A Comparative Study of Open Lightweight LLM Models -Training, Compute and Chatbot Response Quality

Mitali Utekar

ABSTRACT

With the rapid advancements in Artificial Intelligence (AI), organizations from all scales must adopt AI to sustain their businesses. This shift has led to the highest ever and unprecedented demand for computation power, significantly increasing costs. This research addresses the gap between building models from scratch and relying on black-box, pay-as-you-go AI services by focusing on fine-tuning models on limited infrastructure for a custom dataset. We evaluated the open and lightweight models suitable on an individual or small business level developments by choosing DeepSeek-R1-Distill-Qwen (1.5B), GPT-Neo (1.3B) and TinyLLaMA (1.1B). To train a chatbot on AWS Cloud Services domain with limited resources, we applied Transfer learning in combination with Parameter-Efficient Fine-Tuning (PEFT) techniques such as Low Rank Adapters (LoRA). Model performance was evaluated using training time, GPU utilization, evaluation loss and benchmarks such as BLEU, ROUGE, SBERT Semantic Similarity Score and number of training parameters.

Keywords: DeepSeek, GPT, LLaMA, open-weight, lightweight, Security, LoRA, BLEU, ROUGE, SBERT Semantic Similarity Score, Transfer Learning.

Table of Contents

ABSTRACT **2**

CHAPTER 1 – INTRODUCTION **4**

1.1 Business Problem 4

1.2 Research Problem 5

1.3 Aim 5

1.4 Objective: 5

1.5 Hypothesis 5

1.6 Scope 6

1.7 Limitations 6

1.8 Dissertation Roadmap 6

CHAPTER 2 – LITERATURE REVIEW **7**

2.1 Evolution of LLM's 7

2.2 Chatbot 8

2.3 Fine Tuning Approach 9

2.4 Summary and Research gap 10

CHAPTER 3 – METHODOLOGY **11**

3.1 Business Understanding 11

3.2 Data Understanding 12

3.3 Data Preparation 15

3.4 Modelling 16

3.4.1 Overview of Large Language Models (LLMs) and Transformer Architecture 16

3.4.2 Model Architecture: 18

3.4.3 Fine-Tuning Techniques 21

3.5 Evaluation 25

3.6 Deployment 26

CHAPTER 4 – RESULTS AND DISCUSSION **27**

CHAPTER 5 – CONCLUSIONS AND FUTURE WORK **30**

Conclusion 30

Future Work 30

References 31

Appendices 33

Appendices A: Data Source Used 33

Appendices B: Code and Generated Answers 33

Appendices C: Code 33

CHAPTER 1 – INTRODUCTION

1.1 Business Problem

The industry has evolved from using Natural Language Processing (NLP) from its basic applications such as spam email classification, auto text completion or real-time language translations. Modern LLMs are now capable of performing more complex tasks with additional human-like intelligence. The ease of access to applications such as chatbots have led to a first-hand experience for both technical and non-technical users on interaction with Artificial Intelligence (AI). Customer support chatbots now help in reducing wait time by addressing generic queries prior to escalating for a human intervention, AI assistant such as Alexa can be customized and adapt to users' mood and schedule. Even the depth of these conversations can be fine-tuned to align with an individual personality. Companies across all fields are now forced to adopt AI for sustaining their business but often encounter below problems.

High computational cost:
We have surpassed traditional LLMs and now use transformer-based model architecture, including multiple recurrent transformers, attention layers that demands higher computation resource than ever before. Fine tuning models such as GPT, BERT with billions of parameters require high end resources and even a downstream version of these models are resource intensive.

Data Sources: A model is developed from the context size of data and embeddings and thus both quantity and quality of training data is of huge importance. Thus, some of the best industry models are published only by well-funded organization leveraging the technology they already acquire. Forcing small and mid-tier businesses to either scale the budget at the cost of testing a new technology or train models from scratch on a limited data.

Data privacy: Black-box based AI services often provide limited transparency on backend systems and infrastructure. In data-sensitive sectors such as banking, the usage of benchmark models such as GPT often requires switching to external infrastructure (public cloud) due to their higher computational demands. This leads to transferring of sensitive data outside private infrastructure and limiting adoption of AI in such sectors. Open-weight models DeepSeek help to bridge this gap while also leveraging the power of transfer learning.

The best models even when accessible often comes at a huge cost of integration, restricting democratization of AI across businesses of varying scales. As it is only justifiable for companies to invest in resources if they are confident of benefiting from new technology by testing and evaluating it on immediate available hardware and limited data.

1.2 **Research Problem**

This research aims to address the challenges faced on small business and individual level while adopting AI by creating a domain specific chatbot, using model tuning techniques such as Low Rank Adapters (LoRA) and instruction tuning. It uses open-lightweight models and training platforms easily accessible on internet. Response to human queries include an intelligence aspect that comes at high training cost and resources and thus we tend to evaluate the model performance based on:

- Domain Specific Learnings: Can open-lightweight transfer learning models suffice the business expectations of creating an intelligent chatbot with minimum code and training complexity.
- Training efficiency: Can one train the models using Parameter-efficient fine-tuning techniques such as LoRA in case of limited hardware resources and budget constraints.
- Comparative evaluation: How does instruction tuned open weight model perform against each other in terms of compute, training time and evaluation metrics such as BLEU, ROUGE, SBERT Semantic Similarity scores to measure chatbot accuracy and fluency.
- Deployment feasibility: Can the low parameter-based models be deployed on local machine or on platforms free of cost.

1.3 **Aim**

We aim to create a sophisticated and a domain trained chatbot for AWS cloud platform by comparing the open and lightweight models in terms of compute, training time and response quality in order to evaluate their feasibility for training and deployment when fine-tuned with LoRA.

1.4 **Objective:**

- Identifying models and platforms to fulfil the training resource constrains on small business or an individual level in order to train AWS domain specific chatbot.
- Evaluate model by their ability to answer AWS Cloud domain specific questions with accuracy.
- Test models' ability to address out of the box context questions from pretraining knowledge for efficient communication.
- Achieve best performance of a model in minimum compute by fine-tuning using LoRA and optimize limited resource environment.
- To achieve least training time amongst other models.

1.5 **Hypothesis**

While testing the deployment of models we assume that the lighter models can compete with higher-parameter models in domain specific task, while also leveraging least

compute, training time by using Low Rank Adapters (LoRA) mechanism. Also, that the training dataset of 1.3 MB is assumed to be sufficient to avoid under or over fitting issue.

1.6 Scope

The research scope is centred around creating an AWS Cloud domain specific chatbot encompassing the following key elements:

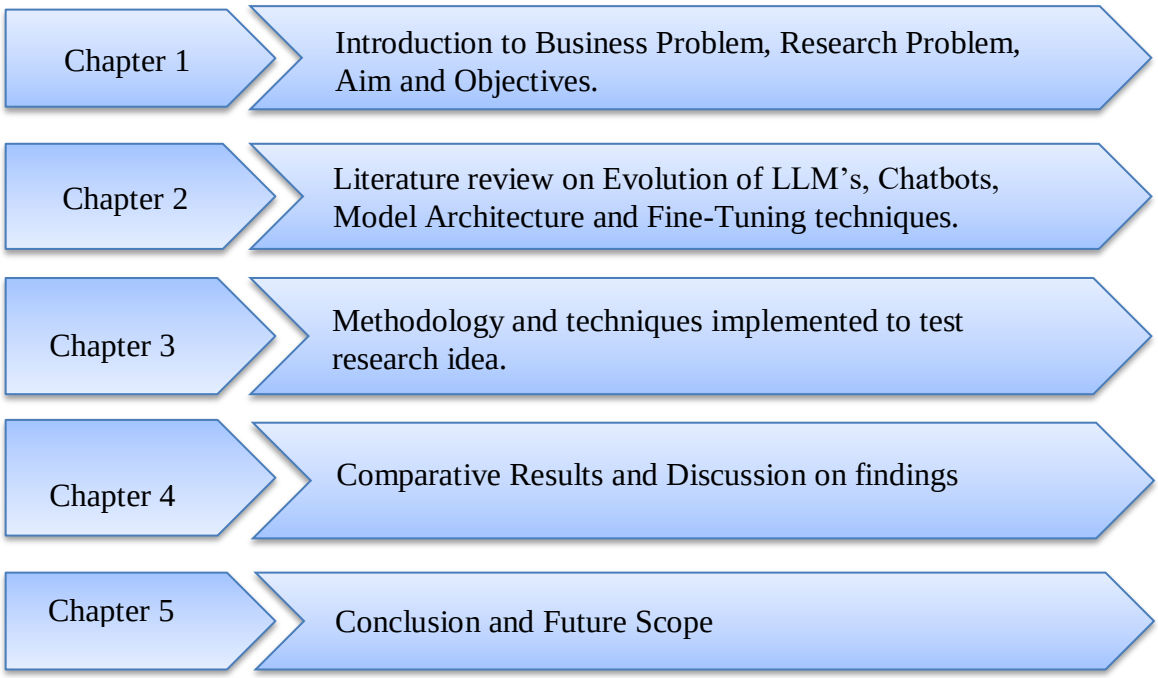
- Primary Model: DeepSeek-R1-Distill-Qwen-1.5B
- Training and fine-tuning method: Low Rank Adapters (LoRA) with instruction tuning.
- Comparative analysis: GPT-Neo-1.3B, TinyLlama-1.1B-Chat-v1.0
- Dataset: AWS FAQ (Alpaca-style).
- Evaluation metrics: BLEU, ROUGE, SBERT Semantic Similarity Score, GPU usage, Training time.
- Deployment: Local server or cloud platform

1.7 Limitations

- Multiple Google Colab session can affect compute association to each session thus influencing results with total available GPU resources in a Colab setup.
- Standard evaluation metrics used may not always align with human evaluation.
- Evaluation limited to English language tasks.

1.8 Dissertation Roadmap

For execution of the dissertation the project follows below roadmap:



CHAPTER 2 – LITERATURE REVIEW

As we aim to test the best open-source lightweight models to create a domain specific chatbot under a limited compute, the goal is to research previous work related to best models and fine-tuning methods that can be applied in our research. Our aim is to also evaluate new models such as DeepSeek and since there is not enough research papers published on this model and on the less compute resource it will be interesting to apply the previous work of researchers on similar but lightweight model to match the requirements on a small business and individual level. Section 2.1 describes the evolution of LLM's though years and the recent published research findings. Section 2.2 explains importance of chatbot and their ease of accessibility to population. Section 2.3 includes different fine-tuning methods implemented by researchers and their outcomes. Section 2.4 includes identification of the gaps in previous research, providing a direction for new research ideas.

2.1 Evolution of LLM's

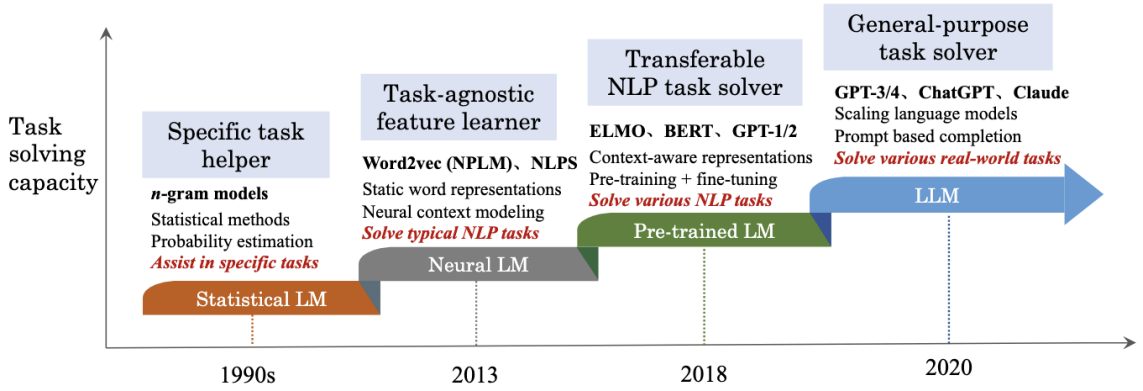


Figure 1: An evolution process of the four generations of language models (LM) from the perspective of task solving capacity. Note that the time period for each stage may not be very accurate, and we set the time mainly according to the publish date of the most representative studies at each stage. For neural language models, we abbreviate the paper titles of two representative studies to name the two approaches: NPLM ("A neural probabilistic language model") and NLPS ("Natural language processing (almost) from scratch"). Due to the space limitation, we don't list all representative studies in this figure. (Zhao, et al., 2025)

Figure 1 highlights the evolution of LLM's through several year as with the aim to expand and democratize AI across industries there has been a need for complex and generalized models that has led to research and development in NLP. The very initial Statistical language Models(SLMs) (based on Markov assumption predicts the next word based on recent context, but suffers from curse of dimensionality and sparsity due to the need for larger transitional probability to be estimated and calculates zero probabilities to unseen words) to Neural Language Models(NLMs) which deals with data sparsity using multi-layer perceptron(MLP)(embedding vectors) and Recurrent Neural Network(RNN) (processes time series data) focusing on feature representations (Zhao, et al., 2025) (Wei, et al., 2022) (Minaee, et al., 2025), however these designs can be more complex in case of representing full sentences, leading to vanishing gradient problem and nearby context bias,

fixed context size limit, slower sequential processing as mentioned in (Li & Sarwate, 2025). NLMs being task specific models they are followed by Pre-trained Language Models (PLMs) which are more task agnostic and represents context aware word using bidirectional LSTM (biLSTM), Large Language Models (LLMs) are large-sized PLMs with scaling of billions of parameters showcased emergent abilities in solving complex tasks. (Zhao, et al., 2025) (Wei, et al., 2022) (Minaee, et al., 2025).

The recent work avoids recurrent neural network and tries to draw dependencies between input and output while relying completely on attention mechanism. A sequence of hidden state h_t is calculated as a function of previous hidden state h_{t-1} and the input from position t. The convolution neural network computes hidden representation in parallel for all inputs and outputs, increasing the number of operations to relate signals from input and output thus making it more difficult to learn dependencies between distant points. The transformer however restricts it to constant number of operations, relating distant positions using self-attention an disprove resolution loss with multi-head-attention (sentence divided into equal pieces with attention model for each m piece (Li & Sarwate, 2025)) which is applied on each transformer layer (Russell & Norvig, 2021). This was a ground baking finding published in the paper 'Attention Is All You Need' and making Transformer the first model architecture to rely entirely on self-attention without RNN or CNN. (Vaswani, et al., 2023)

The transformer architecture has proven to be most successful due to its 'attention mechanism' which directs model to generate each output token while focusing on distinct segments of input sequences. (Inouye, et al., 2024)

To improve models' performance to avoid generating false, biased and any kind of harmful responses Alignment tuning is now employed with Reinforcement learning and reward modelling for model to learn from its own generated responses by updating its parameters (Naveed, et al., 2024).

2.2 Chatbot

Over the years the methods of teaching and learning have also evolved with technology. It has shifted from transferring knowledge by verbal storytelling, to sharing knowledge via books, followed by computers and mobile phones. Speech and chat still remain as one of the meaningful forms of communication (Handoyo, et al., 2018).

Chatbots have become a new way of learning, interacting, problem-solving and decision making and thus creating a hybrid cognitive system with collaboration between human intelligence and machine (Wei, et al., 2022). Allowing dynamic content interaction and providing prompt feedback (Minaee, et al., 2025), leading to more engagement and learnings. They have also emerged as a very convenient and a first stepping stone for industry to adopt AI. However, at times the cost and resource requirements for its implementations becomes a huge huddle to even mid-top tier companies for its adoption.

2.3 Fine Tuning Approach

LoRA:
Some of the methods used for domain specific tuning of the model mentioned under (Hu, et al., 2021) are as Fine-Tuning (FT) in which all model parameters are updated and thus needs highest hardware resources. As studied in in (Hu, et al., 2021)Bias-only or BitFit baseline only bias vectors are trained while freezing everything else. Prefix-embedding tuning (PreEmbed) under which special token which are not a part of model's vocabulary are inserted as 'prefixing' or 'infixing'. Adapter tuning methods inserts adapter layer inside each transformer layer at different modules. LoRA adds rank decomposition matrices in parallel to existing weights (Hu, et al., 2021).

As studied under (Russell & Norvig, 2021) (Rakesh, et al., 2025) (Hu, et al., 2021) LoRA is proven to use less GPU memory, by training only subset of parameters and making fine-tuning feasible on consumer hardware by also ensuring the original model capabilities remain while also adapting to new tasks. The paper supports how GPT-3 175B fine-tuned with LoRA can be reduce by 10,000 parameters and GPU memory requirements by 3 times using LoRA. It provides higher training and inference throughput. It has additionally proven its feasibility of switching task by updating metrics A and B (Russell & Norvig, 2021) (Rakesh, et al., 2025) (Hu, et al., 2021). Thus, proven to be a suitable technique matching our research aim and validating its performance.

Instruction Tuning:
With the probability prediction of text discussed under paper (Liu, et al., 2021) earlier research use to be focused on extracting silent features from the dataset for models to learn from limited data and were sufficient in case of fully supervised learning, following the application of neural network models learned the features during the training phase. However, with the rise in advance models the fully supervised datasets proved to be insufficient and thus the approach of predicting the probability of textual data, with abundance of data available to train the models the models learned general-purpose features in abundance and further can be fine-tuned for specific task. This led to the shift from 'pre-trained, fine-tune' procedure to 'pre-train, prompt, and predict' and instead of object engineering textual prompts formatting were adopted. (Liu, et al., 2021). We adopted the structured prompt engineering strategy explore in (Rakesh, et al., 2025)

Comparative Studies on Models:
As compared under (DeepSeek-AI, 2025) Distilled version of DeepSeek-R1 has proven to be powerful while also leveraging the cold start issue and DeepSeek-R1-Distill-Qwen-1.5B is observed to outperform non reasoning models such as GPT-4o-0513, Claude-3.5-Sonnet, o1-min, DeepSeek-R1-Zero-Qwen-32B across several benchmarks. Specifically for education-oriented knowledge benchmark and long-context-dependent QA task DeepSeek-R1 has even show superior performance compared to DeepSeek-V3, highlighting its capabilities of handing fact-based queries. Thus, supporting our research need for a lighter and open-weight model.

Also, studies by (Rakesh, et al., 2025) by implying LoRA for gender bias mitigation on DeepSeek-R1-Distill-Llama-8B, the model outperform by 80.4 BLEU score, highest of 0.88 cosine similarity and neutralized bias of 91.2% to GPT-3.5, LLaMA 2-7B and Mistral.

As studied under (Zhang, 2024) Tiny Llama also has proven to show competitive performance to other open-source models of similar size as it surpasses OPT-1.3B and Pythia-1.4 and Pythia-1.0B. The model showcases best performance on hellaswag and even when pre-training involved 50% of Chinese data it scored good in English benchmarks too. In case of InstructEval benchmark to evaluate problem solving capabilities, though trained on less code and math corpus it was able to gain ability to improve HumanEval and Drop.

2.4 Summary and Research gap

From previous research we can summarize that there has been different use cases and evolution of approaches to fine tune model, however there is very limited research addressing the problems faced on organization or individual level and the companies still adopt older methods such as RAG, Cosine Similarity or training models from scratch and are deprived of latest research in methods to adopt AI. The open and lightweight models are now capable of proving a leverage to bridge the gap between new findings in the field of research and corporate institutions by using fine-tuning techniques to create models for downstream tasks. Especially evaluating performance of newly released DeepSeek model against other open weight models to support limited resource-based development which has not been tested in past.

CHAPTER 3 – METHODOLOGY

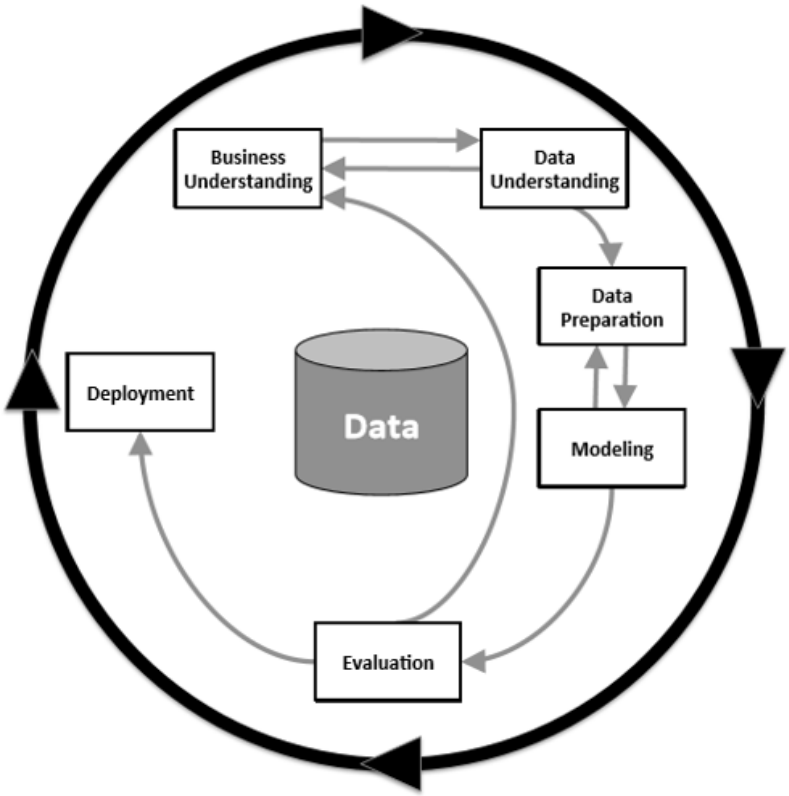


Figure 2: The CRISP-DM process, including the six key phases and the important relationships between them (Wirth and Hipp, 2000, reproduced in Kelleher et al., 2020, p.14)

Above Figure 2 illustrates the CRISP-DM an industry standard framework of end-to-end process involved for successful implementation of data mining projects. It defined the success factors of project implementations. Below are the six key phases involved from Business understand to final deployment phase.

CRISP-DM framework comprises of below six phases:

1. Business Understanding
2. Data Understanding
3. Data Preparation
4. Modelling
5. Evaluation
6. Deployment.

3.1 Business Understanding

Amazon Web Service (AWS) is a cloud platform service provided by Amazon. It enables businesses to benefit from on-demand delivery of compute, storage, application and other IT

resources. There services scale across globe and offer different service packages enabling business to benefit from massive economies of scale and thus provides a knowledge on their portal for customers to understand services offered by them, their integration, pricing etc. It is often difficult to navigate through pages on website and search for any specific use case on a topic and thus chatbot will help users to resolve their queries as per their direction of learnings and querying. Hence, we plan to evaluate if one can design a domain specific chatbot while we try to identify the best open-source model for satisfying this requirement.

3.2 Data Understanding

The data for training the model was collected by web-scraping 108 diverse product related links under Amazon Web Services and originally consisted of 7255 samples as an excel file in the format of 'Source URL', 'Question' and 'Answer' columns. As shown in Figure3 the 'Source URL' field corresponds to the AWS service names and FAQs for respective services are added to 'Question' and 'Answer' column. All columns are of object data types.

Source URL	Question	Answer
https://aws.amazon.com/amazon-mq/faqs/	What will happen when a version of RabbitMQ on Amazon MQ reaches end of support?	Amazon MQ supports RabbitMQ versions up to the end of support dat
https://aws.amazon.com/amazon-mq/faqs/	When I create an Amazon MQ cluster, do the underlying resources (e.g. Amazon EC2 instances)	No, EC2 instances are not visible in your EC2 account as these are man
https://aws.amazon.com/amazon-mq/faqs/	When would I use Amazon MQ vs. managing ActiveMQ, or RabbitMQ, on Amazon EC2 myself?	With Amazon MQ, you do not have to worry about administration task
https://aws.amazon.com/amazon-mq/faqs/	Which versions of ActiveMQ does Amazon MQ support?	Amazon MQ supports ActiveMQ Classic version 5.18 by default. Custo
https://aws.amazon.com/amazon-mq/faqs/	Which versions of RabbitMQ does Amazon MQ support?	Amazon MQ supports RabbitMQ version 3.13 by default. Customers c
https://aws.amazon.com/amazon-mq/faqs/	Who should use Amazon MQ?	Amazon MQ is suitable for enterprise IT professionals, developers, and
https://aws.amazon.com/amazon-mq/faqs/	Why can I not create a ActiveMQ broker on a particular version?	In some cases, Amazon MQ may end support for specific versions with
https://aws.amazon.com/amazon-mq/faqs/	Why can I not create a RabbitMQ broker on a particular version?	You can manually upgrade your broker at any time to the next support
https://aws.amazon.com/amplify/faqs/	Q: Are prices different per region?	Prices are the same across all regions.

Figure 3: Dataset Samples

It is important to avoid duplicate data to avoid training instability and prevent overlapping in training and evaluation set. On removal of duplicates, we had 7230 unique samples with no missing or null values. This FAQ based data format was selected as it is compatible for training instruction tuned chatbot models. By reviving the content of each column, we need to identify which if any data cleaning steps are needed. One must also check for skewness in tokens for questions and answers.

Exploratory Data Analysis (EDA)

The EDA on data helps in understanding the characteristics, quality and identify issues that needs to be handled to avoid any blind spots in data before training the model. The details can be observed in numerical or graphical formats. Followed by knowing the insights one must take steps to clean or process the data to make it suitable for the training model as the requirements for training data differs across LLM models and training goals.

EDA helped in identifying the spread of words and token for question-and-answer block and get below insights on training data:

Word Count Analysis:

- Word count analysis provides human readable measure of verbosity on both answer and question columns whereas token count gives us more language model-based perspective.
- Figure 4 below provide insights on word count perspective as we identified that the longest question is of 105 words and the shortest is of 1 word indicating that not all

Question sections are direct questions but are mapping of source URL i.e. AWS service and its available knowledge base.

- Most questions have between 7 and 13 words with an average of 10 words.
- The shortest answers are of single word as seen below can be a direct 'yes' or 'no' as an answer.
- The longest answer is of 873 words and the shortest is of 1 word
- Most of the answer's length are comparatively longer than that of questions.
- Most answers are of words in range of 29 to 81 words.

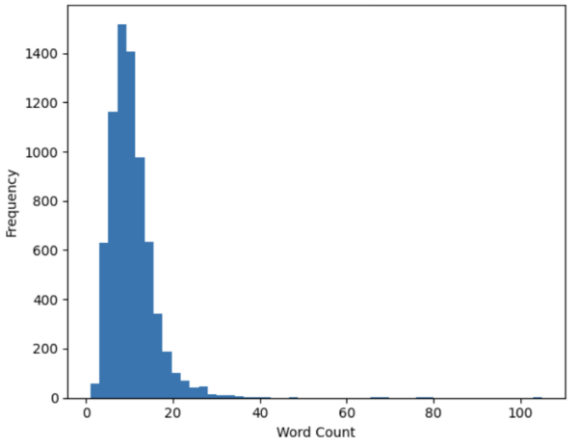


Figure 4 (a): Question Word Count Distribution

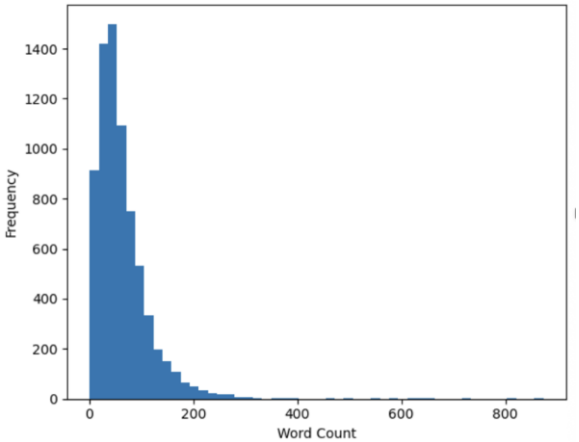


Figure 4 (b): Answer Word Count Distribution

Token Count Analysis:

- Below Figure 5 indicates that most question tokens ranges between 10-16 tokens and very few question exceed 40 tokens.
- Also, most answer are in between 29 to 81 word counts with at least 3 words to max 1750 tokens in an answer.

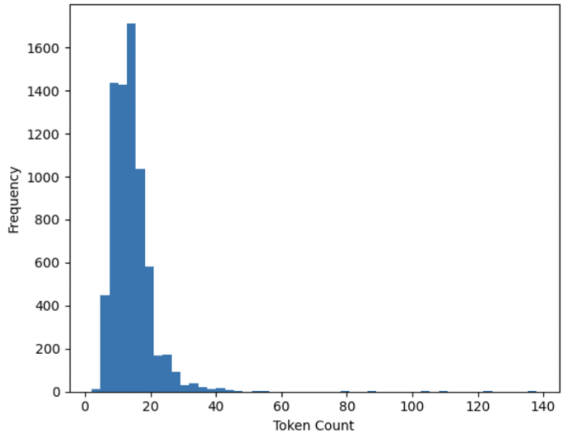


Figure 5(a): Question Token Count Distribution

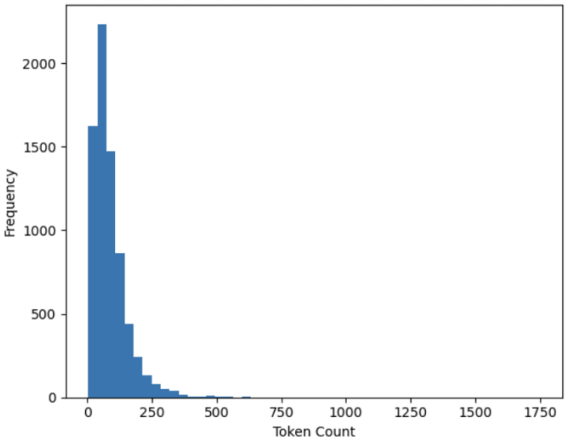


Figure 5 (b): Answer Token Count Distribution

- The graphs are right skewed indicating few questions and answers do tend to be longer than others.

- This analysis shall help in defining the limits for min and max length for optimizing token generation while inference.

Word cloud:

- The word cloud shows the most frequently appeared words in database based on their size and boldness.
- Before preprocessing and cleaning of data word cloud reflected single letter such as ‘Q’ and ‘A’ appearing as prefix for most questions thus we removed noise from data and standardized the format.
- On data cleaning the word cloud in Figure 6 highlights that most of the questions are related to usage, data, applications and support related queries. Hence, answers are also subjected to usage, application, data and specific to AWS Amazon services.

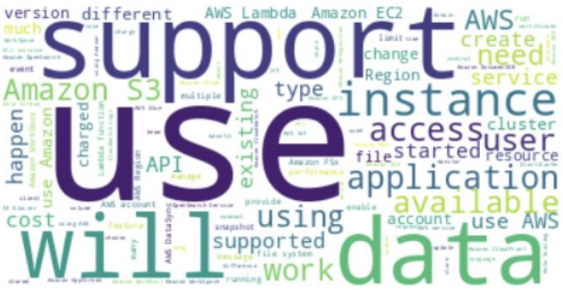


Figure 6 (a): Common Terms in Questions

Figure 6 (b): Common Terms in Answer

Other findings:

- Below bar chart in Figure 7 highlights top 20 services with highest number of questions. EC2 service has highest number of questions i.e. more than 500 questions in dataset, followed by S3, EMR and thus can highlight that they are the most widely queried and used services across platform.

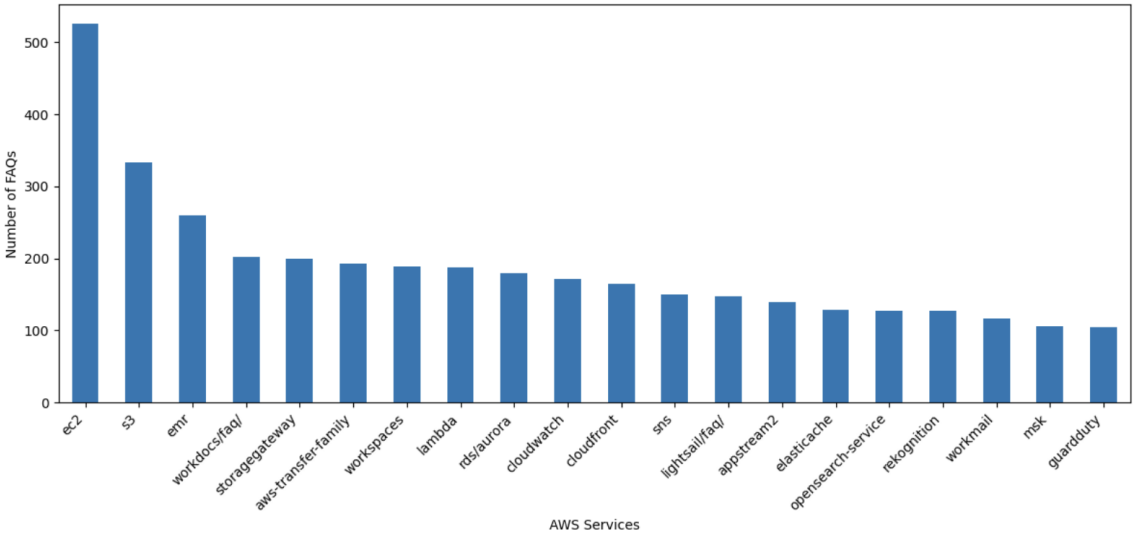


Figure 7: Top 20 FAQ Count by AWS Service

- Below Figure 8 indicates that most questions start with ‘what’ to an extend indicating that they are based on understanding different services.
- Questions starting from ‘how’, ’can’, ’does’ are the next most frequent questions in the dataset.
- Very few questions are observed to start with ‘could’, ’should’ or ‘who’.

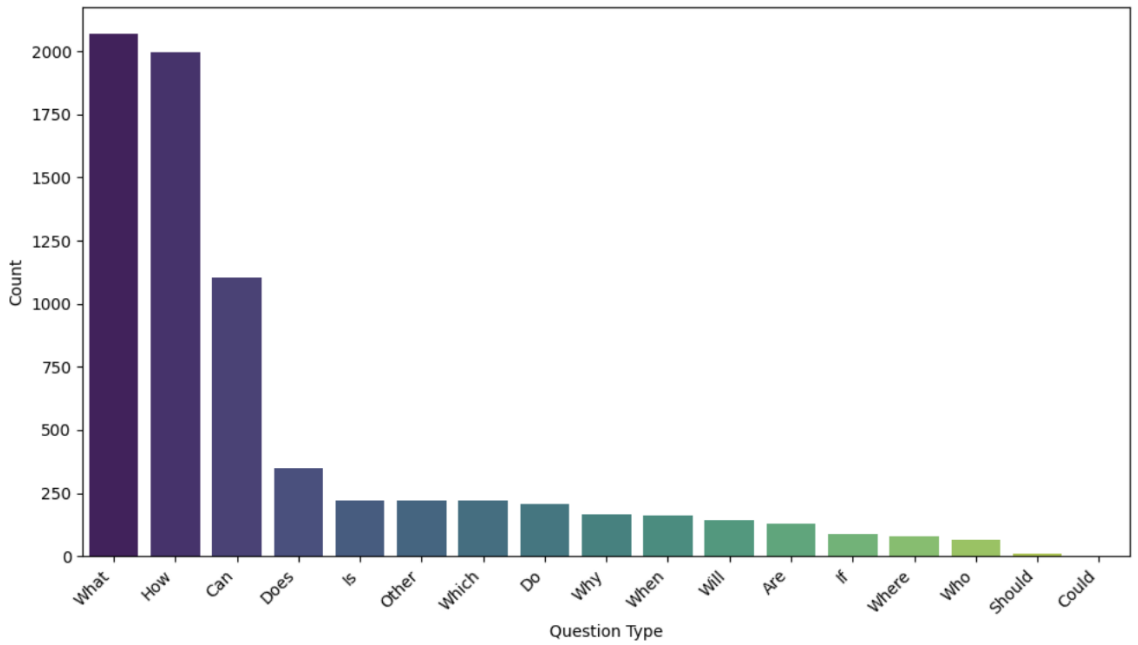


Figure 8: Frequency of Question Types

3.3 Data Preparation

This stage includes cleaning and formatting the data as per model architecture. While web scraping the data it was downloaded in FAQ format with Questions and respective answers. Typical data preparation task includes application of techniques such as tokenization, stop words, lemmatization etc., however in case of training transformer-based models one has to follow different approach since the models understand raw text and has inbuild BBPE tokenizer and also being already being trained on massive corpus with embedding. However, under data preparations stages we have taken steps to ensure to remove html tags or symbols, duplicate rows and ensure no user specific details included in training data.

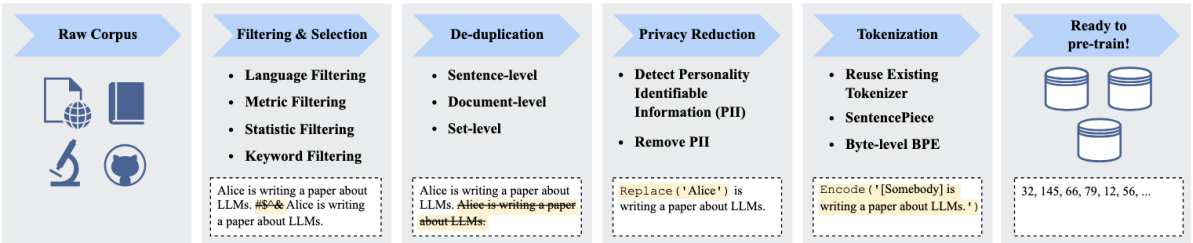


Figure 9: An illustration of a typical data preprocessing pipeline for pre-training large language models

Existing work [234] has found that duplicate data in a corpus would reduce the diversity of language models, which may cause the training process to become un- stable and thus affect the model performance. Therefore, it is necessary to de-duplicate the pre-training corpus (Zhao, et al., 2025)

Transformer models follow different architectures than classic models and thus while training the same one must ensure the training data is aligned with model. Thus, prompt formatting was done using Alpaca-style format, including:

- “Instruction”: “Answer the following AWS FAQ question”
- "input": "What search features does Amazon CloudSearch provide?"
- "output": "Amazon CloudSearch provides features to index and search both structured data..."

This format was adopted as it allowed training for both instruction and non-instruction tuned models. As discussed in paper (Rakesh, et al., 2025) it is often recommended to split data into training (80%), validation (10%) and testing (10%). After training the model on training data., validation data is used for intermediate testing and the test set is used to for final testing of the model (Inouye, et al., 2024). To ensure train, test and eval set had at least a single sample of each AWS service one had to ensure we have sufficient questions for each service this was done using stratify split. Thus, services with less than or equal to 3 questions were dropped and total 6 questions were dropped form the dataset. This prompt formatted data was split into 80:20 ratio in which 80 is the ratio of training set. To further split 20 was into test and eval we had to ensure at least 2 samples of each service existed in dataset thus based on filtering the samples were divide to 50 test and 50 evaluation samples.

The evaluation dataset will be used to measure validation loss during training phase and test dataset to test model performance after training. The samples were downloaded in jsonl format to maintain same sets of data for evaluating of all the models.

3.4 Modelling

3.4.1 Overview of Large Language Models (LLMs) and Transformer Architecture

LLMs are mainly build on Transformer based architecture which can be referred to as pre-training bidirectional (captures context from past and future input tokens) language models (Naveed, et al., 2024) (Zhao, et al., 2025). With LLM such as GPT the models have transitioned from simple text generation to complex task solving (Zhao, et al., 2025). LLMs with transformer-based architecture (Figure 10) supports parallelization and capacity, and a possibility to scale models to billions of parameters (Zhao, et al., 2025), simultaneously increasing computational and data. The transformer model was first introduced in 2017 by google which was later adapted by OpenAI to release first Generative Pre-Training GPT-1 model (Zhao, et al., 2025). The Transformer architecture has different variation based on the application of attention and transformer blocks connections (Naveed, et al., 2024). The best models connect encoder and decoder through an attention mechanism (Vaswani, et al., 2023).

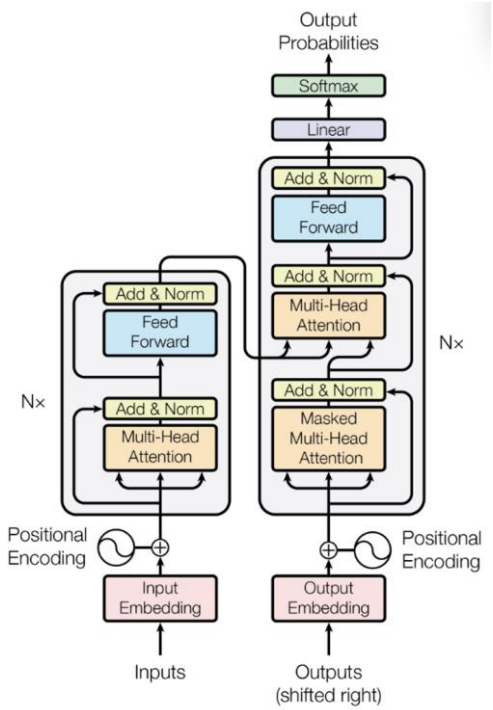


Figure 10: The Transformer – model architecture (Vaswani, et al., 2023)

Few models' architecture includes both Encoder for processing inputs and Decoder for generating output. Causal Decoder architecture (Figure 11) uses only decoder for both tasks, depending on the past information whereas Prefix Decoder is a Non-Causal Decoder where attention is bidirectional. Mixture-of-Experts (MOE) based transformer architecture includes feed-forward layers after attention blocks (Naveed, et al., 2024).

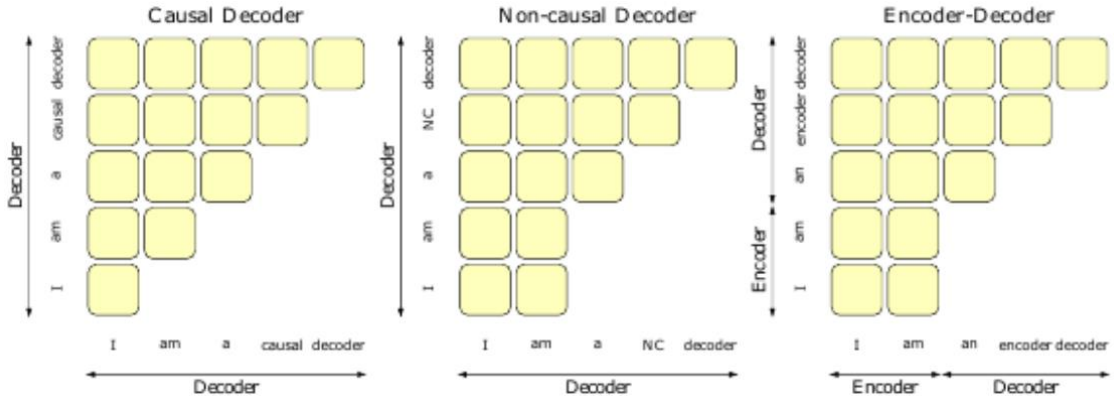


Figure 11: An example of attention patterns in language models (Naveed, et al., 2024)

Attention mechanism was also a part of sequence modelling and in conjunction with recurrent network (Vaswani, et al., 2023) where attention was applied from target to source RNN and self-attention that extends to hidden states (Russell & Norvig, 2021). As shown in Figure 12 attention is a mapping a query and a set of key-value pairs to an output which is calculated as a weighted sum in forms of vectors. Here the weight for a value is a compatibility function of query and corresponding key (Vaswani, et al., 2023) where

Q(query) represents the current token, K(key) represents feature information of all tokens that query can be matched against.

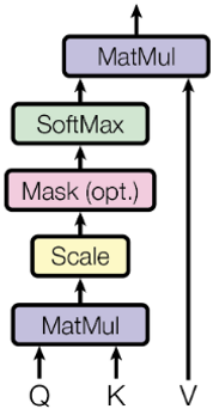


Figure 12(a): Scaled Dot-Product Attention.

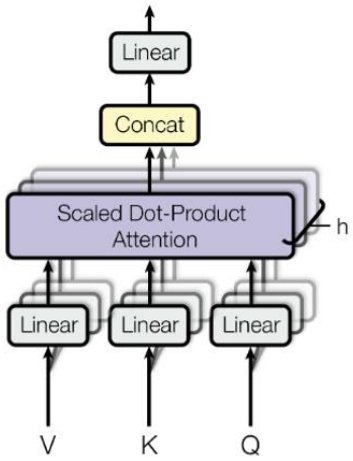


Figure 12(b): Multi-Head Attention (Vaswani, et al., 2023)

Most of the models such as GPT-1, LLaMA follow decoder-only transformer architecture.

3.4.2 Model Architecture:

As highlighted under (Hoffmann, 2022) the model size and training data must be proportionate to train a compute-optimal model leading us to explore the potential of training small models helping us to focus on optimal performance at an acceptable inference cost (Zhang, 2024) thus we choose below model and discuss their architecture.

DeepSeek (DeepSeek-R1-Distill-Qwen-1.5B) vs GPT:

The DeepSeek model has led to overcome the limitation that existed even with GPT and outperform other open-source models while balancing model size, efficiency and performance.

The design of GPT and DeepSeek outshines other models in terms of performance and hence their outreach to the global population. Many benchmarks showcase GPT outperforming competitors in terms of engagement and adaptability to problem solving (Smiju & Adinath, 2025) whereas DeepSeek is best suited for precision based complex tasks, prioritizing scalability, cost-effectivity, coding (DeepSeek-AI, 2025) and its acceptance on a larger scale. GPT is a generative pre-trained transformer based deep learning model, which means it can handle huge amounts of text (Smiju & Adinath, 2025) and image data and generate contextual responses, DeepSeek can also handle image data however the same has not been explicitly published.

In reference to below Table 1 we can say that both models perform well in terms of complex scientific tasks however the main distinguishing factors are Transparency and Cost. DeepSeek is best suited for precision-based tasks, prioritizing scalability, cost-effectivity, coding (DeepSeek-AI, 2025) and its acceptance on a larger scale.

Category	DeepSeek	ChatGPT
Similarities		
Performance	Excels in clinical data interpretation, manuscript drafting and complex scientific tasks.	Matches DeepSeek in clinical data interpretation and manuscript drafting capabilities.
Limitations	Struggles with basic tasks (e.g., counting U.S. states with 'W') and occasional hallucinations.	Shares similar limitations, including errors in simple tasks and hallucinations in niche domains.
Differences		
Transparency	Open-weight architecture allows auditing, modification and retraining for niche applications.	Proprietary 'black box' system restricts access to underlying algorithms.
Cost and accessibility	Free-tier access (via DeepThink) and self-hosting options democratize AI for smaller institutions.	Paywall barriers limit accessibility for resource-constrained users.
Cultural nuance	Inconsistent handling of politically sensitive topics in China [35]; unclear if model or interface-driven.	No documented region-specific constraints in published literature.

Note: Hallucinations: Both models may generate plausible but incorrect outputs in specialized domains. Self-hosting: DeepSeek's offline deployment enhances data privacy, while ChatGPT relies on cloud-based access. Ethical Scrutiny: DeepSeek's openness invites community-driven improvements but raises risks of misuse. Abbreviation: AI, artificial intelligence.

Table 1: Similarities and differences between DeepSeek and ChatGPT (Kayapalı, et al., 2025)

Newer versions of these models are continuously being released with improvements in reasoning, factual accuracy and contextual understanding, pushing the integration boundaries of human intelligence with Artificial Intelligence (Smiju & Adinath, 2025). To compare the published and most recent model versions, as of March 2025 GPT-3 is trained on 175 billion parameters (Brown, et al., 2020), whereas DeepSeek-V3 is trained on 671 billion parameters (DeepSeek-AI, 2025).

Overall GPT model can sometimes not understand the specifics and can misalign the contexts, specifically in coding, data analysis where context is the key to execution (Smiju & Adinath, 2025). DeepSeek is more popular in the context of efficient computation, accuracy and structured resolution to problems and being more task specific thus powerful in technical concepts like proving theorems, providing more detailed outputs (Smiju & Adinath, 2025) whereas GPT gives more importance to natural conversational understanding, and supports interactions like a tutor (Kotsis, 2025). This power of DeepSeek comes with a tradeoff of flexibility in conversations, due to strict content moderation preventing its acceptance in more contextual or variation responses. The main design factor in DeepSeek of only activating desired model components has reduced overhead computation (Smiju & Adinath, 2025).

The DeepSeek model focuses on bridging the gap between consistently striving for a strong model performance and its economic inference costs. This was possible by adopting a large Mixture-of-Experts (MoE) model with 671B parameters where only a portion of 37B parameters is activated for each token. Thus, pushing the limits of open-source models and making it accessible for smaller organizations to operate at a larger scale. Due to this possibility of flexible parameters and reduction in model size, different versions of model based on its training parameters are now available for its local deployment, as compared to other models which are accessible only via API, charging per token and hence does not support training models locally (DeepSeek-AI, 2025).

The development was focused on how a new model could meet the existing benchmarks by re-designing the algorithm, framework and hardware while significantly enhancing training

efficiency and reducing training cost. It also aimed to overcome the drawbacks in existing models which have adopted Mixture-of-Experts (MoE) models (DeepSeek-AI, 2025).

Multiple approaches were developed during the training and testing phase of the model. They implemented Multi-Token Prediction (MTP) where instead of predicting just a single token it predicts the next two tokens. FP8 mixed precision training was used to represent numbers in 8 bits and the system would dynamically select which tensors should be in FP8 or FP16/FP32 and this was validated for the first time on large-scale models. Mostly critical parameters are kept at higher precision (FP16/FP32) to avoid loss. Instead of using a single large model(expert) a group of smaller sub-models are trained, each learning specific patterns, this process Is called Mixture-of-Experts (MoE). Compared to the traditional approach where each expert covers multiple tasks, they do not share information because of which some experts may be underutilized leading to higher compute. DeepSeek MoE used finer-grained experts, more specialized and narrower tasks, experts are shared across tasks healing in balancing workloads and more efficient routing. Instead of having all independent experts, it isolates some experts as shared ones for general purpose knowledge and rest as specialized for specific tasks. Team adopted 671B total parameters, and activated only 37B per token, 8 experts to be activated per token in a given input. They designed a new Dual Pipeline algorithm which overlaps the two computation and communication phases across both forward and backward processes (DeepSeek-AI, 2025).

The model has achieved state-of-the-art results in multiple benchmarks. Specially outperforming with huge differences on multilingual, code, and math benchmarks. The MTP enhanced the text fluency, coherence and context understanding where the second token prediction range was between 85-90%. The standard deviation of 0.006 is very low and the best amongst the other large-scale models, providing stable performance and lesser risk of overfitting as compared to GPT which has a value of 0.01-0.05. FP8 mixed precision training framework led to faster calculation than FP16/FP32, accelerating training and reduced GPU memory usage. MoE has helped in reducing memory bottleneck by activating only specific experts for specific inputs. It proved to be a cost-effective training. Also designing Dual pipeline algorithms for pipeline parallelism has improved communication and reduced bottlenecks by addressing the challenge of heavy communication overhead due to cross-node expert parallelism while also developing kernels facilitating the same (DeepSeek-AI, 2025).

With rapid evolution on LLMs the industry has been trying to dimmish gaps towards Artificial General intelligence (AGI). With DeepSeek-R1 fine-tuned on DeepSeek-V3-Base has incorporated cold-start data and multi-stage training pipeline to an extent (DeepSeek-AI, 2025). As our research is more focused on training models on individual or small businesses level on a limited data, DeepSeek-R1 completely supports the research idea. We are using an opensource DeepSeek-R1-Distill-Qwen-1.5B as a primary model of interest which is smaller dense model and takes distillation from of DeepSeek-R1 using Qwen2.5-32B as base model. The application of RL directly to base model without supervised fine-tuning (SFT) has helped model explore chain-of-thoughts (CoT) essential for solving complex problems but the distilled models include excludes application of RL due to the enormous computational power and only has SFT to demonstrate effectiveness of distillation. It demonstrates that reasoning patterns of large models can be distilled into

smaller models and even outperforms application of RL for the same case (DeepSeek-AI, 2025).

Model evaluated in research (Li & Sarwate, 2025) DeepSeek-R1-Distill-Qwen-1.5B and LLama-3.2-1B are also good considerations to balance performance with computation resource feasibility.

Tinyllama:

TinyLlama-1.1B-intermediate-step-1431k-3T is a 1.1B parameter trained with around 1 trillion tokens for 3 epochs and is built on tokenizer of Llama2 and supports achieve computational efficiency and yet remarkable performance on downstream task. Similar to GPT and DeepSeek TinyLlama is also a transformer decoder only model using Rotary Positional Embedding (RoPE) as positional embeddings to insert positional information into model, executing pre-norm instead of post-norm to stabilize training efficiency, fused SwiGLU instead of ReLU activation function and grouped-query attention to reduce memory overhead and improve inference speed. By integrating codebase with FSDP they leveraged multi-GPU and multi-node efficiency for training (Zhang, 2024). TinyLlama-The chat based TinyLlama-1.1B-Chat-v1.0 used in comparison is finetuned on this base model.

3.4.3 Fine-Tuning Techniques

Low-Rank Adaptation (LoRA):

Training an LLM involves researchers and engineers solving critical engineering issues with data preprocessing and parallel training processes (Zhao, et al., 2025) and thus being a very resource and time intensive process because of which LLMs are generally published by large and well-funded organizations.

The models are published as a pre-trained model, trained on a large corpus capable of predicting next tokens as per input query. Fine-tuning LLMs for a downstream task has proven to improve performance of models including approaches such as enhancing architectures and training strategies, but fine-tuning LLMs consisting of billions or parameters are computationally intensive and time consuming (Naveed, et al., 2024). To avoid training mode from scratch we focus on improving model's domain understanding of the AWS Cloud platform using Transfer Learning a Fine-Tuning method. To improve model efficiency in answering users query and reduce compute requirements we adopted LoRA based Parameter-efficient fine-tuning (PEFT) as in comparison to FFT, PEFT is proven to perform better in case of low-resources, comparable performance on medium and worst under high-resources (Naveed, et al., 2024).

(Russell & Norvig, 2021) (Rakesh, et al., 2025) (Hu, et al., 2021) Tested Low-Rank Adaptation (LoRA) which is a Parameter-efficient fine-tuning (PEFT) is a model fine tuning technique which modifies only a subset of model's parameter and can reduce the computational cost and models adaptability significantly by reducing the number of re-training parameters while retaining efficiency.

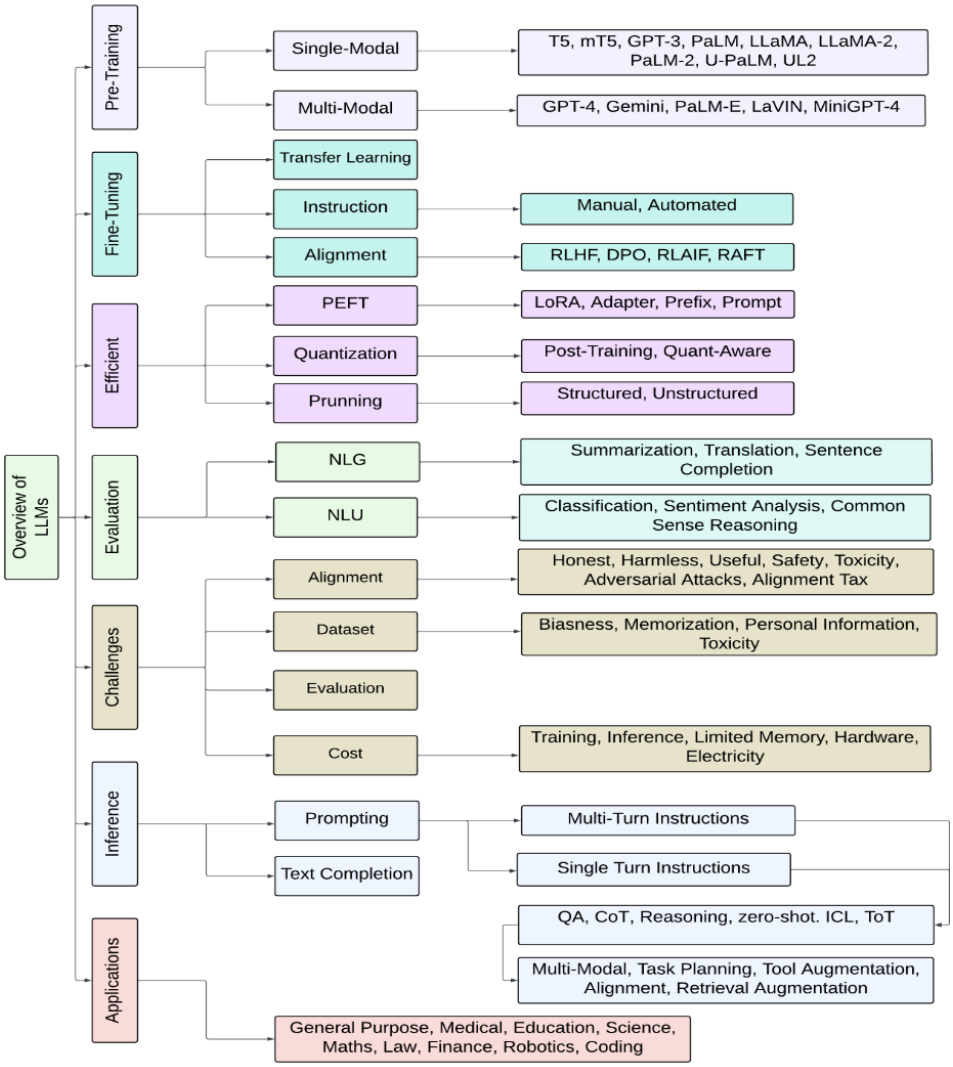


Figure 13: A broader overview of LLMs, dividing LLMs into seven branches: 1. Pre-Training 2. Fine-Tuning 3. Efficient 4. Inference 5. Evaluation 6. Applications 7. Challenges (Naveed, et al., 2024)

- It introduces low-rank trainable metrics into specific layer of model to capture the fine-tuning updates
- Freezes the pre-trained model weights while updating only specific sub-components.
- With reduction in parameters, we only need to store and load only task-specific parameters thus reduces memory overhead, enabling faster training without sacrificing performance.

It updates the transformer layer using an additional low-rank decomposition matrix. Considering ‘W’ as a pre-trained model matrix, LoRA introduces ‘ΔW’ as learnable weight update and generates the new weight matrix as below:

$$W' = W + \Delta W$$

The ΔW is not learned directly but decomposed into two smaller metrics ‘A’ (low-rank matrix of size ‘d x r’), ‘B’ (low-rank matrix for size ‘r x k’) and r is the rank of adaptation supposedly much smaller than full weight dimension as below:

$$\Delta W = A * B$$

Lora is often applied to key layer of transformer and specifically in self-attention mechanism which include:

- Query Projection (q proj)
- Key Projection (k proj)
- Value Projection (v proj)
- Output Projection (o proj)

The new trainable parameters are defined by the rank ‘r’ and the shape of original weights (Hu, et al., 2021).

Instruction tuning with Alpaca format:

Since the dataset is in FAQ format and from below Figure 14 we understand that DeepSeek being an instruction-tuned model (Naveed, et al., 2024) in order to improve the response to user queries, train model on limited hardware the model needs to be parameter-efficient fine-tuned (PEFT) DeepSeek-R1-Distill-Qwen-1.5B on Instruction formatted data which consist of instruction and input-output pairs. Instead of using Alpaca model directly we replicated its instruction tuning schema (instruction, input, output) as referred under ((CRFM), 2023) by applying manual prompt formatting by creating instruction-based dataset i.e. instruction and an input-output pair.

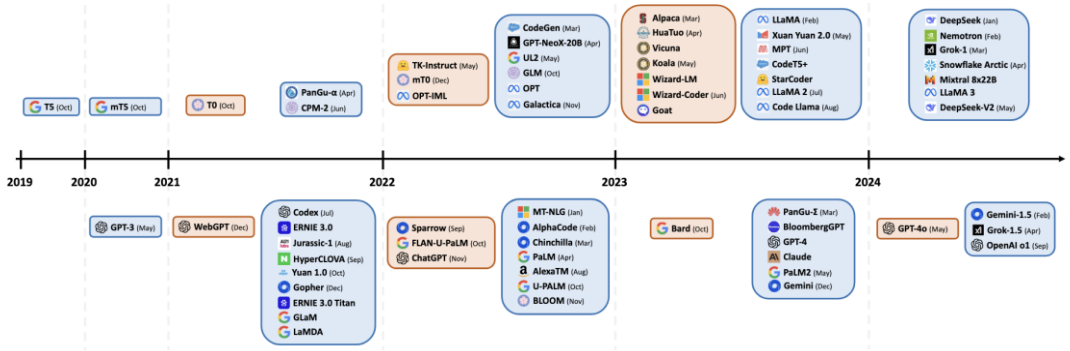


Figure 14: Chronological display of LLM releases: blue cards represent ‘pre-trained’ models, while orange cards correspond to ‘instruction-tuned’ models. Models on the upper half signify open-source availability, whereas those on the bottom are closed-source. The chart illustrates the increasing trend towards instruction-tuned and open-source models, highlighting the evolving landscape and trends in natural language processing research (Naveed, et al., 2024)

Under the research of paper (DeepSeek-AI, 2025) it is observed that DeepSeek-R1 is sensitive to prompts and few-shot prompting degrades the performance hence training this model require description of problem and output format such as Alpaca format using zero-shot for better performance (DeepSeek-AI, 2025). Considering the instruction tuning dataset can be generated manually or automatically we refer to use manual approach.

As studied by (Liu, et al., 2021) in contrast to supervised learning which for an input ‘x’ predicts output ‘y’as P (y|x), prompt-based learning models the probability of text directly. This is done by modifying the original input ‘x’ using template into text string prompt x′, with some unfilled slot the model tends to fill the slot with probabilistically and obtain final string x̂ from which final y is derived (Liu, et al., 2021).

Matching the format used in (Rakesh, et al., 2025) we adopted the format as below and extending output section by combining Source URL column to provide reference link for the customers.

{"instruction": "Answer the following AWS FAQ question", "input": "Does Amazon WorkMail support public folders?", "output": "No, WorkMail does not offer public folders. [Source: https://aws.amazon.com/workmail/faqs/]"}

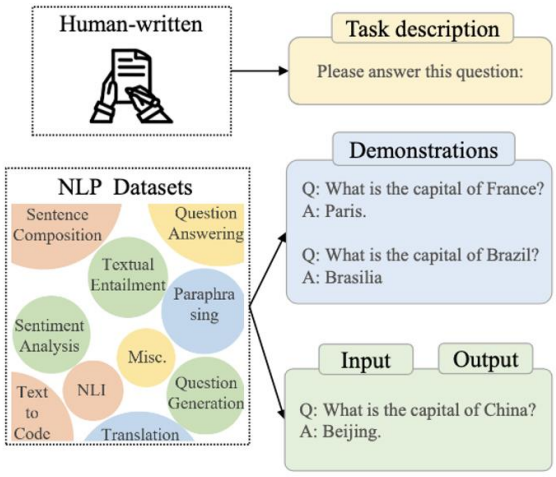


Figure 15: An illustration of instance formatting for constructing the instruction-formatted instances (Zhao, et al., 2025).

Where instruction defines the description of task, input corresponds to Question column in dataset while training the model or user’s question in inference stage and the output corresponds to ‘Answer’ column in training phase or answer to user’s query in inference stage.

Experiment Setup:

The model was trained on Google Colab platform considering easy access and scope to scale in term of GPU resources in case needed. The training failed with CUDA out-of-memory errors on free tier CPU, T4 GPU or V2-8 TPU thus proceeded with purchase of google pro compute and trained models on L4 GPU which provides 1 Nvidia L4 GPU unit with 53 GB Systems RAM, ~22.5GB GPU RAM and 235.7 GB disk space.

max_new_tokens: While tuning max_new_tokens token as supported by (Li, et al., 2025) to balance between reasoning and security (safety score). the output length is observed to impact the reasoning capability of model. With longer output responses model tends to longer thinking however in case of forced thinking and long token outputs the thinking token themselves shortens. Thus, one must limit the output token for robustness in response (Li, et al., 2025). We tested different answers responses generated for

max_new_tokens values such as 512, 200 and 100 the value 100 was able to generating concise and avoid force thinking to match the training dataset requirements.

LoRA Rank: This parameter affects the trainable weights. Higher the rank more the number of trainable weight and model being more specific to domain and with lower rank model learns to generalize well (Inouye, et al., 2024). During hyperparameter tuning we observed the

LoRA Alpha: Defines the scaling factor and to what extend the original weights changes. With lower value the impact of weights decreases

Learning Rate: Determines the speed of weight changes towards optimal value. Higher value may lead to missing or overshoot the optimal value as it never stabilizes and a low learning rate may lead to higher computation.

In case of overfitting LoRA dropout can be hyper tuned which drops a set of random weights lowering training loss. It is often recommended to use power of 2 such as (8,32), (16,8) and (64, 4) and learning rate between 0.00001 and 0.01which were found to be best in cases. Also, it suggests the rank and alpha values of higher digit 128 or 256 may lead to higher training time and loss and that lower rank with decrease in parameters often corresponds to reducing training time as compared to higher rank(Inouye, et al., 2024).

3.5 Evaluation

To evaluate model's interims of resource utilization we are using training time, GPU and RAM utilization on Google Colab. The training time is calculated using simple python code in training process.

- Training time was measured using Python timestamps around the training loop.
- We measure the peak tensor memory allocated for training (for model weights, activations and gradients) using PyTorch's CUDA memory tracking utilities.
- The CUDA overhead (caching and runtime) is recorded from Google Colab resource section as GPU RAM.

To evaluate quality of generated responses we used below benchmarks suggested under (Zhao, et al., 2025) (Gillgren, 2024) which compares generated text against validation dataset.

- **BLEU (Bilingual Evaluation Understudy):** Measures precision of n-gram in generated text to the reference text, reflecting accuracy of word choices being correct and relevant while penalising wrong words in generated answer.
- **ROUGE (Recall-Oriented Understudy for Gisting Evaluation):** Calculates degree of overlapping between generated and reference text by comparing the overlapping n-grams, word sequences and word pairs in order to primarily measure recall, penalising missed words from reference text in generated text. ROUGE-1, ROUGE-2 and ROUGE-L measures the unigram, bigram and longest subsequence.

- **SBERT Semantic Similarity:** Compares the sentence level overlap using sentence embeddings from Sentence-BERT model, computing cosine similarity to reflect semantic alignment between generated and reference answers.

ROUGE and BLEU emphasize on lexical overlap (intrinsic hallucination and narrative fidelity) while SBERT captures semantic alignment.

3.6 Deployment

In this phase of CRISP-DM framework engineers tend to deploy the trained model on available resources locally or on cloud. Based on the results observed below we can conclude that the minimum compute needed to deploy of of the tested model is of ….. GPU which exceeds the limit of current local environment or long term pay-as-you-service on an individual level.
Due to this GPU constraints gave insufficient memory on hugging Face Space under free tier. However, the trained model can be packages as container for deployment on platforms providing an API endpoint or URL to be accessible over DNS.

Few of the use cases of this model include:

- Customer Support AWS Chatbot: To ensure customer can query on bot to learn more about AWS service offering.
- This shall reduce dependencies on customer support agent and help with efficient human resource management on business level.

Engineering learning:

- By providing instant answers to user queries the model can resolve user issues on the go and in case of no resolution can be further diverted to agent if required.
- It also supports onboarding of new employees in a company to troubleshot while learning using chatbot trained on updated knowledge base.

CHAPTER 4 – RESULTS AND DISCUSSION

Results:

- Before tuning models with LoRA we tested the performance of DeepSeek-R1-Distill-Qwen-1.5B model in order to evaluate its pre-training knowledge on AWS cloud platform.

Methology	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	Semantic Similarity
Base model(no training)	0.028	0.2114	0.0499	0.1425	0.582
Training without LoRA	CUDA out of memory error				

Table 2: DeepSeek-R1-Distill-Qwen-1.5B base model performance scores

- The training model with stratify is capable of returning correct context, domain specific answers which are easier for user interpretation, formatted to precision which is crucial for a domain specific assistant.
- For DeepSeek-R1-Distill-Qwen-1.5B we tested application of LoRA on different layers with num_train_epochs=0.5, r=4, lora_alpha=16 and learning rate = 2e-4 and observed below results:

LoA application on number of layers	All Parameters	Training Parameters	Training time in min	Training GPU in GB	Collab RAM	Val loss
4 layers	1778177536	1089536	31	12	20	2.332504
7 layers	1781704192	4616192	43	14	18.2	1.624013

Table 3(a): DeepSeek-R1-Distill-Qwen-1.5B LoRA application on different

LoA application on number of layers	Scores
4 layers	BLEU: 0.030310835868820184
	ROUGE: {'rouge1':0.250255, 'rouge2': 0.06646, 'rougeL': 0.175809, 'rougeLsum': 0.161674
	SBERT Semantic Similarity (avg): 0.662667325404888
7 layers	BLEU: 0.018265778121403228
	ROUGE: {'rouge1':0.24037, 'rouge2': 0.064044, 'rougeL': 0.1718500, 'rougeLsum': 0.172345
	SBERT Semantic Similarity (avg): 0.6485593308445433

Table 3(b): Scores for DeepSeek-R1-Distill-Qwen-1.5B LoRA application on different layers

- Models were evaluated on identical setup of num_train_epochs=0.5, r=4, lora_alpha=16 and learning rate=2e-4 except for GPT with 0.25 epochs, half the epochs for other models.

Model	Training Paras	Total Paras	Training epo	Training Time	Training GPU	Total GPU
DeepSeek-R1-Distill-Qwen-1.5B	1089536	1778177536	0.5	31	12	20
TinyLlama-1.1B-Chat-v1.0	1126400	1101174784	0.5	23	6	15
GPT-Neo-1.3B	1179648	1316755456	0.25	23	13	20

Table 4(a): Comparative scores of models across different benchmarks.

Model	BLEU	ROUGE-1	ROUGE-2	SBERT Semar	Validation Loss	Eval Loss
DeepSeek-R1-Distill-Qwen-1.5B	0.0303	0.25025	0.0664	0.6626	2.3245	2.3245
TinyLlama-1.1B-Chat-v1.0	0.0115	0.26008	0.0902	0.7069	1.3291	1.2706
GPT-Neo-1.3B	0.0118	0.19991	0.0676	0.6094	1.6228	1.5347

Table 4(b): Comparative scores of models across different benchmarks.

- The training and validation loss for all models consistently decreased, providing effective learning.

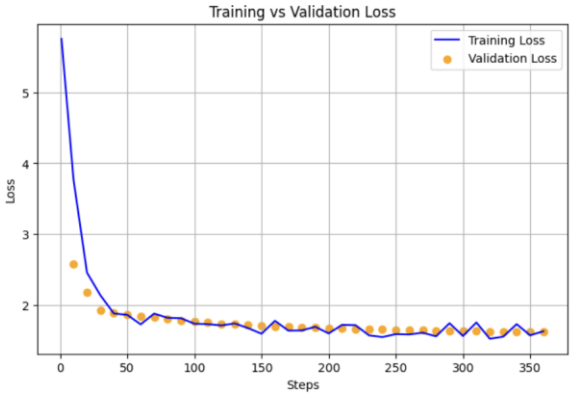


Figure 16(a) : Loss graph for Deepseek Lora 7layer

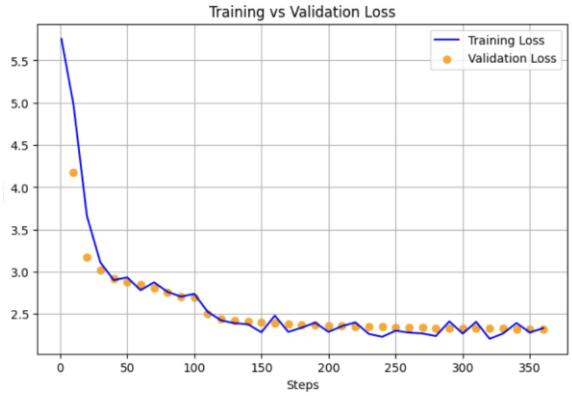


Figure 16(b): Loss graph for Deepseek Lora 4layer

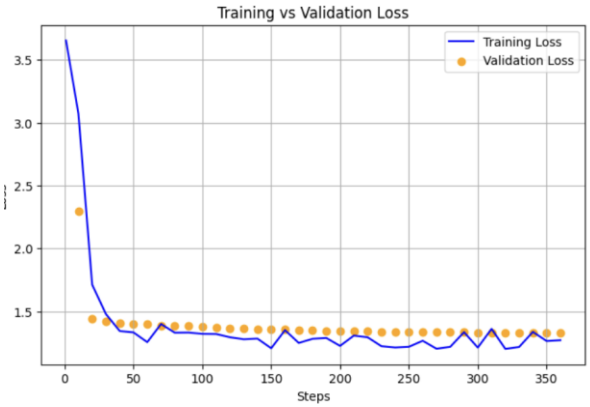


Figure 16(c): Loss graph for TinyLlama Lora 4layer

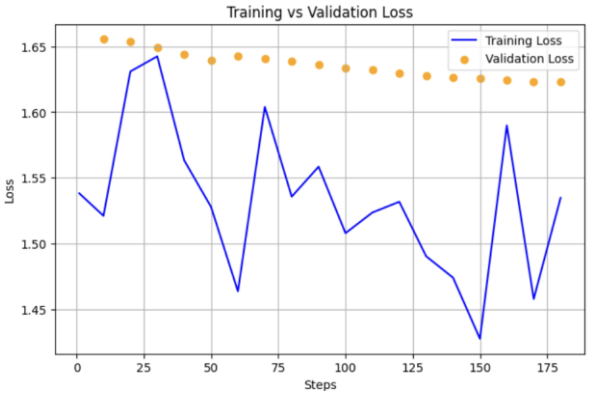


Figure 16(d): Loss graph for GPT-Neo Lora 4layer

Discussion:

- The limited domain understanding of DeepSeek-R1-Distill-Qwen-1.5B model highlighted the need to train model on domain specific database.
- The CUDA out of memory error while fine-tuning DeepSeek model highlighted the needs for LoRA and minimum resource requirement.

- In case of training model without stratify, the model has failed to answer for samples excluded in training set and generated a more generic answer, highlighting the domain knowledge of base model.
- On manual evaluation of generated answers, though the scores across benchmark metrics differs moderately, it is observed to affect the model generation capabilities significantly.
- Below are the observations on comparison of DeepSeek-R1-Distill-Qwen-1.5B, GPT-Neo-1.3B and TinyLlama-1.1B-Chat-v1.0 models.
  - LoRA 4-layer vs 7-layer application:
    - Table 3(a) highlighted that the trainable parameters increase with application of LoRA on more layers.
    - 7-layer LoRA indicating ~4.24 of more trainable parameter than 4-layer LoRA.
    - Higher number of parameters lead to more training time and training GPU while improving the model performance.
    - An increase in trainable parameters leads to storing more weights, gradients and optimizer states, thus increase GPU memory as well as the overhead on host memory.
  - Training Time and GPU Memory (refer Table4(a)):
    - DeepSeek-R1-Distill-Qwen demonstrated a slightly more training times by 7 minutes compared to other models considering that the overall model still has highest number of total parameters and a larger network.
    - On evaluating exported answers of models DeepSeek have generated superior quality answers with the minimum trade-off of training time.
    - Tiny Llama proved to be most lightweight model utilizing least GPU and time however compromising with quality of answers to greater extend.
    - GPT was observed to take nearly 3 hours to train over 0.5 epochs. Thus, to maintain fairness in comparison and training time across models we GPT was evaluated over 0.25 epochs. The generated answers are not superior and efficient to DeepSeek but were acceptable to Tiny Llama.
  - Scoring benchmarks (refer Table4(b)):
    - DeepSeek scored highest SBERT Score indicating best precision with lowest of incorrect words in generated answer and a significant difference with other models.
    - The SBERT score of all models are generally inline
    - Tiny Llama scored lowest in BLEU (precision) and highest in ROUGE and SBERT Score, capturing more semantic overlap with reference answers, this can also be since the shorter answers generated help improve recall but reduce precision.
    - GPT has lowest of the scores is proven to be less competitive and demands extensive training to match performance of other models.
  - Loss Graph (refer Figure 16):
    - The loss graph indicates smoother convergence for models, except GPT-Neo-1.3B model indicating the need for more training epochs to stabilize.
    - Despite DeepSeek having highest loss it provided a stable score and answer generation capability.

CHAPTER 5 – CONCLUSIONS AND FUTURE WORK

Conclusion

The research evaluated three open and lightweight models' - DeepSeek-R1-Distill-Qwen-1.5B, TinyLlama-1.1B-Chat-v1.0 and GPT-Neo-1.3B, fusion with LoRA fine-tuning and instruction tuning techniques to explore their suitability for small scale businesses by highlighting trade-offs between model size, efficiency and output quality and overall reducing the training time and resource requirements.

LoRA along with instruction tuning has proven critical for fine-tuning large models under limited GPU memory by enabling training with fewer parameters. DeepSeek-R1-Distill-Qwen-1.5B benefited the most from LoRA application, achieving highest quality generated answers with acceptable training time and GPU consumption. TinyLLaMA emerged as the intermediate option, utilizing least GPU and training time while balancing semantic accuracy for its short answers. GPT-Neo with unstable loss curve and benchmark scores proved to be insufficient to match efficiency of other models, making it less optimal for constrained environments and needed extended training epochs.

DeepSeek demonstrated consistent benchmark performance and superior answer quality to other models and is more practical for domain-specific adoption in resource-constrained settings. Proving that small-scale industries can realistically adopt open-source lightweight models, supporting minimum compute to build domain-specific assistants achieving competitive accuracy without extensive hardware costing, training time with minimum code.

Future Work

Followed by the models and methodology tested under this research the scope can be extended to below avenues:

1. Quantization and Compression:
   Quantized LoRA (QLoRA) methods such as 4-bit NormalFloat, Double Quantization and Page Optimizers can further reduce memory utilization by implementing low-precision storage data types (for model weights and parameters), optimizing both model speed and accuracy without sacrificing model performance.
2. Domain-Adaptive Pretraining (DAPT):
   As small-scale industries often operate within niche domains, adapting lightweight models with domain specific pre-training before LoRA fine-tuning can significantly improve performance on downstream tasks by learning domain specific vocabulary and style to match the contextual alignment.
3. Real-World Deployment Studies:
   Conducting longitudinal studies by deploying DeepSeek-powered systems in small enterprises would provide practical insights into usability, cost savings, and adaptability in real operational environments.

References

1. (CRFM), S. C. f. R. o. F. M., 2023. Alpaca: A Strong, Replicable Instruction-Following Model. *Stanford University.*
2. Brown, T. B. et al., 2020. *Language models are few-shot learners.* s.l.:arXiv preprint.
3. DeepSeek-AI, 2025. *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning.* s.l.:arXiv preprint.
4. DeepSeek-AI, 2025. *DeepSeek-V3 technical report.* s.l.:arXiv preprint.
5. Gillgren, F., 2024. *Improving Article Summarization by Fine-tuning GPT-3.5.* s.l.:Uppsala University, Sweden.
6. Handoyo, E. et al., 2018. *Ticketing Chatbot Service using Serverless NLP Technology.* s.l., IEEE.
7. Hoffmann, J. B. S. M. A. B. E. C. T. R. E. d. L. C. D. H. L. W. J. C. A. H. T. N. E. M. K. v. d. D. G. D. B. G. A. O. S. S. K. E., 2022. *raining Compute-Optimal Large Language Models.* s.l.:arXiv preprint.
8. Hu, E. et al., 2021. *LoRA: Low-Rank Adaptation of Large Language Models.* s.l.:arXiv preprint.
9. Inouye, D., Lindo, L., Lee, R. & Allen, E., 2024. *Applied Auto-tuning on LoRA Hyperparameters.* s.l.:https://scholarcommons.scu.edu/cseng_senior.
10. Kayapalı, M. E. et al., 2025. DeepSeek versus ChatGPT: Multimodal artificial intelligence revolutionizing scientific discovery. From language editing to autonomous content generation—Redefining innovation in research and practice. *Knee Surgery, Sports Traumatology, Arthroscopy,* 33(5).
11. Kotsis, K., 2025. ChatGPT and DeepSeek evaluate one another for science education. *Journal of Effective Teaching Methods,* 3(1).
12. Liu, P. et al., 2021. *Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing.* s.l.:arXiv preprint.
13. Li, X., Li, Z., Kosuga, Y. & Bian, V., 2025. *Output length effect on DeepSeek-R1's safety in forced thinking.* s.l.:arXiv preprint.
14. Li, X. & Sarwate, A. D., 2025. *Unraveling the localized latents: Learning stratified manifold structures in LLM embedding space with sparse mixture-of-experts.* s.l.:arXiv preprint.
15. Minaee, S. et al., 2025. *Large Language Models: A Survey.* s.l.:arXiv preprint.
16. Naveed, H. et al., 2024. *A Comprehensive Overview of Large Language Models.* s.l.:arXiv.
17. Rakesh, C. et al., 2025. *Enhancing gender-neutral language with LoRA fine-tuning of DeepSeek-R1 for offline AI applications.* s.l.:SSRN preprint.
18. Russell, S. J. & Norvig, P., 2021. *Artificial Intelligence: A Modern Approach (4th Edition, Global Edition).* s.l.:Pearson (Global Edition).
19. Smiju, I. & Adinath, D., 2025. *Advancements in AI-powered NLP models: A critical analysis of ChatGPT and DeepSeek.* s.l.:Cochin University of Science and Technology, School of Management Studies.

20. Vaswani, A. et al., 2023. *Attention is all you need.* s.l.:arXiv preprint.
21. Wei, J. et al., 2022. *Emergent Abilities of Large Language Models.* s.l.:arXiv preprint.
22. Zhang, P. Z. G. W. T. a. L. W., 2024. *TinyLlama: An Open-Source Small Language Model.* s.l.:arXiv preprint.
23. Zhao, W. X. et al., 2025. *A Survey of Large Language Models.* s.l.:arXiv.

Appendices

Appendices A: Data Source Used

1. WebScarpedDataset_for_DeepSeekchatbot_training.xlsx : Web Scrapped data from AWS Platform
2. val.jsonl : Dataset used for model validation
3. test.jsonl : Dataset used for model evaluation
4. train.jsonl : Dataset used for model training

Appendices B: Code and Generated Answers

1. Data Preparation.ipynb: Data Understanding, EDA and Preparation.
2. Evaluating base model without training.ipynb : DeepSeek base model evaluation without fine-training.
3. Training Deepseek without Lora gave CUDA out of memory error.ipynb : Training DeepSeek without Lora.
4. Deepseek_LORA_7layers.ipynb : DeepSeek model training with LoRA 7 layers
5. Deepseek_LORA_4layers.ipynb : DeepSeek model training with LoRA 4 layers
6. EleutherAI_1_3b.ipynb : GPT model trained model training with LoRA 4 layers
7. TinyLLaMA_1_1B.ipynb: TinyLlama model training with LoRA 4 layers
8. DeepSeek_Model1_generated_answers.xlsx : Answers generated by Deepseek_LORA_7layers.ipynb
9. DeepSeek_Model2_generated_answers.xlsx : Answers generated Deepseek_LORA_4layers.ipynb
10. GPT_generated_answers.xlsx : Answers generated by GPT model
11. TinyLlama_Model2_generated_answers.xlsx : Answers generated by TinyLlama model
12. base_model_evaluation_results.xlsx: Answers generated by DeepSeek base model without fine-training.

Appendices C: Code

GitHub link: https://github.com/utekarmitali/AW_Chatbot