any other witnesses who mention it in their evidence. If it has not been recovered its value or approximate value should be entered in the particulars of the charge and proved in evidence so that the court, if it convicts the accused,may add an award of stoppages to its sentence.	What should be done with recovered stolen property in a court proceeding?	The recovered stolen property should be produced in court, identified by its owner and any witnesses, and entered in the particulars of the charge if not recovered, with its approximate value (a).
where an offender is sentenced by court-martial to be placed under stoppages in respect of any property stolen, etc., by him, due allowance must be made, in enforcing such stoppages, for money, or the value of any property found upon him and appropriated by way of restitution under AA.s.15	What is the allowance that must be made when enforcing stoppages for an offender sentenced by a court-martial?	Due allowance must be made for any money or property found upon the offender and appropriated by way of restitution under AS.151 (b).
Captured enemy property becomes the property of the Government.	What becomes the property of the Government in certain cases?	Captured enemy property becomes the property of the Government (5).
One of the essential ingredients of the offence of theft is that the property must be taken out of the possession of another person. It is not necessary that the property should have been owned by such person. When a person has the 'physical' or "constructive possession of property. dishonest removal of the same from the possession of such person without his consent constitutes theft.	Is it necessary for the stolen property to be owned by the person from whom it is taken for a theft offense to be committed?	No, it's not necessary that the property should have been owned by such a person. The essential ingredient is that the property must be taken out of someone else's possession (6a).
'Does any other thing'.— An act or omission which would fall under any other clause or any other section of AA should not be made the subject matter of a charge under this clause. But in doubtful cases, the charge should be laid under this clause	What type of act or omission should not be made the subject matter of a charge under clause (f)?	An act or omission which would fall under any other clause or any other section of AA should not be made the subject matter of a charge under this clause, but in doubtful cases, the charge should be laid under this clause (23).

Figure 3.3-1Screen shot related to data used

3.3.3 Data Cleaning and Pre-processing:

- Remove irrelevant or duplicate data.



Rows with missing values:

	context	question	answer
5017	NaN	NaN	NaN
5037	NaN	Will a competent authority be able to substitu...	Yes, a competent authority can substitute a va...
5038	NaN	Will a court of inquiry be mandatory if a pers...	Yes, a court of inquiry is mandatory under AA....
5039	NaN	Can a person who has served continuously in a...	No, a person cannot be tried for desertion whi...
5040	NaN	Will intention to desert be inferred only from...	No, while these facts can suggest the intentio...
...	...	...	...
7186	NaN	When can an officer, junior commissioned offic...	An officer, junior commissioned officer, warra...
7216	NaN	Does an officer face imprisonment for a term e...	Yes, an officer can be liable to suffer impris...
7217	NaN	Does a sepoy come under the purview of Section...	No, a sepoy cannot commit an offence under thi...
7264	NaN	What will an accused be charged under if they ...	The charge should be laid under AA.s.63 or 69....
7266	NaN	NaN	Section - 49

195 rows × 3 columns

Figure 3.3-2Identify Missing Value in data during data preprocessing

- Standardize the text by converting it to lowercase, removing punctuation, and handling special characters.
- Tokenize the text into words or sub words using a tokenizer appropriate for our language model (e.g., Word Piece, Byte Pair Encoding).

3.3.4 Data Formatting:

- Supervised Learning:
  i. If task requires supervised learning (where we provide both input and desired output), format data into pairs.
  ii. Each pair will have an "Input Prompt" and a corresponding "Desired Output".
  iii. The prompt should be clear and concise, setting the context for the desired output.
  iv. The output could be a completion of a sentence, a summary of a passage, or an answer to a question, depending on our goal.
- Unsupervised Learning:
  i. If we're going for unsupervised learning (where the model learns patterns from raw text), format data into single text chunks.
  ii. Ensure these chunks are of appropriate size and represent the variety of your textual data.

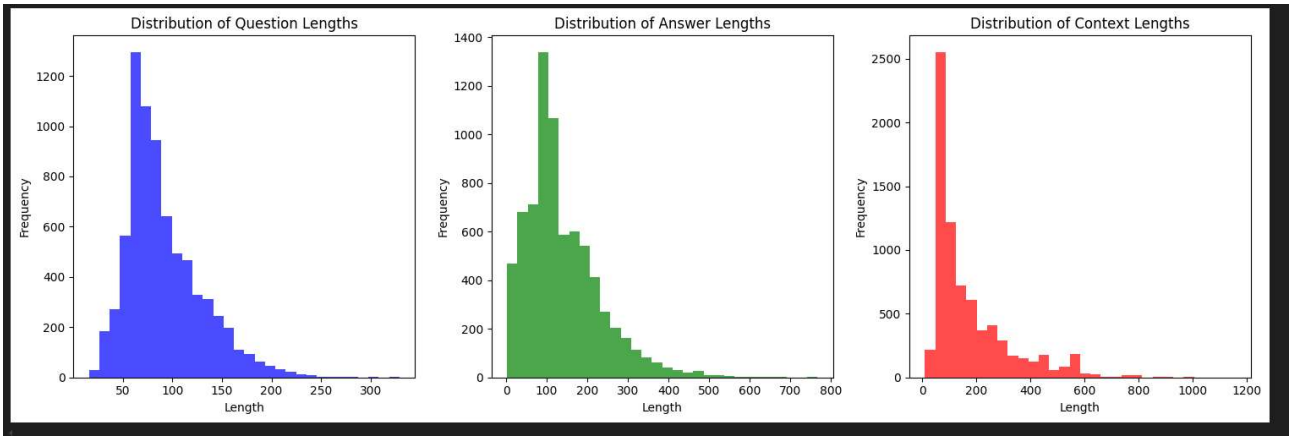


Figure 3.3-3 Data Distribution

3.3.5 Data Augmentation (Optional):

- If dataset is small or lacks diversity, consider data augmentation techniques such as paraphrasing, text swapping, or adding noise to increase the dataset size and variety.

3.3.6 Splitting the Dataset:

- In this we divide our dataset into Two Split or Three Split (Train, Test & Validation).
- During the model development process, the training data is utilized to train the model and facilitate learning. The validation data is employed for tuning hyperparameters and assessing the

model's performance throughout the training phase. Finally, the test set is reserved for evaluating the performance of the finalized model to ensure its effectiveness on unseen data.

3.3.7 Data Encoding:

- Encode the text data into numerical format suitable for input to LLM.
- Use techniques such as one-hot encoding, token embedding, or word embedding (e.g., Word2Vec, GloVe, BERT embeddings).

3.3.8 Data Loading and Batching:

- Implement data loading and batching mechanisms to efficiently feed the data into training pipeline.
- Use libraries like TensorFlow's tf.data.Dataset or PyTorch's Data Loader for handling large datasets and batching.

3.3.9 Data Pre-processing Pipeline (Optional):

- Create a data pre-processing pipeline that encapsulates the cleaning, tokenization, encoding, and batching steps.
- This pipeline ensures consistency and reproducibility in data preparation.

3.3.10 Data Quality Assurance:

- Perform quality checks on your dataset to ensure it meets the desired standards.
- Check for data imbalance, data biases, and data integrity issues.

3.3.11 Data Versioning and Management:

- Implement a system for versioning and managing dataset to track changes, updates, and annotations.
- Use tools like Git LFS (Large File Storage) or cloud-based data versioning platforms.

3.3.12 Documenting the Dataset:

- Create documentation for dataset, including information about its source, pre-processing steps, data format, and any annotations or labels.
- Document any licensing or usage restrictions associated with the data Pre-processing

3.4 Prepare Input Dataset (Implementation)

3.4.1 Create Custom Dataset

Model: LLM (LLaMA3 8B/Gemma 8B)

Developing a custom **dataset for training LLM** is an important part of building a strong AI system. It's about gathering and organizing the right data that the model needs to learn and improve its understanding of language. This process helps the model become more accurate and effective in its tasks.

Preparing a dataset to train LLaMA3 involves several key steps.

Define the Task and Goals:

Understanding the use case for generating a custom dataset is key to creating a dataset that meets the specific needs of the project. It involves identifying the purpose and goals of the dataset, as well as the target audience or application for which the model will be used. By clarifying the use case upfront, developers can ensure that the custom dataset is relevant, comprehensive, and suitable for training the model effectively.

> Example use-case: **Develop a conversational dataset for each ongoing company project and train a language model to answer questions related to any project accurately.**
>
> **Constraint:** *The data is very confidential and cannot be sent to GPT or any external OpenAI API for processing.*

Data Collection:

Collecting the data to fulfill the above use case plays a vital role in efficient training of LLaMA3. The data should be accurate.

Let's say, I have a database which has all the necessary details about all the ongoing projects in the company.

Step 1: Create a Prompt that converts the dataset into a conversational dataset such that all the information is transformed in the required format.

Example Prompt:

> "*For each project in the database, create a conversation that includes the project name, description, status, team members, and key milestones. Format it as a Q&A conversation.*"

Step 2: Utilize an internal LLM or a locally hosted LLM for inferencing to generate the conversational dataset to train LLaMA3.

```
from transformers import pipeline  # Load a locally hosted model (e.g., LLaMA3 or another compatible model)
model_name = "your-local-llm-model"
generator = pipeline('text-generation', model=model_name)
def generate_conversation(project_details):
prompt = f"Create a conversation for the following project details:\n\n{project_details}\n\nConversation:"
    response = generator(prompt, max_length=150, num_return_sequences=1)
    return response[0]['generated_text'].strip()
project_details = """
Project Name: Project A
Description: Development of a new AI model for customer support.
Status: Ongoing
Team Members: Alice, Bob, Charlie
Key Milestones: Initial Planning, Prototype Development, Testing, Deployment"""
conversation = generate_conversation(project_details)
print(conversation)
```

Step 3: Generate multiple conversations related to every project, so that a large amount of dataset is created to train the LLM.

```
projects = [
    {
        "name": "Project X",
        "description": "Development of a new AI model for customer support.",
        "status": "Ongoing",
        "team_members": ["A", "B", "C"],
        "milestones": ["Initial Planning", "Prototype Development", "Testing", "Deployment"]
    },
    {
        "name": "Project Y",
        "description": "Redesign of the company website for better user experience.",
        "status": "Completed",
        "team_members": ["D", "E", "F"],
        "milestones": ["Research", "Design", "Development", "Launch"]
    }
]

conversations = []

for project in projects:
    details = f"""Project Name: {project['name']}\n
                  Description: {project['description']}\n
                  Status: {project['status']}\n
                  Team Members: {', '.join(project['team_members'])}\n
                  Key Milestones: {', '.join(project['milestones'])}"""
    conversation = generate_conversation(details)
    conversations.append(conversation)

print(conversations)
```

Data Cleaning and pre-processing:

Data cleaning acts as a filter layer after data generation. We have generated the data, but the data is not in the format that is understandable by LLaMA3. Every LLM has its own way to utilize data while training.

Data Formatting:

There are two types of Data Formatting available:

- Supervised Learning:
  Formatting the data into pairs. Each pair will have an "Input Prompt" and a corresponding "Desired Output".

```
# Data Formatting
formatted_data = []

for conv in cleaned_conversations:
    input_prompt = "Project conversation"
    desired_output = conv
    formatted_data.append({"input": input_prompt, "output": desired_output})

print(formatted_data)
```

Figure 3.3.4-1Formatting of Dataset

- Unsupervised Learning: **(Not applicable for our use case)**
- Data Augmentation (Optional): **(Not applicable for our use case)**

Used if the dataset is small or lacks diversity, consider data augmentation techniques such as paraphrasing, text swapping, or adding noise to increase the dataset size and variety.

Splitting the Dataset:

Dataset is divided into **training and validation sets.**

- Training data - 80%
- Validation data - 20%

Example Code:

```
# Splitting Data

from sklearn.model_selection import train_test_split

train_data,val_data=train_test_split(formatted_data,test_size=0.2,random_state=42)

print(f"Training Set: {len(train_data)} samples")

print(f"Validation Set: {len(val_data)} samples")
```

Figure 3.3.4-2 Train and Validation Dataset

Data Parsing:

Data parsing involves converting the cleaned and formatted data into a structure that LLaMA3 can easily consume for training. This step is crucial for ensuring that the data is correctly understood by the model.

```
import json

def parse_data(data):
    parsed_data = []
    for item in data:
        input_text = item['input']
        output_text = item['output']
        parsed_data.append({"prompt": input_text, "completion": output_text})
    return parsed_data

parsed_train_data = parse_data(train_data)
parsed_val_data = parse_data(val_data)

# Save parsed data to JSON files
with open('parsed_train_data.json', 'w') as f:
    json.dump(parsed_train_data, f, indent=4)

with open('parsed_val_data.json', 'w') as f:
    json.dump(parsed_val_data, f, indent=4)

print("Data parsing complete. Parsed data saved to JSON files.")
```

Figure 3.3.4-3 Parsed and Split Dataset

The **custom dataset is now ready** and **properly structured for fine-tuning the LLaMA3** model. With this dataset, you can effectively train LLaMA3 to perform well on your specific tasks and achieve the desired outcomes for your project.

3.5 Fine Tune LLaMA3/Gemma model on Custom Dataset (setup)

LoRA(Part 1)

3.5.1 Introduction

Fine-tuning a pre-trained model like LLaMA3/Gemma on a custom dataset involves **adjusting the model parameters** to better fit the specific data and tasks at hand. This process enhances the model's ability to perform specific tasks and improves its performance on the given dataset. In this guide, we will use the **Low Rank Adaptation (LoRA) technique for fine-tuning**, which is efficient in terms of computational resources and memory.

3.5.2 Partial vs Full Fine-Tuning

3.5.2.1 Partial Fine-Tuning:

Partial fine-tuning involves updating only a subset of the model's parameters. This approach is useful when you want to adapt the model with fewer computational resources or when you aim to prevent overfitting on a small dataset. It often involves fine-tuning the last few layers of the model or using techniques like LoRA to inject learnable parameters into specific parts of the model.

Advantages:

- Reduced computational and memory requirements.
- Faster training times.
- Lower risk of overfitting, especially with smaller datasets.

Implementation:

- With LoRA, only specific layers or modules are adapted.
- Example: Injecting low-rank matrices into the attention mechanisms.

3.5.2.2 Full Fine-Tuning:

Full fine-tuning involves updating all the parameters of the pre-trained model. This approach can leverage the full capacity of the model to adapt to the new dataset but requires more computational resources and carries a higher risk of overfitting if the dataset is not sufficiently large or diverse.

Advantages:

- Potentially better performance if the dataset is large and diverse.
- Full utilization of the model's capacity.

Disadvantages:

- Higher computational and memory requirements.
- Longer training times.
- Increased risk of overfitting with smaller datasets.

Implementation:

- Update all model parameters during training.

3.5.3 Define the Task and Goals

The goal of fine-tuning process is to ensure the LLaMA3/Geema model to accurately respond to questions and **perform tasks specific to the custom dataset prepared from the organization's project details**. The fine-tuned model should be able to provide precise and relevant answers based on the project information.

3.5.4 Data Preparation:

Ensure that the custom dataset is properly parsed and split into training and validation sets as described in the previous document.

Example Code to load parsed data:

```
import json

# Load the parsed training and validation data
with open('parsed_train_data.json', 'r') as f:
    train_data = json.load(f)

with open('parsed_val_data.json', 'r') as f:
    val_data = json.load(f)
```

Figure 3.3.5-1 Saved in JSON Format

3.5.5 Setup for Fine-Tuning:

3.5.6 Environment Setup:
Ensure we have the necessary libraries installed. These libraries help in model loading, tokenization, and fine-tuning.

```
pip install transformers datasets accelerate
```

- Transformers: A library by Hugging Face that provides tools to work with transformer models.
- Datasets: Another library by Hugging Face to easily load and pre-process datasets.
- Accelerate: Helps in speeding up training and evaluation processes.
    - Before starting your training script, you need to configure the accelerate library. This step can be done using a command-line interface provided by accelerate:

```
accelerate config
```

- This command will guide you through setting up the configuration file for distributed training and mixed precision.

3.5.7 Download and Setup LLaMA3 8B/Gemma Instruct Model:

Download the pre-trained LLaMA3 8B/Gemma 8B instruct model. This involves downloading the model weights and configuration files.

3.5.8 Steps to download and setup:

- Follow the instructions from the model provider (such as Hugging Face or Meta AI) to download the model files.
  Links: Hugging Face Git Hub
- Ensure the files are saved in a directory you can access.

3.5.9 Python Code to load the model and tokenizer:

```
from transformers import LLaMATokenizer, LLaMAForCausalLM

# Replace with the correct path to the downloaded model files
model_path = "path/to/llama3-8b-instruct"

# Load the tokenizer and model
tokenizer = LLaMATokenizer.from_pretrained(model_path)
model = LLaMAForCausalLM.from_pretrained(model_path)
```

- LLaMATokenizer: Converts text to tokens that the model can understand.
- LLaMAForCausalLM: Loads the pre-trained LLaMA3 model for causal language modeling.

3.5.10 Low Rank Adaptation (LoRA) Configuration:

3.5.10.1 What is Low Rank Adaptation (LoRA)?

LoRA is a technique that adapts a pre-trained model by injecting low-rank matrices into its layers. The key idea is to reduce the number of parameters that need to be updated during fine-tuning, which makes the process more memory and computationally efficient. This is especially useful when working with large models like LLaMA3.

3.5.10.2 Why use LoRA?

- **Efficiency:** LoRA reduces the number of parameters that need to be fine-tuned, making the process faster and less memory-intensive.
- **Scalability:** Allows fine-tuning of very large models on limited hardware.

- **Flexibility:** Can be applied to various parts of the model architecture.

3.5.10.3 LoRA Parameters:

- **rank:** This determines the size of the low-rank matrices that are injected into the model's layers. A higher rank increases the capacity of the adaptation but also increases the computational cost. Common values are 4, 8, or 16.
- **alpha:** A scaling factor for the adaptation matrices. It controls the extent to which the low-rank adaptation influences the model.
- **target_modules:** Specifies which layers or modules of the model will be adapted. This depends on the model architecture and the specific layers you want to fine-tune.

LoRA technique can be implemented with fewer resources and are memory efficient.

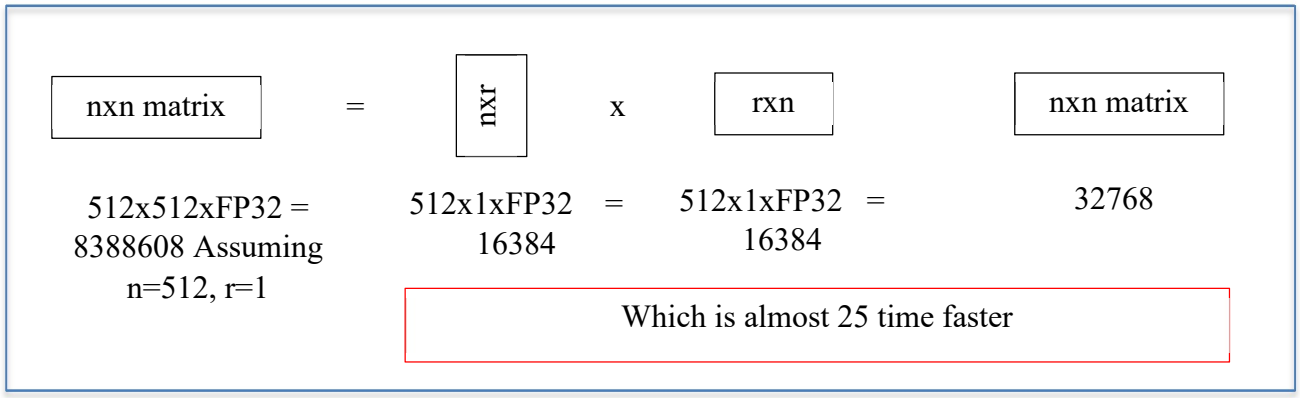


Figure 3.5-2Matrix handling for computer recourse

Example Configuration:

```
# LoRA configuration
lora_config = LoraConfig(
    lora_alpha=16,
    lora_dropout=0.1,
    r=6,
    target_modules=["q_proj", "o_proj", "k_proj", "v_proj", "gate_proj", "up_proj", "down_proj"],
    task_type="CAUSAL_LM"
)
```

Figure 3.3.5-3 LoRA Fine Tuning Scale

LoRA configuration with a rank of 16 and alpha of 16. If we want the model to learn new knowledge, we can increase the alpha to may 2x the rank, but we might need a lot of data to prevent the model from overfitting. If we want slightly change the output style of the model we can set alpha to about half of the rank.

3.6 Hyper parameters

Fine Tune LLaMA3 model on Custom Dataset (implementation)

3.6.1 Hyperparameter:

Hyperparameter are settings that need to be specified before training. They significantly impact the training process and the performance of the model.

3.6.1.1 Detailed Explanation of Key Hyperparameter:

- **learning_rate:** The learning rate determines how large the steps are during each iteration of the optimization process as the model works towards minimizing the loss function. A high learning rate can cause the model to converge too quickly, possibly settling on a non-ideal solution. On the other hand, a low learning rate can slow down the training, resulting in a prolonged and inefficient convergence process.
    - **Typical values:** 1e-4 to 1e-5.
    - **Example:** learning_rate=5e-5.

- **batch_size:** batch size is the number of samples that are processed prior the model is updated. A larger batch size can make training faster and more stable, but it requires more memory.
    - **Typical values:** 16, 32.
    - **Example:** per_device_train_batch_size=2, per_device_train_batch_size=16.

***One step will contain 32(1x2x16) row of training data, because of single GPU. Max_steps =100, so the training doesn't take too long time. If we calculate it, training will go through 3200 rows of training dat. If we don't set max_steps the training will go through 1 Epoch, which will whole dataset on time.**

Save Check Points 20 steps by setting and save_steps =5 over the entire training process 100/20 = 5. Then pass the Peft_config and training argument into SFT Trainer (Supervised fine Tuning trainer)

neftTune_noise_alpha = 5 - to increase fine tuning performance by adding nose to the embedding layer.

- **num_train_epochs:** Number of times the entire training dataset is passed through the model. More epochs can lead to better performance to model and it may also increase the risk of overfitting.
    - **Typical values:** 3-5.

- **Example:** num_train_epochs=3.

- **weight_decay:** it is a regularization technique used to reduce overfitting by penalizing large weights. It helps in generalizing the model better to unseen data.

  - **Typical values:** 0.01.
  - Example: weight_decay=0.01.

Example Hyperparameters: (*model specific)

```
from transformers import Trainer, TrainingArguments

training_args = TrainingArguments(
    output_dir="./results",  # Directory to save model checkpoints
    overwrite_output_dir=True,  # Overwrite existing files in the output directory
    num_train_epochs=3,  # Number of epochs for training
    per_device_train_batch_size=16,  # Batch size for training
    per_device_eval_batch_size=16,  # Batch size for evaluation
    learning_rate=5e-5,  # Learning rate for optimizer
    weight_decay=0.01,  # Weight decay for regularization
    logging_dir="./logs",  # Directory for logs
    logging_steps=10,  # Log every 10 steps
    evaluation_strategy="steps",  # Evaluate model at regular intervals
    save_steps=500,  # Save model every 500 steps
    eval_steps=500,  # Evaluate model every 500 steps
    warmup_steps=100  # Number of warmup steps for learning rate scheduler
)
```

Figure 3.3.6-1 Hyperparameter Selection

3.6.2 Data Collator:

A data collator helps in preparing batches of data during training. It handles the input and output sequences.

Example Data Collator:

```
from transformers import DataCollatorForLanguageModeling

data_collator = DataCollatorForLanguageModeling(
    tokenizer=tokenizer,  # Tokenizer to convert text to tokens
    mlm=False  # Masked language modeling is not used for causal models
)
```

3.6.3 Trainer Setup:

The Trainer class from the transformers library simplifies the training and evaluation process.

3.6.3.1 Steps to setup Trainer:

1. **Create Datasets:**
   Convert the list of training and validation data into Hugging Face datasets.

```
from datasets import Dataset

train_dataset = Dataset.from_list(train_data)
val_dataset = Dataset.from_list(val_data)
```

2. **Initialize Trainer:** Pass the model, training arguments, datasets, and data collator to the Trainer.

```
from transformers import Trainer

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    data_collator=data_collator
)
```

Figure 3.3.6-2 Training Argument

3. **Modify Training Script to Use accelerate:** Adjust your training script to utilize accelerate. Below is an example of how to modify the script:

```
# Use accelerator to prepare trainer
trainer = accelerator.prepare(trainer)
```

- After initializing the trainer, add the below code to integrate accelerate.

3.7 Fine-Tuning Process:

The fine-tuning process involves training a pre-trained model on a specific dataset to adapt it to new tasks or improve its performance on specific types of data. Here's a step-by-step explanation of the fine-tuning process for LLaMA3:

3.7.1 Start Training:

Starting the training process is where the model learns from your custom dataset. In this step, the model adjusts its parameters based on the input data to minimize the error in its predictions.

Detailed Steps:

- **Initiate Training:** Use the train method of the Trainer object. This method begins the fine-tuning process using the settings and data you have provided.
- **Monitor Progress:** As the training progresses, you can monitor the loss values. Lower loss values typically indicate that the model is learning well from the data.

Example Code to Start Training:

```
trainer.train()
```

After running this code, the Trainer will use the training dataset to update the model's weights iteratively. The process involves the following:

- **Forward Pass:** The model makes predictions based on the input data.
- **Calculate Loss:** The difference between the predicted values and the actual values (targets) is calculated using a loss function.
- **Backward Pass:** The gradients are computed, which indicate how the model's parameters should be adjusted to reduce the loss.
- **Parameter Update:** The model's parameters are updated using an optimization algorithm like Adam.

3.7.2 Evaluate the Model:

Evaluation is an essential step to check how well the fine-tuned model performs on unseen data. It helps in understanding if the model is generalizing well or if it's overfitting to the training data.

Detailed Steps:

- **Use Validation Data:** The evaluation is typically done using a separate validation dataset that the model hasn't seen during training.

Example Code to Evaluate the Model:

```
results = trainer.evaluate()
print(f"Validation Loss: {results['eval_loss']}")
```

After running this code, the Trainer will:

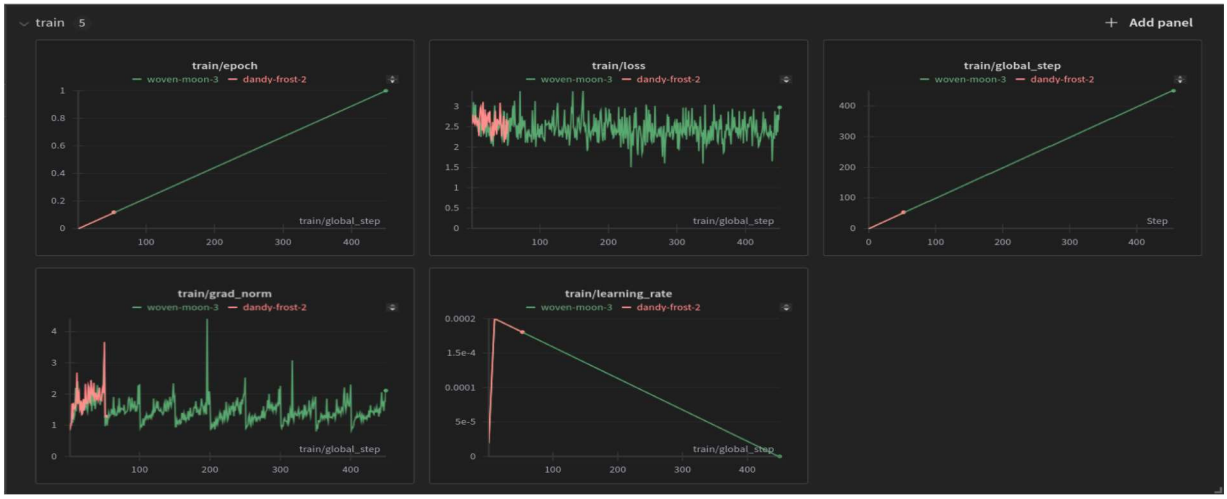
- **Make Predictions:** Use the validation dataset to make predictions.

```
messages = [{"role": "user", "content": "Hello doctor, I always feel weak, can you help me with that?"}]

prompt = tokenizer.apply_chat_template(messages, tokenize=False, add_generation_prompt=True)

inputs = tokenizer(prompt, return_tensors='pt', padding=True, truncation=True).to("cuda")

outputs = model.generate(**inputs, max_length=150, num_return_sequences=1)

text = tokenizer.decode(outputs[0], skip_special_tokens=True)

print(text.split("assistant")[1])
```

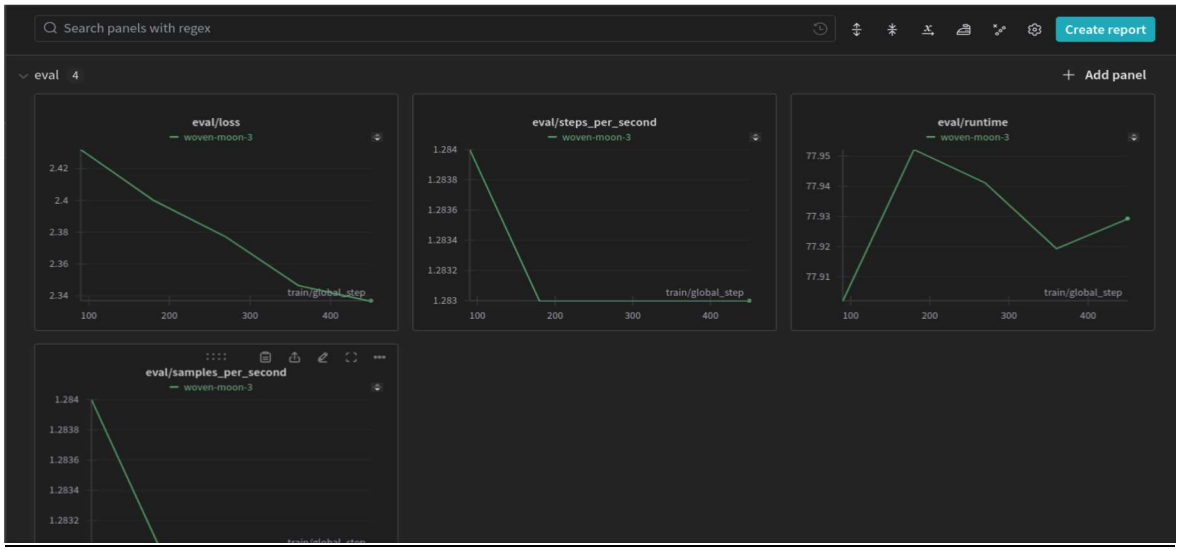
```
Hello. For more information consult a family physician online -->https://www.icliniq.com/ask-a-doctor-online/family-physician-onlineAll the bes
t. For more information consult a family physician online -->https://www.icliniq.com/ask-a-doctor-online/family-physician-onlineAll the best. F
or more information consult a family physician online -->https://www.icliniq.com/ask-a-doctor-online/family-physician-onlineAll the best. For m
ore information consult a family physician online -->https://www.icliniq.com/ask-a-doctor-online/family-physician-online
```

- **Calculate Evaluation Metrics:** Compare the model's predictions with the actual values to compute evaluation metrics like loss.

Train Evaluation Metrics

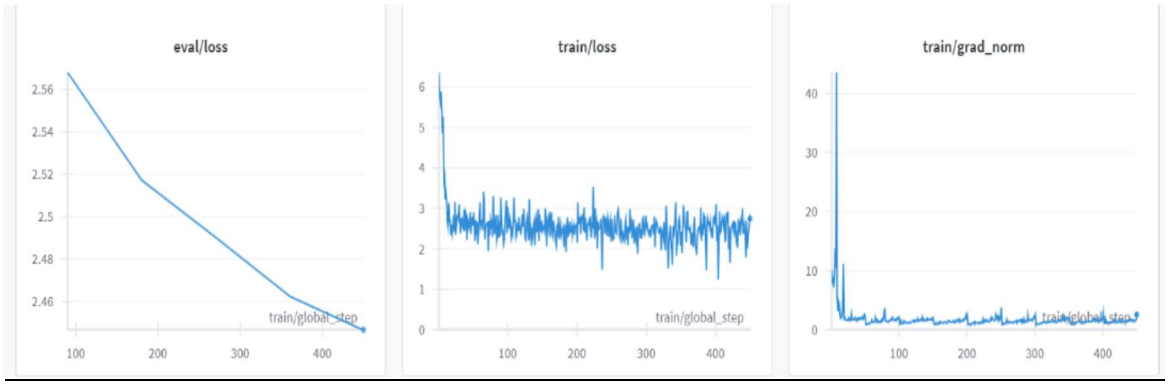


Evaluation Score



System Score

Model Performance Metrics



Comparison of Fine Tune model vs Base Model : The dataset on which the newer fine tune model is made so to get answers on the specific questions which the Base model is not trained on and gives the more accuracy in same alignment so the as an user we get improve the efficiency of our Job.

3.7.3 Save Fine-Tuned Model:

Saving the fine-tuned model is crucial for future use. Once the model is saved, it can be loaded and used for inference without retraining.

Detailed Steps:

- **Save Model Weights:** The model's parameters (weights) are saved to a file.
- **Save Tokenizer:** The tokenizer, which is used to convert text to tokens and vice versa, is also saved.
    - **Tokenizer:** The tokenizer is saved separately and independently from the model. It is crucial to save the tokenizer alongside the model to ensure that the model can properly tokenize inputs in the same way it was during training.

Example Code to Save the Model and Tokenizer:

```
trainer.save_model("./fine-tuned-llama3")
tokenizer.save_pretrained("./fine-tuned-llama3")
```

3.7-1 Fine Tune Model

After running this code, the Trainer will:

- **Save Model:** Write the model's weights to the specified directory (". /fine-tuned-llama3").
- **Save Tokenizer:** Write the tokenizer configuration to the same directory.

Summary of the Fine-Tuning Process:

1. **Start Training:** Initiate the training process to update the model's parameters based on the custom dataset.
2. **Evaluate the Model:** Assess the model's performance on a validation dataset to ensure it is learning correctly and not overfitting.
3. **Save the Fine-Tuned Model:** Save the trained model and tokenizer for future use.

The LLaMA3 model has been successfully fine-tuned using the Low Rank Adaptation technique on the custom dataset. The model is now better adapted to respond to the specific project-related queries, enhancing its performance for use case.

3.7.4 Quantization of fine-tuned LLAMA-3-8b/Gemma -2b-IT

Quantization:-

Quantization refers to a method of compression where values of higher precision are converted to values of lower precision. In the context of large language models (LLMs), this process involves adjusting the precision of model weights and activations, resulting in reduced memory usage. While this transformation can affect the model's performance, including its accuracy, the decision to quantize often depends on specific application needs. In some scenarios, it is possible to maintain performance levels close to the original with much lower precision. The benefits of quantization include decreased demands on memory bandwidth and enhanced cache efficiency.

Quantization Work: -

Quantization functions by reducing the precision of numerical values used within models, such as large language models (LLMs). Typically, LLMs are trained using either full precision (float32) or half precision (float16) floating point numbers, with a float16 taking up 16 bits or 2 bytes. Consequently, a model with one billion parameters trained in float16 would require two gigabytes of memory. Quantization involves mapping these full or half precision values (ranging from [min, max] for a given datatype like float32) to lower precision formats such as float16 or INT4 (4-bit integer). This is often achieved by converting from float32 to an 8-bit integer format (INT8).

The impact of quantization on an LLM's quality varies depending on the specific quantization technique employed. This conversion allows for more efficient use of memory and processing power, although it may affect the model's performance depending on how the reduction in precision is managed.

In our case we are using 4 bit quantization so that it can run on low end PCs with normal specifications and no GPU.

Converting Safe tensors to GGUF model format

Then Quantizing the GGUF model.

Regular laptops don't have enough RAM and GPU memory to load the entire model, so we have to quantify the GGUF model, reducing the 16 GB model to around 4-5 GB.

Use llama.cpp for quantization of model. The quantize script requires a GGUF model directory, output file directory, and quantization method. We are converting the model using the Q4_K_M method.

3.8 Resource Requirements

3.8.1 Hardware Requirements

- **Computational Resources**: Training LLM model requires high-performance computational resources. Recommended hardware includes:

    - **GPUs**: NVIDIA A100, V100, or similar high-end GPUs with substantial VRAM (16GB or more). GPUs accelerate the training process by enabling parallel processing of large-scale data.
    - **TPUs**: Tensor Processing Units (TPUs) for optimized training of large models. TPUs provide specialized hardware acceleration for tensor computations.
    - **CPUs**: Multi-core CPUs for data preprocessing and auxiliary tasks. High-performance CPUs with multiple cores (e.g., Intel Xeon or AMD Ryzen) ensure efficient handling of non-GPU tasks.
- **Storage**: Adequate storage capacity for managing the model, training data, and intermediate results. This includes:
    - **SSD Storage**: High-speed SSDs with large capacity (1TB or more) for rapid data access and processing.
    - **Backup Storage**: Reliable backup solutions to safeguard against data loss, including cloud storage or external hard drives.
- **Memory**: Sufficient RAM (64GB or more) to handle large datasets and ensure smooth operation during training and evaluation.

Ensuring that hardware is optimized both prior to and after development is crucial for efficient model training and deployment. Due to GPU memory constraints, we employed strategies such as using 4-bit precision and integrating adapter layers with a minimal number of parameters to enhance speed and memory efficiency.

3.8.2 Software Requirements

- **Operating System:** A suitable operating system is essential for running the model and its associated software. Preferred options include:
    - **Linux:** Widely used in machine learning due to its compatibility with various frameworks and ease of configuration. Popular distributions include Ubuntu and CentOS.
    - **Windows:** An alternative option, though Linux is generally favored for better performance and broader compatibility with machine learning tools.

- **Machine Learning Frameworks:** The following frameworks are required for developing and training LLAMA 3:

- **PyTorch:** The main framework used for training LLAMA 3, known for its support of dynamic computation graphs and efficient model training.
- **Transformers Library:** Hugging Face's Transformers library, utilized for accessing pre-trained models and performing fine-tuning. This library offers a comprehensive set of tools for working with transformer-based models.

- **Data Processing Tools:** These tools are needed for extracting, preprocessing, and transforming text data:

- **PyMuPDF:** Used for accurate extraction of text from PDF documents.
- **NLTK or SpaCy:** Employed for text processing tasks such as tokenization and named entity recognition.

- **Security and Privacy Tools:** To implement data privacy and security measures, the following tools are recommended:

- **Encryption Libraries:** For securing sensitive data during processing. This may include libraries supporting AES encryption or secure data storage solutions.
- **Privacy Tools:** Techniques to ensure data privacy, such as frameworks for differential privacy or federated learning libraries【1】.

3.8.3 Deployment

1. **Infrastructure Setup**: Establish the necessary infrastructure for deploying the fine-tuned LLM, considering factors like scalability, latency, and security.
2. **Integration**: Integrate the model into the existing systems and workflows of the organization, ensuring seamless interaction with other components.
3. **Monitoring and Maintenance**: Set up monitoring tools to track the model's performance in real-time, addressing any issues promptly. Plan for regular updates and maintenance to keep the model up-to-date with evolving domain knowledge.

CHAPTER 4

RAG PIPELINE

4.1 Retrieval-Augmented Generation:-

Retrieval-Augmented Generation (RAG) is a technique designed to enhance the outputs of large language models (LLMs) by allowing them to refer to an external, authoritative knowledge base beyond their initial training data before generating responses. LLMs, which are trained on extensive datasets and utilize billions of parameters, are capable of performing various tasks such as answering questions, language translation, and text completion. RAG further amplifies these capabilities by tailoring the model's outputs to specific domains or an organization's internal knowledge sources, without necessitating retraining of the model. This approach is cost-effective and enhances the relevance, accuracy, and utility of LLM outputs across different applications.

4.2 Challenges of LLM which are overcome:

1. **Misinformation:** LLMs sometimes generate false information when they lack the correct answer.
2. **Outdated or Generic Responses:** They may provide outdated or overly broad information when a specific and current answer is needed.
3. **Non-Authoritative Sources:** Responses may be created using non-reliable sources, which can undermine the quality of the information.
4. **Terminology Confusion:** Inconsistencies can arise when different training sources use the same terminology to describe different concepts, leading to inaccurate responses.

4.3 RAG Techniques

RAG technique can help as pre-trained LLMs access to very specific information as additional context when answering our questions.

An overview of RAG pipeline is shown as in picture below: -

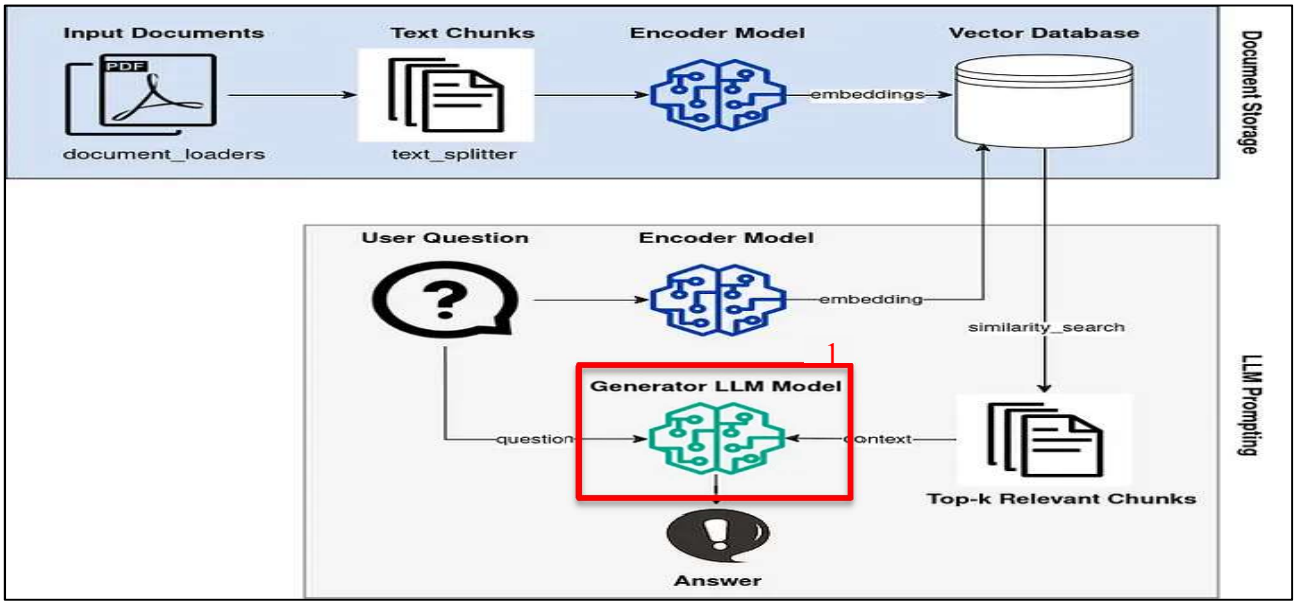


Figure 4.3-1 RAG PIPELINE

The term "Retrieval Augmented Generation "(RAG) comes from the research paper [18] from the year 2020 by researchers at Facebook AI Research, University College London and New York University as shown in Figure below.

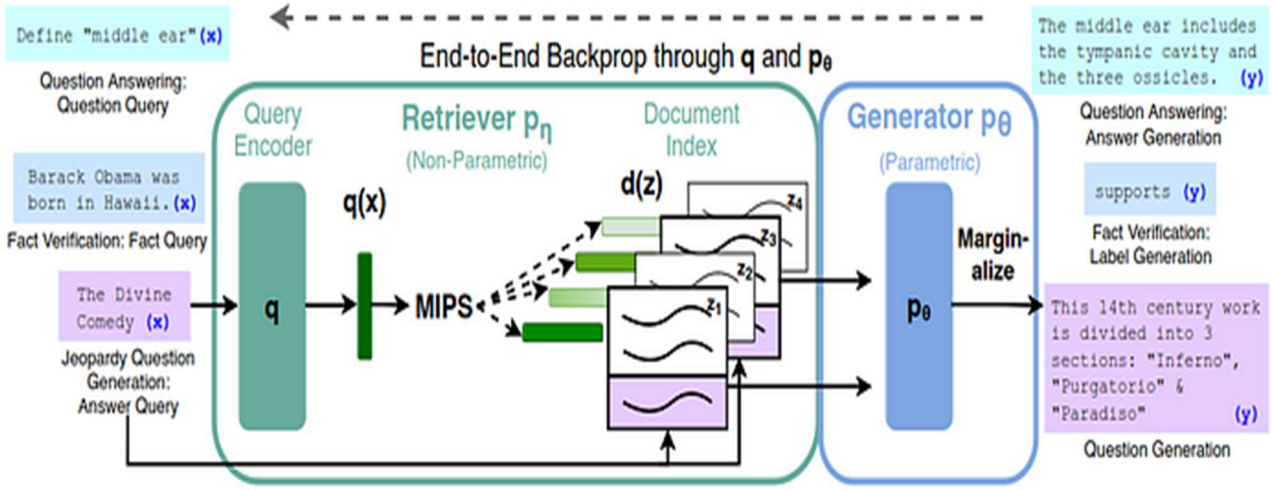


Figure 4.3-2RAG Overview from the original paper. Image by P Lewis et ai.

The above figure described about the bunch of text documents z_i from web source, transform into dense vector representation d(z) which we called embedding by using the embedding model. When user ask question x (this also get transformed into embedding vector q(x) by using the same encoder model. By finding the similar vector to q(x) from all available d(z) using similarity matric. This term we say as **Retriever Component** where we encode our question and find the similar document in our knowledge bank or database and the **Generator Component** from our given question and the additional context from the retrieved documents which we feed into the LLM.
Generator is usually an encoder-decoder or decoder-only LLM as marked 1 in Figure 4.2.1. The generator is an LLM that take question in term of text as input and produce new text as output. In our case we are using the Fine Tuned LLM model as "**fine_tuned_law"** which is small in size and quantized using **4-bit quantization** as specifically fine-tuned over the domain specific dataset to enhancement of the accuracy of our task.

Here is the visual summary of the model inference process, how actually it is working in backend:-

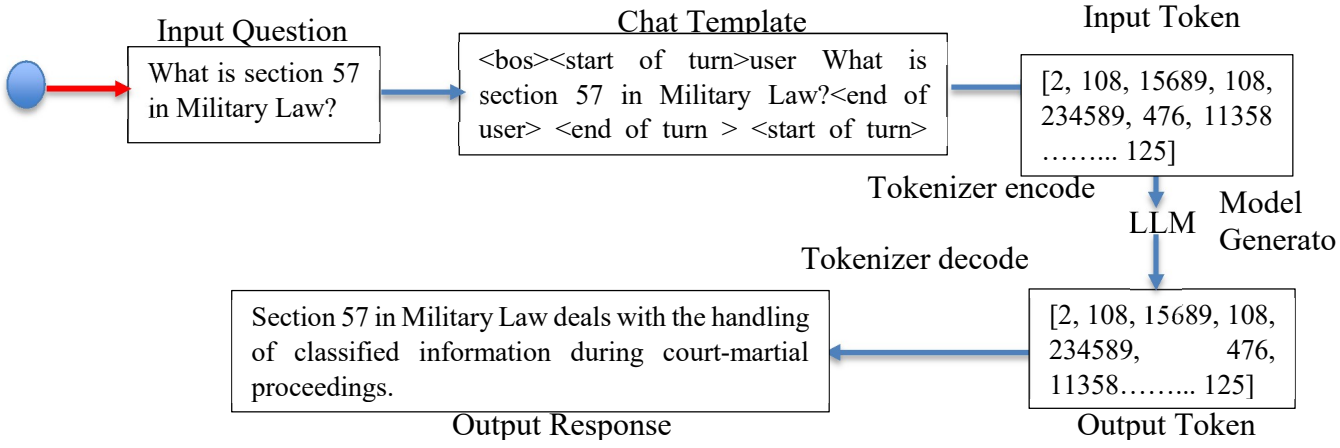


Figure 4.3-3 Model Inference Overview

Here is breakdown related to the internal functions as RAG performed, for every file in the specified directory are as given below: -

1. Split Text into Paragraphs: - The retrieved text from the pdf into paragraphs using a specified separator.
2. Paragraph chunking into Words:- the paragraphs are split into words and create the chunks of these words without exceeding a specified token count(chunk size).
3. Refine Chunks – this is the process when chunk size exceeds the chunk size then further split of using the regular expression based on punctuation. And merge sub chunks if required to optimize the chunk size.
4. Apply Overlap: -multiple chunks sequence, the overlap between them to ensure contextual continuity.
5. Compile and Return Chunks – loop over every final chunk, we assign it a unique ID to map to the text and metadata of related chunk and finally assign chunk dictionary to particular document ID.
6. Indexing – the process to create the document store, we need to create a vector store. In our case we are using **bge-small-en-v1.5** as embedding model.
7. Retrieval - Here we embed the query string and calculate the COSINE Similarity between the query vector and the chunk vector **np.linalg.norm** is used to compute Euclidian norm (L2 norm) of the vectors, which is required for COSINE Similarity.

   COSINE_SIMILARITY = (A.B)/||A|| ||B|| where A & B are vectors and ||A|| and ||B|| are their norms.

These all step reflect the overflow that can be represent as:-

Query -> query embedding -> similarity search-> retrieval -> context -> answering

a. query: The user may not write a good query
b. query embedding: The query model does not create a good representative embedding
c. similarity search: Similarity search misses the most relevant information from the document source or the document source does not have the relevant information. To compute the similarity between vectors, we can choose multiple method as DistancStrategy, EUCLIDIAN_DISTANCE, COSINE SIMILARITY and DOT_PRODUCT.
d. retrieval: Most of the retrieved information is irrelevant or lacking context
e. context: While answering the query may use our past knowledge or not properly use the information from the retrieved context

Here, we use a vector database to store embedding. Using similarity search (cosine similarity between vectors) we fetch similar data which is most closely associated with the dataset.

i.) The Fine-tuned model trained is used for the Retrieval Augmentation Retrieval (RAG).
ii.) Documents are stored in the database and when query is asked it is checked with the vector database.

4.4 RAG EVALAUTIONS

4.4.1 BLEU (Bilingual Evaluation Understudy) SCORES:

The BLEU score is an established metric for evaluating machine translation, where text is translated from one language to another automatically. This metric compares machine-generated translations to several human-provided reference translations to assess translation quality.

How does BLEU score work?

The BLEU score quantifies the similarity between a machine translation and human reference translations by examining n-grams, which are consecutive word sequences of varying lengths such as unigrams (one word), bigrams (two words), and trigrams (three words). It calculates the precision of these n-grams in the translated text relative to the references, adjusting for translation brevity with a penalty to discourage overly short translations.

The BLEU scoreformula is

BLEU = BP * exp(Σ pn)

Finally we got the BLEU Score for our LLM fine tune model result is 30**.**

4.4.2 Recall-Oriented Understudy for Gisting Evaluation Score - ROUGE:

ROUGE scores are primarily used to evaluate text summarization quality, aiming to generate a succinct summary from a longer text. This metric measures how well the machine-generated summary captures the essence of human-written reference summaries.

ROUGE score work:-

ROUGE scores are determined by the overlap of n-grams and word sequences that appear both in the machine-generated summary and the reference summaries, focusing primarily on recall to assess the extent to which important content is captured.

The ROUGE score is as follows:

ROUGE = Σ (Recall of n-grams)

ROUGE-L score is **48.3** for LLM model.

Advantages of Fine-Tuning Over Retrieval-Augmented Generation (RAG):

1. **Architecture**: Unlike RAG, which integrates retrieval into the model architecture, fine-tuning directly adjusts a pre-trained LLM to better suit specific tasks.
2. **Training Data**: Fine-tuning utilizes task-specific labeled data, whereas RAG leverages both supervised data for retrieval and generation.
3. **Model Customization**: Fine-tuning allows for more tailored adjustments to the model's writing style and behavior, while RAG prioritizes enhancing information retrieval capabilities.
4. **Accuracy**: Generally, fine-tuning yields higher accuracy in specialized tasks compared to RAG.

4.5 User Interface as GUI (Streamlit)

The Graphical User Interface (GUI) serves as the frontend representation for the user, enabling interaction with the backend systems through a visually intuitive platform. Streamlit was chosen as the GUI framework for this project due to its simplicity, real-time interactivity, and ease of integration with Python-based machine learning models.

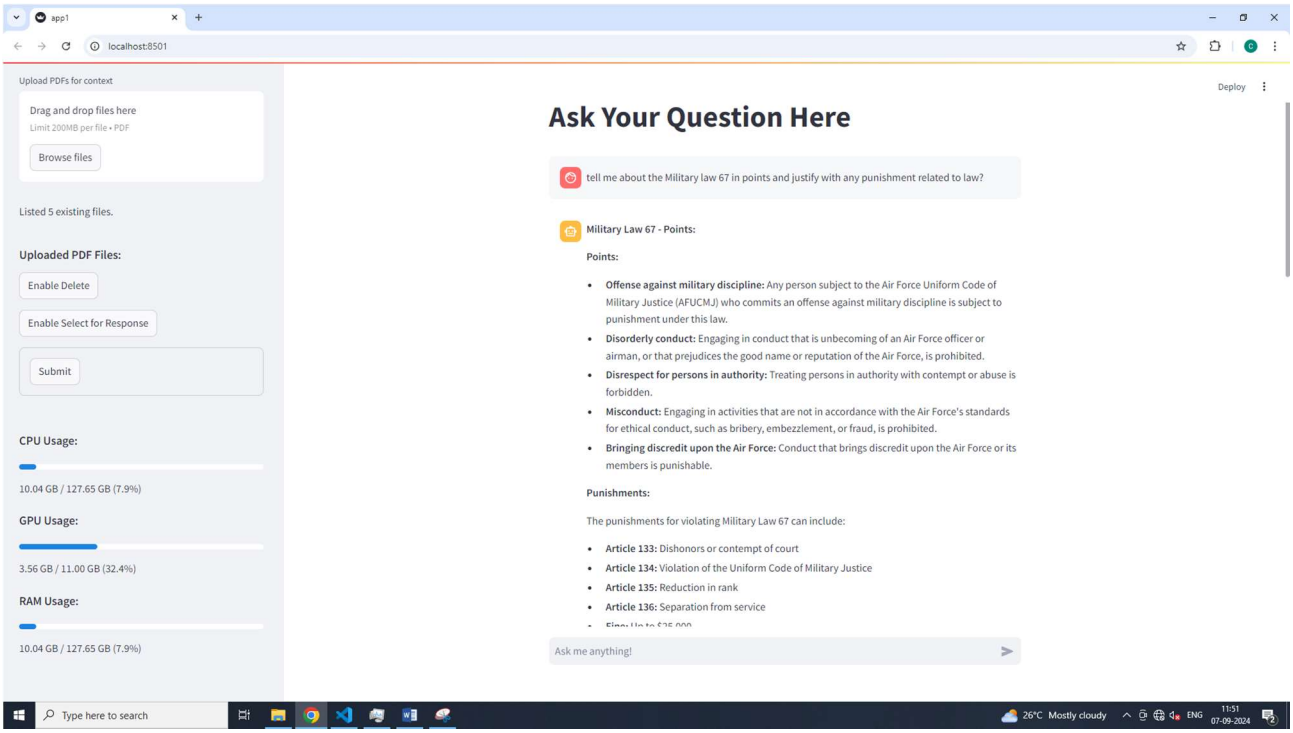


Figure 4.5-1GUI Interface

4.5.1 Key Features of the Streamlit GUI:

- **User Query and Response Handling:** The GUI allows users to input queries and receive responses in real-time, leveraging a backend chat model. This interaction is facilitated through a streamlined interface that supports dynamic text input and response generation.

- **Document Management:** Users can upload, select, and delete documents directly through the GUI. This feature is essential for Retrieval-Augmented Generation (RAG), where the backend system retrieves context from selected documents to enhance response accuracy.

- **Dynamic Document Selection:** A distinctive feature of the GUI is its ability to allow users to toggle between selecting documents for context and deleting them. The system maintains a persistent state for selected documents, highlighted in the GUI, which provides visual feedback to users about their choices.

- **File Upload and Listing:** Uploaded files are stored in a designated directory (rag-data), and the GUI lists these files with internal scrolling, ensuring a compact and organized presentation. The system handles duplicate file checks and provides appropriate feedback to users.

- **Performance Monitoring:** The sidebar of the GUI includes real-time memory usage monitoring for CPU, GPU, and RAM, which is crucial for managing resource-intensive tasks like document processing and model inference. This feature helps in identifying and mitigating performance bottlenecks during operation.

- **Backend Integration with Streamlit:** The backend comprises key functionalities such as loading the chat model, encoder initialization, file handling, document database setup using Faiss, and response generation. Each of these components is seamlessly integrated into the Streamlit interface, allowing users to operate complex backend processes through simple GUI interactions.

- **Real-Time Feedback and Error Handling:** Error handling mechanisms are embedded within the GUI to capture and log issues such as failures in loading models, saving files, or performing similarity searches. This approach not only improves user experience by providing clear feedback but also aids in debugging and maintaining the system.

- **Interactive Chat Flow:** The chat interface dynamically updates with user queries and model responses, mimicking a conversation flow. This design enhances user engagement and makes the interaction more relatable and intuitive.

- **Reference Display:** When generating responses, the system references documents that contributed to the answer, providing transparency and traceability for users. This feature is particularly valuable in applications requiring evidence-based responses, such as legal queries.

4.5.2 Implementation

4.5.2.1 Document Upload and Management

The code snippet below shows how the application handles document uploads. Uploaded files are saved to a specified directory (rag-data). If a file with the same name already exists, a warning is displayed to the user.

```
def save_file(uploaded_file):
    file_path = os.path.join(FILES_DIR, uploaded_file.name)
    if os.path.exists(file_path):
        st.warning(f"File '{uploaded_file.name}' already exists.")
        return None
    try:
        os.makedirs(os.path.dirname(file_path), exist_ok=True)
        with open(file_path, "wb") as f:
            f.write(uploaded_file.getbuffer())
        logger.info(f"File '{uploaded_file.name}' saved successfully.")
        st.write(f"File '{uploaded_file.name}' saved successfully.")
        return file_path
    except Exception as e:
        logger.error(f"Failed to save file '{uploaded_file.name}': {e}")
        st.error(f"Failed to save file '{uploaded_file.name}'. Please try again.")
        return None
```

Figure 4.5-2 Document management using GUI

This function, save_file, checks if the uploaded file already exists and saves new files to the designated directory. It handles file writing operations and logs relevant messages, enhancing the user experience with appropriate feedback and error handling.

4.5.2.2 Dynamic Document Selection

The GUI allows users to manage document selection dynamically for response generation. Users can toggle between selecting documents for response context or enabling document deletion. The following code snippet illustrates this functionality:

```
def display_existing_files(existing_files):
    if existing_files:
        st.write("### Uploaded PDF Files:")
        for file_name in existing_files:
            is_selected = file_name in st.session_state.selected_files
            selected_style = "selected-file" if is_selected else ""
            if st.checkbox(f"Select {file_name}", value=is_selected):
                if file_name not in st.session_state.selected_files:
                    st.session_state.selected_files.append(file_name)
            else:
                if file_name in st.session_state.selected_files:
                    st.session_state.selected_files.remove(file_name)
```

Figure 4.5-3 Display Existing File at GUI

Code manages the display and selection of existing files within the GUI. Checkboxes allow users to select files for context in query responses. Selected files are highlighted, providing a clear visual indication of the user's choices.

4.5.2.3 Response Generation and Reference Display

The core functionality of the GUI is to handle user queries and generate responses using a fine-tuned model. Below is a simplified code snippet demonstrating how the application processes user input and generates responses:

```
if prompt := st.chat_input("Ask me anything!"):
    st.session_state.messages.append({"role": "user", "content": prompt})
    with st.chat_message("user"):
        st.markdown(prompt)
    with st.chat_message("assistant"):
        response_text = model.generate(user_prompt)
        st.markdown(response_text)
```

The captures user queries and appends them to the chat history. It then utilizes the fine-tuned model to generate responses based on the query, which are displayed in the GUI. This approach provides a seamless chat experience, enhancing user interaction.

Conclusion

The implementation of Retrieval-Augmented Generation (RAG) to enhance the capabilities of Large Language Models (LLMs) by integrating them with an external, authoritative knowledge base. By dynamically retrieving relevant documents to provide context for response generation, the RAG system addresses common LLM challenges such as outdated, inaccurate, or irrelevant outputs.

The use of a Streamlit-based Graphical User Interface (GUI) further augmented the user experience by providing an interactive platform for real-time query processing, document management, and system monitoring. The GUI's dynamic document selection and error handling capabilities made the system more user-friendly and efficient.

Moving forward, the project can be enhanced by refining the document retrieval processes, optimizing the embedding models, and updating the document store to maintain relevance with new information. Additionally, scalability and security measures need to be emphasized to manage larger data volumes and protect sensitive information. Regular system evaluations using metrics like BLEU and ROUGE scores will help monitor performance and integrate user feedback for continuous improvement.

In conclusion, the integration of RAG with a Streamlit GUI presents a robust framework for enhancing the accuracy and relevance of responses generated by LLMs, proving essential for specialized domains requiring up-to-date and precise information.

CHAPTER 5

DOCUMENTATION AND DISSEMINATION

This chapter details the documentation of the development process, preparation and finalization of the dissertation, and a reflection on the challenges encountered along with their respective solutions during the project. The focus is on ensuring that the development journey, technical details, and the learning outcomes are thoroughly documented and communicated.

5.1 Document the development process:

Documenting the development process was a crucial step in capturing the comprehensive efforts and methodologies applied throughout the project. This included:

5.1.1 Project Planning and Setup:

Initial project setup involved defining the scope, selecting appropriate models, and setting up the necessary hardware and software environments. The selection process was driven by the specific needs of legal query handling and the constraints imposed by the available resources. Here I want to highlight the mathematic calculation for the selection during the LLM that we should consider while we are going to plan our project:-

To use LLM, we have to fit the model into memory.

Using 32 bit floating point (FP32) - 1 Parameter require 4 bytes of RAM

Using 16 bit Quantization (BFLOAT16 or FP16) – we can reduce it to 2 bytes of RAM for 1 Parameter.

Using 8 bit integer (INT8) – we need 1 byte of RAM for 1 Parameter.

So storing 1 billion (1B) parameter of a LLM in memory require approx. 4 GB of memory for 32 bit full precision , 2GB for 16 bit half Precision and 1 GB for 8 bit precision. Here we can explain with the help of example : - **6 GB of memory which could fit around 1.5B parameter @32bit , 3B parameter @16 bit and 6 B parameter @ 8bit** .However , just loading CUDA kernel can consume 1-2Gb of memory , So while in practice we cannot fill the entire GPU memory with just parameter. Training an LLM require even more GPU RAM, because optimizer states, Gradient and forward activation require additional memory per parameter.so finally we can say to use **1GB parameter = 2GB @16bit and 1GB @8bit** and we should adopt this as a thumb rule when choose LLM.

Through the implementation of quantization and other optimization strategies, the project has managed to balance the trade-offs between model performance and computational efficiency. This has enabled the deployment of a robust AI solution on limited hardware resources without compromising response quality.

5.1.2 Iterative Development:

The project followed an iterative development approach, where continuous testing, validation, and refinement were employed in term of model selection setting of parameter, making model quantized and switched over multiple model and failure and achievement in the training /validation/ Accuracy calculation. Each iteration aimed to address specific challenges, refine model outputs, and improve overall system performance.

5.1.3 System Design and Integration:

Detailed documentation of the system architecture was created, outlining the integration between the RAG components, the fine-tuned LLM, and the Streamlit-based GUI. This included the data flow, backend processing, and user interaction points.

The architecture of the system integrates advanced AI techniques with a user-friendly interface, ensuring that users can easily interact with the system to receive accurate legal information.

5.1.4 Testing and Evaluation:

Comprehensive testing was performed at each stage to evaluate the performance, accuracy, and reliability of the system. Documentation included test cases, results, and the adjustments made based on the findings.

5.1.5 Deployment and User Interface:

The deployment process was documented, highlighting the setup of the Streamlit GUI, user accessibility features, and performance monitoring aspects. This also covered the deployment environments used, including local hardware and cloud platforms.

5.2 Prepare and finalize the dissertation

The preparation and finalization of the dissertation involved compiling all the research, development insights, and results into a cohesive document. Key activities included:

5.2.1 Drafting and Structuring:

The dissertation was structured to logically present the project objectives, methodology, implementation, and findings. Each chapter was carefully drafted to ensure clarity and a smooth flow of information.

5.2.2 Incorporation of Code and Technical Details:

Code snippets and technical details were included to provide a clear representation of the implementation. This was done in a way that aligns with academic standards, ensuring that the complexity of the work was adequately conveyed to the reader.

5.2.3 Review and Revisions:

Multiple rounds of review were conducted, both internally and with supervisors, to refine the content and address any gaps or areas requiring additional explanation. Feedback was incorporated to enhance the quality and coherence of the dissertation.

5.3 Challenges and Solutions

This section provides a detailed account of the challenges encountered during the project and the solutions implemented to overcome them:

5.3.1 Hardware Constraints:

The initial hardware setup with 11GB Graphic card and 128GB RAM was insufficient for training large models like LLAMA 3 model with 8B parameter, all leading to frequent OOM errors. As far as security concern it was restrict to me to develop my project with online platform and bound me as hardware constraint. So optimizing memory utilization, speeding up training or both is need to understand as a developer is very essential. If we refer the **Model Training anatomy**, we will find the insight the useful for practical implication.

When we train a large language model, there are two aspects that should be considered at the same time:

- Data Through put/ training time
- Model performance

Maximizing the throughputs (samples/seconds) lead to lower training cost. This can be achieved by utilizing the GPU as much as possible and thus filling the GPU memory to its limits of the GPU memory optimization techniques, such as gradient accumulation. The method and tools covered in the reference help a lot while we work with such large model and the classification based on the effect on the training process as below and will provide the cumulative effect: -

Ser No	Method/Tool	Improves training speed	Optimize memory utilization
1.	Batch size choice	Yes	Yes
2.	Gradient accumulation	No	Yes
3.	Gradient check point	No	Yes
4.	Mixed Precision training	Yes	Yes
5.	Torch empty cache steps	Yes	May be
6.	Optimizer choice	Yes	Yes
7.	Data Preloading	Yes	No
8.	DeepSpeedZero	No	Yes
9.	Torch.compile	Yes	No
10.	Parameter Efficient Fine Tuning(PEFT)	No	Yes

Table 5.3.1-1 Method and tools for efficient training on a single GPU

*Note: - If we used mixed precision with a small model and a large batch size there will be some memory saving but when work with large model and small batch size, the memory will be larger.

5.3.2 Quantization Trade-offs:

Implementing 4-bit quantization helped reduce memory usage but introduced challenges in maintaining model accuracy and response quality.

5.3.3 Out-of-Memory Errors in Training and Validation:

Simultaneous training and validation sessions caused OOM errors due to the high memory demands. Implemented gradient check pointing and reduced batch sizes. Separated training and validation into distinct sessions to manage memory usage more effectively.

5.3.4 Incomplete Responses from the Model:

The model sometimes generated incomplete responses, particularly for long or complex legal queries. Adjusted prompt lengths, token and implemented response streaming techniques to allow the model to generate answers token by token, reducing the likelihood of truncation.

5.3.5 Prompt Design for Legal Contexts:

Creating effective prompts that could guide the model to produce legally accurate and context-specific responses was a significant challenge. Developed and iterated on prompt templates specifically tailored for legal queries, incorporating legal terminology and structure to align with expected outputs.

5.3.6 Consistency in Response Formats:

The model occasionally generated responses in varied formats, which was problematic given the structured nature required for legal outputs. Included format-specific instructions in the prompts and fine-tuned the model with examples of desired output formats. This helped guide the model towards producing more consistent and structured responses.

The project "Design of GenAI Solution for Domain Specific Application" has successfully demonstrated the capability to develop a specialized legal Question Answering (QA) system by leveraging Retrieval-Augmented Generation (RAG) with a fine-tuned Large Language Model (LLM). The system has been meticulously engineered to handle complex legal queries, providing precise and context-aware responses, which is a significant advancement over traditional legal research tools that rely heavily on manual input and navigation. Future improvements could include upgrading hardware resources, further optimizing quantization methods, expanding prompt designs to enhance model accuracy, reliability across a wider range of legal topics, the improvement on data and improvement on RAG Agentic method for enhancement the accuracy. This is not stop

here it is the beginning – the real time responses, suggestion in term of law decision and the multiple question answering for the specific use case can be integrate in future.

The project successfully developed a robust, context-aware legal query system using a RAG framework with a fine-tuned LLM. Future enhancements could focus on upgrading hardware, optimizing quantization methods, and expanding the model's capabilities to handle a broader array of legal topics. The project sets a foundation for more advanced features, including real-time legal suggestions and multi-faceted query handling.

Final Thoughts: This project lays a robust foundation for the future of AI in legal applications, presenting a scalable model that can be expanded and enhanced with emerging AI technologies. The continuation of this work promises not only to refine the capabilities of legal AI systems but also to revolutionize the accessibility and efficiency of legal services globally.

5.4 CHECK LIST

(a)	Is the Cover page in proper format?	Y / N
(b)	Is the Title page in proper format?	Y / N
(c)	Is the Certificate from the Supervisor in proper format? Has it been signed?	Y / N
(d)	Is Abstract included in the Report? Is it properly written?	Y/N
(e)	Does the Table of Contents page include chapter page numbers?	Y / N
	(i) Are the Pages numbered properly?	Y / N
	(ii) Are the Figures numbered properly?	Y / N
	(iii) Are the Tables numbered properly?	Y / N
	(iv) Are the Captions for the Figures and Tables proper?	Y / N
	(v) Are the Appendices numbered?	Y / N
(f)	Does the Report have Conclusion / Recommendations of the work?	Y / N
(g)	Are References/Bibliography given in the Report?	Y / N
(h)	Have the References been cited in the Report?	Y / N
(j)	Is the citation of References / Bibliography in proper format?	Y / N

5.5 REFERENCES

[1] A. Vaswani *et al.*, “Attention Is All You Need,” Aug. 01, 2023, *arXiv*: arXiv:1706.03762. Accessed: Jul. 22, 2024. [Online]. Available: http://arxiv.org/abs/1706.03762

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” Sep. 06, 2013, *arXiv*: arXiv:1301.3781. Accessed: Jul. 23, 2024. [Online]. Available: http://arxiv.org/abs/1301.3781

[3] T. Shi and Z. Liu, “Linking GloVe with word2vec,” Nov. 26, 2014, *arXiv*: arXiv:1411.5595. Accessed: Jul. 23, 2024. [Online]. Available: http://arxiv.org/abs/1411.5595

[4] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to Sequence Learning with Neural Networks,” Dec. 14, 2014, *arXiv*: arXiv:1409.3215. Accessed: Jul. 23, 2024. [Online]. Available: http://arxiv.org/abs/1409.3215

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” May 24, 2019, *arXiv*: arXiv:1810.04805. Accessed: Jul. 23, 2024. [Online]. Available: http://arxiv.org/abs/1810.04805

[6] T. B. Brown *et al.*, “Language Models are Few-Shot Learners,” Jul. 22, 2020, *arXiv*: arXiv:2005.14165. Accessed: Jul. 23, 2024. [Online]. Available: http://arxiv.org/abs/2005.14165

[7] H. Touvron *et al.*, “Llama 2: Open Foundation and Fine-Tuned Chat Models,” Jul. 19, 2023, *arXiv*: arXiv:2307.09288. Accessed: Jul. 23, 2024. [Online]. Available: http://arxiv.org/abs/2307.09288

[8] S. Gururangan *et al.*, “Don’t Stop Pretraining: Adapt Language Models to Domains and Tasks,” May 05, 2020, *arXiv*: arXiv:2004.10964. Accessed: Jul. 23, 2024. [Online]. Available: http://arxiv.org/abs/2004.10964

[9] I. Beltagy, K. Lo, and A. Cohan, “SciBERT: A Pretrained Language Model for Scientific Text,” Sep. 10, 2019, *arXiv*: arXiv:1903.10676. Accessed: Jul. 23, 2024. [Online]. Available: http://arxiv.org/abs/1903.10676

[10] J. Lee *et al.*, “BioBERT: a pre-trained biomedical language representation model for biomedical text mining,” *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, Feb. 2020, doi: 10.1093/bioinformatics/btz682.

[11] A. Abdallah, B. Piryani, and A. Jatowt, “Exploring the State of the Art in Legal QA Systems,” *J. Big Data*, vol. 10, no. 1, p. 127, Aug. 2023, doi: 10.1186/s40537-023-00802-8.

[12] N. Yagnik, J. Jhaveri, V. Sharma, and G. Pila, “MedLM: Exploring Language Models for Medical Question Answering Systems,” Mar. 05, 2024, *arXiv*: arXiv:2401.11389. Accessed: Jul. 23, 2024. [Online]. Available: http://arxiv.org/abs/2401.11389

[13] M. Aitsam, “Differential Privacy Made Easy,” in *2022 International Conference on Emerging Trends in Electrical, Control, and Telecommunication Engineering (ETECTE)*, Lahore, Pakistan: IEEE, Dec. 2022, pp. 1–7. doi: 10.1109/ETECTE55893.2022.10007322.

[14] M. Abadi *et al.*, “Deep Learning with Differential Privacy,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, Oct. 2016, pp. 308–318. doi: 10.1145/2976749.2978318.

[15] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data,” Jan. 26, 2023, *arXiv*: arXiv:1602.05629. Accessed: Jul. 23, 2024. [Online]. Available: http://arxiv.org/abs/1602.05629

[16] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “SQuAD: 100,000+ Questions for Machine Comprehension of Text,” Oct. 10, 2016, *arXiv*: arXiv:1606.05250. Accessed: Jul. 23, 2024. [Online]. Available: http://arxiv.org/abs/1606.05250

[17] C. Alberti, K. Lee, and M. Collins, “A BERT Baseline for the Natural Questions,” Dec. 09, 2019, *arXiv*: arXiv:1901.08634. Accessed: Jul. 23, 2024. [Online]. Available: http://arxiv.org/abs/1901.08634

[18] P. Lewis *et al.*, “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” Apr. 12, 2021, *arXiv*: arXiv:2005.11401. Accessed: Sep. 04, 2024. [Online]. Available: http://arxiv.org/abs/2005.11401