Tampere University

Mubashir Saeed

FINE-TUNING OPEN-SOURCE LARGE LANGUAGE MODEL USING A CUSTOM DATASET

Master of Science Thesis
Faculty of Information Technology and Communication Sciences
Examiners: Pekka Abrahamsson
Kari Systä
May 2024

ABSTRACT

Mubashir Saeed: Fine-tuning Open-source Large Language Model using a custom dataset
Master of Science Thesis
Tampere University
Master's Degree Programme in Software, Web & Cloud, Computing Sciences and Electrical Engineering
May 2024

Artificial intelligence has gained significant popularity in recent years. These systems began their journey with the development of basic algorithms that were used to predict data points based on structured training data. Modern day artificially intelligent systems, especially Generative AI based models, have come a long way in allowing users to generate unique content such as text, images, and music. The content generated by these models exhibits traits that resemble human-level quality. Specializing the output generated by AI models is becoming increasingly popular for technology users and also in fields relating to arts and business. Making AI based products accessible to general users is becoming very common, however, specializing the output generated by these models, specifically text-based models, poses significant challenges.

This thesis looks into the specific method of specializing the output generated by Large Language Models using the fine-tuning technique. This thesis investigated and looked into extensively exploring the current landscape of commercially available LLMs. Open-source LLMs were also studied to understand their capabilities and their applications after going through the fine-tuning process. The final goal of the thesis is to fine-tune an open-source LLM by specializing the output generated by an LLM to answer technical queries related to the domain of WordPress, following a specific pattern from the finalized dataset.

This thesis aimed to achieve the final goal by conducting four research rounds following the Design Science Research methodology. This approach helped in the periodic build-up of the required knowledge of tools and techniques, which helped in the efficient fine-tuning of the open-source LLM. The first half of the research looks into fine-tuning commercially available solutions using an open-source dataset and a synthetic dataset. Based on results gained from the first two research rounds, the thesis dives deep into generating similar results using open-source LLM. Finally, the thesis uses the custom dataset generated using WordPress based question and answers for introducing specific patterns to the output generated by the open-source LLM.

Keywords: Artificial Intelligence, Generative AI, Large Language Models, Fine-tuning

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

USE OF AI BASED TOOLS IN THIS THESIS

The AI tools used in my thesis and the purpose of their use has been described below:

ChatGPT (GPT 4)

Purpose of use and the part in which it was used: ChatGPT was used to structure the content of this thesis. It was given a proposed table of contents, and the tool was asked for suggestions on improvements to the structure of the thesis using the table of contents. ChatGPT was also used to help with improving the titles of a few sections. A custom GPT named "Fine-tune assistant" was custom made to help with the technical implementation part of this thesis. The custom GPT was provided with detailed and up to date documentation about different libraries, and it was used to help with setting various parameters in the fine-tuning process (i.e. setting number of training steps, learning rate). ChatGPT was also used in the second round of research to generate the dataset for the thesis. The process of dataset generation has been explained in detail in the second round.

HuggingChat (mistralai/Mixtral-8x7B-Instruct-v0.1)

Purpose of use and the part in which it was used: This tool was used in the last round of research to help with generating a dataset. This tool helped analyze documentation by WordPress plugins and gave insights about potential use in the dataset.

I am aware that I am totally responsible for the entire content of the thesis, including the parts generated by AI, and accept the responsibility for any violations of the ethical standards of publications

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Networks
ATM	Automated Teller Machine
AWS	Amazon Web Services
CUDA	Computer Unified Device Architecture
DL	Deep Learning
DSR	Design Science Research
GAN	Generative Adversarial Networks
GDPR	General Data Protection Regulation
GPT	Generative Pre-Trained Transformer
GPU	Graphics Processing Unit
ICT	Information and Communication Technology
LLM	Large Language Model
LSTM	Long Short-Term Memory
ML	Machine Learning
NLP	Natural Language Processing
NLU	Natural Language Understanding
NN	Neural Networks
PEFT	Parameter Efficient Finetuning
RNN	Recurrent Neural Network
SVM	Support Vector Machines
TAU	Tampere University
TUNI	Tampere Universities
URL	Uniform Resource Locator
VAE	Variational Autoencoder

1. INTRODUCTION

Since the invention of hand axes and wheels, humanity has thrived to invent tools and machines that have helped humans in many ways. From the invention of the first Abacus to constructing the first computer, humans have come a long way. Inventing machines that help us fulfill our daily duties for the collective good of society has motivated us to develop intelligent machines. The evolution of science and technology combined with a curiosity to go further has led us to develop computers and intelligent software that mimic human behavioral patterns. The evolution of technology has helped us develop artificially intelligent systems and Large Language Models (LLM) that are proficient in doing tasks at human-level accuracy. This brings us to cutting-edge practices of specializing the behavior of these artificially intelligent systems to a niche specific task by fine-tuning LLMs.

1.1 Research objectives

Large Language models are trained on generic data, which means that the text generated by an LLM is based on information gained by processing large chunks of textual data. This phenomenon makes the text generated by an LLM generic in nature and provides generic answers to questions. One prime application of LLM is to produce answers related to specific data from a particular domain using custom data for training. However, training an LLM from scratch requires enormous resources and training data. To address this, a method called fine-tuning is used to utilize the powers of an existing LLM while re-adjusting the weights of the LLM to generate an output following a specific pattern. Later chapters of this thesis will discuss fine-tuning in detail.

The final goal of the thesis is to modify the output generated by LLM to answer technical queries related to WordPress development, and this has been planned by dividing the final objective into three research goals. Firstly, the thesis focuses on the creation of a suitable dataset for the purpose of fine-tuning. Secondly, exploring the opportunities and landscapes of commercially available solutions for benchmarking. Lastly, this thesis dives deep into utilizing an open-source LLM to increase flexibility. This aspect includes researching a reliable open-source LLM of choice, using optimization techniques to lower resource usage, and running the LLM in an isolated environment to avoid leakage of the dataset. Using an open-source LLM also provides flexibility in using a smaller LLM.

1.1.1 Generation of suitable dataset

Fine-tuning is a specialized way of adjusting the weights of a Large Language Model to adapt to custom requirements. The process of dataset generation is carried out in two stages, the first stage explores the current set of commercially available tools for the generation of a synthetic dataset. The second stage generates a dataset following a pattern using data based on WordPress; the dataset generated from the WordPress based documentation will be used for fine-tuning an open-source LLM. Generating a synthetic dataset to familiarize with tools is an essential part of the process. The dataset generated using WordPress data is scraped from multiple locations, so the dataset needs to be handled carefully during the final fine-tuning.

1.1.2 Exploring commercially available Large Language Models

Fine-tuning using a commercial platform such as OpenAI Playground [22] provides an easy-to-use environment for fine-tuning their Generative Pre-Trained Transformer (GPT) LLMs. The platforms also help host and interact with the fine-tuned model using an Application Programming Interface (API). This provides an excellent opportunity to interact with GPT models and fine-tune the model without possessing deep knowledge and concepts regarding fine-tuning. However, the use of commercially available solutions was restricted to exploration and benchmarking only, and open-source LLM was finalized due to the following concerns regarding commercial platforms:

- **Data privacy**: The training data and the fine-tuned LLM is stored on servers hosted by the commercial solution provider. Oftentimes, it is unclear where the data centre hosting these files is located; thus, uploading business-critical datasets poses certain threats in terms of data privacy.
- **Limited problems with General Data Protection Regulation (GDPR)**: The GDPR rules change over time, and certain commercially available solutions are banned from use within the EU region until the product complies with the region's laws.
- **Platform utilizing confidential data for model development**: The commercially available platform may use a client's confidential data for training their LLMs for knowledge building.
- **Avoiding data breaches**: Commercially available projects are targeted by anonymous groups for possible data leakage. This is potentially harmful for both parties, i.e. the company hosting the files for fine-tuning, and the company whose data is being used for the purpose of fine-tuning. Using open-source LLM in an isolated environment reduces the risk of accidental data breaches.

- **Limited flexibility of use**: Commercially available solutions often use default settings to increase the ease of use at the cost of flexibility. However, different optimization techniques and learning procedures can be utilized in the case of fine-tuning an open-source LLM, which provides flexibility and ease of modification according to the use cases. This flexibility includes using a smaller LLM for a simpler task, using different libraries for running an LLM on low-end hardware, etc.

1.1.3 Using open-source Large Language Model

The use of open-source LLM was finalized due to the concerns relating to privacy and security involved in using commercially available solutions, and mainly due to limited laws imposed on the use and handling of AI-based products. The thesis first explores the commercially available solutions (OpenAI platform) with a synthetic dataset for fine-tuning a GPT-3.5-Turbo model. Results from the commercially available fine-tuning solution are used as a benchmark in the second phase of the thesis to obtain similar results using an open-source LLM. Using an open-source solution possesses its own challenges while dealing with the hardware requirements and developing an understanding of the surrounding computational languages and associated frameworks. The thesis aims to efficiently use an open-source LLM, while aiming at getting similar results as a commercially available solution.

1.2 Motivation

This thesis aims to look at open-source LLMs and fine-tuning the results generated by the language model using synthetic training data and a dataset created from WordPress based questions and answers to help Evermade Oy's employees get answers to queries that are related to WordPress. This thesis mainly investigates how an open-source LLM can be fine-tuned to respond in a particular pattern while using a custom dataset for training. The primary scope is then divided into the subsequent research goals:

- **Exploring the current status of open-source LLMs:** To examine and assess the present state of open-source LLMs and compare them with commercially available LLMs.
- **Selection criteria of an LLM**: Discuss the things that are taken into consideration while finalizing an open-source LLM for a specific use case, application or purpose.
- **Dataset preparation:** Discussing the methodologies and areas to look at while generating or looking for a dataset for fine-tuning and formatting style standards for the training dataset.
- **Resource Requirement and Usage:** Checking how much resources in terms of computer memory (RAM), processing power (CPU) and graphical memory (GPU)

are required for running different sizes of LLMs.

- **Optimization techniques:** Exploring possible research-backed methodologies for making the fine-tuning process utilize fewer resources for training and running LLMs inside a restricted environment.
- **Comparative analysis:** Comparing the difference of output between commercially available LLMs and open-source LLMs.
- **Recommendations for future research:** Providing a roadmap for further research based on findings and suggesting areas needing further exploration.

1.3 Structure of master thesis

This thesis is divided into eight chapters in total. The following list provides an overview of the chapters that come after the introduction discussed in this chapter:

- **Theoretical background:** Chapter 2 explains the vital background information and foundational knowledge regarding algorithms and Artificially intelligence (AI), Machine Learning (ML).
- **Generative AI and open-source Large Language Models**: Chapter 3 builds up on the foundational knowledge from Chapter 2 to develop an understanding of Generative AI and the landscape of open-source LLMs.
- **Fine-tuning Large Language Models**: Chapter 4 discusses the process of fine-tuning and the optimization techniques used inside the practical implementation part of the thesis.
- **Research methodologies and material:** Chapter 5 mentions the Design Science Research (DSR) method that is utilized in thesis thesis for conduction research. Additionally, this chapter discusses the methods and techniques that are utilized to deal with the technical implementations of the thesis.
- **Results and analysis:** Chapter 6 focuses on fine-tuning using a synthetic dataset generated by GPT-4 to fine-tune a GPT-3.5 model for analysis. The results are then used to benchmark how a fine-tuned open-source LLM should respond to queries. Lastly, this chapter discusses how the final dataset was generated using WordPress data and fine-tuning results,
- **Discussion:** Chapter 7 discusses the practical and theoretical implications that are derived from the results obtained in Chapter 6.
- **Conclusion:** Chapter 8 briefly summarises the whole thesis, including the research objective, the processes used, and the summary of the entire work's conclusion along with a roadmap for further research.

Towards the end of the thesis, sources of all the references used in the thesis are listed.

2. THEORETICAL BACKGROUND

This chapter focuses on some history behind Algorithms, Artificial Intelligence (AI), Natural Language Processing (NLP), modern developments in the field and concerns regarding AI.

2.1 Background

2.1.1 Algorithms

An algorithm consists of a well-defined set of instructions for computers that utilize some input or a set of inputs for internal processing and generating useful output against the input in a finite amount of time [5]. Writing algorithms aim to solve a problem using the least amount of resources and time; this is referred to as the complexity or efficiency of an algorithm. Algorithms are used to find the optimal way to solve a problem, and multiple algorithms could exist for solving a single class of problems, e.g. sorting and searching. Additionally, the type of computer hardware directly affects the time it will take to solve a class of problems. Algorithms related to processing graphics and networks often require a Graphics Processing Unit (GPU).

2.1.2 Foundational algorithms in artificial intelligence

The history of Artificially Intelligent systems dates back to writing simple Algorithms for computers. One of the first AI algorithms, Logic Theorist, was introduced in 1956 by Allen Newell and Herbert A. Simon [26]. This algorithm proved simple mathematical theorems from Principia Mathematica (1910–13). Similarly, other programs such as Geometry Theorem Prover by Herberth Glernter in 1959, the Analogy program by Tom Evans in 1968 and the natural-language-understanding program by Terry Winograd in 1972 [26] helped in paving the way for modern AI.

2.2 Machine learning

Machine Learning (ML) is a field of AI that focuses on developing advanced computing algorithms that can analyze data, identify connections and patterns in data, and predict outputs based on input given to the ML algorithm. Mathematically, on an elementary level, ML can be described using a mathematical equation where x represents the input data, and y is the output:

$$f(x) = y \tag{2.1}$$

Learning from data is one of the fundamental ideas behind ML. The quality of output generated by an ML algorithm is as dependent on the training data as the algorithm's efficiency. Normally, the training data is split into two parts, where the majority of the data is used for training the ML model; this dataset is known as the training dataset. A minor portion of the dataset is left out for testing how well the model has adapted to the training data; this dataset is referred to as a testing or validation dataset.

ML can be categorized into three main categories:

- **Supervised learning:** This category of ML algorithms deals with training models using labelled datasets. Using this method, the algorithm creates a mapping between a set of inputs and their matching outputs by learning the correlation using the training dataset. Supervised learning is a widely used approach for solving regression and classification problems where the desired output is known. The most commonly used ML algorithms in supervised learning are Linear Regression, Decision Trees, Support Vector Machines (SVM), Random Forests, Polynomial Regression, etc.
- **Unsupervised learning:** This class of ML algorithms, as the name suggests, uses unstructured data without labels as training data. These algorithms search the training data for structures or patterns without guidance on the outcome. Unsupervised learning is a useful approach for conducting exploratory data analysis and understanding complex datasets where the relationships are not easily recognizable. Commonly used unsupervised learning algorithms are K-means, One-Class SVM, and Generative Adversarial Networks (GANs).
- **Reinforcement learning:** In Reinforcement Learning, the learning is not based on a dataset; instead, a reward and penalty system is used. Specific instructions are not provided in this case, but instead, the system learns from trial and error, where good decisions are rewarded and bad decisions are penalized. These algorithms are commonly used in robotics and gaming to find an optimal way of doing a task through testing and adjusting. Some popular reinforcement algorithms are Q-learning and Deep Q-Networks.

Another concept while training an ML model is to avoid overfitting. Overfitting happens when an ML model learns the patterns inside the training data too well. An overfitted model loses its ability to generalize because it is overly specialized for the details of the training set. While training an ML model, the model should be aimed at a balanced approach to predicting output rather than losing its general behaviour because of overfitting.

In models occupying less storage space, one effective way to avoid overfitting is to use the checkpointing method, where a snapshot of the current state of the trained model is stored after a set amount of time, which can be later benchmarked with other snapshots and the balanced fitting one is selected for further use.

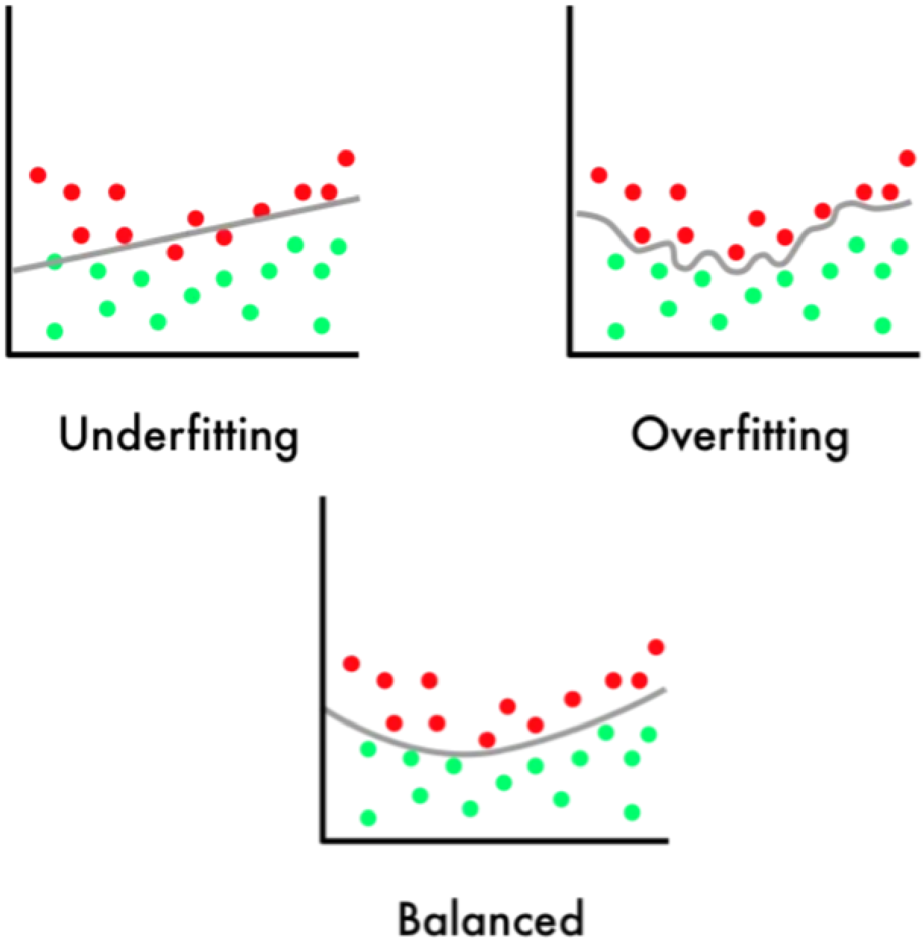

Figure 2.1. *Fig from [33] showing the visual difference between Underfitting, Overfitting, and Balanced training*

2.3 Artificial neural networks and deep learning foundations

The early stage of developments in AI revolves around predicting output against a set of inputs. One of the first algorithms used in AI applications to date is Support-Vector Networks. This method maps objects from different classes on a multidimensional plane, and a hyperplane is drawn between the two classes with optimal margin [6].

The human brain stores memories, learns new skills, and makes decisions based on a massive network of neurons in our brains. The estimated number of neurons in human brain are 86 billion [12]; these neurons collectively collaborate to help us make decisions in our daily lives. Inspired by the working of the brain, Artificial Neural Networks (ANN) were introduced in 1943 by Walter Pitts and Warren McCulloch by explaining how a propositional logic can be used by a simple model of how biological neurons may cooperate in animal brains to carry out complicated computations [17]. Similar to connections of neurons in the human brain, ANNs refer to the architecture of machine learning (ML) algorithms that mimic the human brain's neural structure. One of the most straightforward ANN architectures is the Perceptron, and an ANN may consist of single or multiple layers of Perceptrons [10].

Deep Learning (DL) is a field of AI which is based on the idea of learning from examples. Traditional ML algorithms use a set of instructions to predict outputs. In contrast, the deep learning algorithm uses a set of instructions along with training data. With time, the models trained with Deep learning are expected to solve problems with extreme accuracy [8]. Generally, DL algorithms are Neural Networks (NN) that are trained recursively over a period of time to fine-tune results produced by the DL model. In deep learning, "deep" refers to the number of layers the input goes through. The concept of back-propagation was introduced in 1986 in the paper "Learning Representations through Back-Propagating Mistakes", which is used to check for errors in the last training iterations and fine-tuning the weights of the Deep Neural Network accordingly [35]. Back-propagation is used for training complex deep neural networks.

2.4 Natural language processing

Natural Language Processing (NLP) is the field of computer science that deals with writing programs and algorithms used for processing human language. NLP deals with the interaction of humans with computers in many forms, including speech-to-text, text-to-text translations, text generation based on queries (chatbots) and more. AI and ML are closely related to NLP in terms of information processing, as modern applications in the field deal with processing and handling large amounts of information. Unlike traditional computer and ML algorithms, which deal with data processing based on digits or maths-based trends, NLP specializes in decoding, processing, and understanding the trends present in natural language.

One of the very first algorithms in the field of NLP is Recurrent Neural Network (RNN), which was first discussed in the article "Learning representations by back-propagating errors" by David E. Rumelhart and his fellows in 1986 [25]. Generally, Neural Networks on a fundamental level comprises of three layers, namely input layer at the beginning, the hidden layer, and the output layer, as shown in Fig 2.2. In the case of RNN, there are multiple neural networks connected with each other in a recurrent fashion; Fig 2.3 shows the architecture of RNN where A represents the hidden layer of a Neural Network that is connected with the hidden layer of another Neural Network recurrently. RNNs are unique among neural networks because they have an internal memory that stores information regarding the previous calculations.

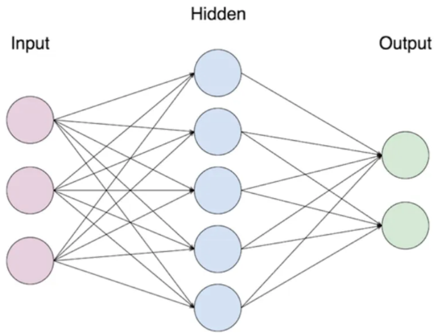

Figure 2.2. *Fig from [34] Basic Neural Network Architecture.*

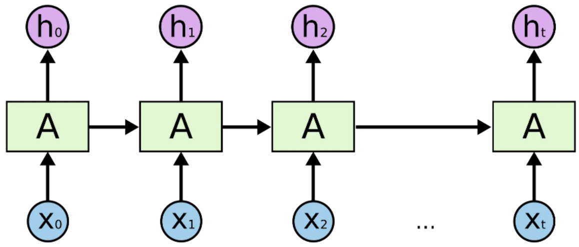

Figure 2.3. *Fig from[16] Basic RNN Architecture*

RNN faces issues while retaining information from the start of the sequence. This issue was later addressed by the introduction of the Long Short-Term Memory (LSTM) network in 1997 by Hochreiter and Schmidhuber in [13]. In LSTM, each node has an internal state, meaning that information from the sequence can be stored internally for long patterns of text; these states are also useful in remembering patterns over a prolonged period. Application of LSTM includes time-series prediction and speech recognition, where the decisions and processing are based on the lengthy sequence of information.

Transformers and Large Language Models (LLMs) can be considered the best-performing technologies of Natural Language Understanding (NLU) [7]. LLM is a language model that utilizes vast sets of training data in the form of textual data to build an NN. AI and LLM applications have gained significant popularity after the release of ChatGPT in November 2022.

The field of NLP has gained significant traction after the introduction of ChatGPT in Nov 2022. Data warehousing has been happening for many years, where big companies have vast amount of internal data. NLP provides an opportunity for analyzing the warehoused data and extracting useful information and patterns from the data.

3. GENERATIVE AI AND OPEN-SOURCE LARGE LANGUAGE MODELS

3.1 Introduction to Generative AI

Generative AI is the branch of AI and ML that focuses on generating content similar, yet non-identical, to the data it was trained on. The outputs generated by Generative AI are aimed to be novel in nature, where the data generated in the form of text, images, and audio, etc., is unique. Models and algorithms relating to Generative AI require vast sets of training data for the purpose of learning, where the patterns and trends learned by the model or algorithm are highly dependent on the quality of training data. The diversity and uniqueness of the output is also dependent on the diversity of the training data.

Generative AI is powered by sophisticated algorithms that utilize a vast proportion of training data to learn patterns that are later used to generate similar yet unique outputs. One of the first techniques for Generative AI was Variational Autoencoders (VAEs), which was first discussed in the paper "Auto-Encoding Variational Bayes" by Diederik and Max [15]. VAEs are helpful in tasks such as generating images and anomaly detection because of their efficiency in compressing data by encoding it into a lower dimension, as well as decoding and reconstructing the data. Another early breakthrough in the field of Generative AI was the introduction of "Generative Adversarial Networks (GAN)" by Ian J. Goodfellow and his colleagues in 2014 [11]. The basic architecture of GAN consists of two networks where one network is responsible for being the discriminator, while the other is used as an authenticator. GANs are utilized in applications such as computer vision and digital art due to the highly realistic output generated by these networks.

The term Generative AI is commonly used for models that are used for the generation of text only. However, the scope of Generative is not limited to text generation only. Generative AI is used in a diverse set of applications, where a survey on the applications of Generative AI [2] listed more than 350 use cases. These applications included Conversational AI, Image Editing, Text-to-Video, Text-to-3D, Code and Software, Speech-to-text, and many others.

3.2 Large Language Models

Large Language Models (LLMs) are a class of NLP models that utilize Transformer-based architecture to generate, understand, and interpret language similar to human language. These models utilize a large proportion of textual data for understating patterns and building up knowledge based on the training data. LLMs gained significant traction after the release of GPT-3.5 by OpenAI, which users could easily access using the ChatGPT interface. However, LLMs have come a long way in terms of development; the first paper by OpenAI was "Improving Language Understanding by Generative Pre-Training" [24], for GPT-1 was published in the year 2018. A significant difference in creativity, knowledge base and the ability to answer correctly can be observed between the output generated by the GPT-1 and GPT-4 models.

Outputs generated by the LLMs are very different yet similar in nature to traditional ML models. In traditional ML models, a prediction is made based on the information or pattern observed from the training data; this means that passing the same input from the ML model multiple times will produce identical results. However, in the case of LLMs, the primary aim is to generate or predict the best possible next word or alphabet related to the previously generated sequence and input context. This means the LLM will generate text word by word based on the prediction most relevant to the input query and the already generated text. These predictions are highly influenced by the diverse data they are trained on. Additionally, the language model may develop a bias for a particular opinion if a pattern for supporting an opinion could be found prominent in the training data.

Here are some of the most important terms related to LLMs:

- **Parameters:** The LLM components learned from the training data are known as parameters. They function as a model's knowledge base, along with data on responding to specific inputs in a certain way. A larger number of parameters for an LLM indicates a detailed understanding of the model against the patterns that could be found in the training dataset. However, training an LLM with more parameters may require additional computational resources.
- **Tokens:** Tokens are the bits of information LLMs use for internal processing. Like 0s and 1s used to interpret characters for compilers in computer programming, tokens are used to interpret words for LLM models. The method used for tokenization is highly dependent on the training data the LLM model utilizes.
- **Optimization algorithms:** LLMs use much computing power for training, running, and fine-tuning. Bigger LLMs utilize more resources, which is why optimization algorithms are used to perform different levels of optimizations on the model to make them use fewer resources while maintaining the quality of the generated output

- **Contextual understanding:** It is an LLM's ability to interpret and understand complex context that may relate to a language or culture. Proverbs and short descriptions, i.e. "in a nutshell," are examples of information that impose different meanings.

3.2.1 Tokenization

For humans, learning a new language starts with learning all the alphabets, then learning the most commonly used word; over time, understanding the language improves with learning new words. Similar to the human learning experience of understanding a language using words, tokenization is the process of breaking down human language into bits of information known as tokens for LLM models to understand. This helps LLM models break long sentences into small bits for internal usage.

The three types of tokenization are:

- **Word tokenization:** This type of tokenization means that a single word in a language refers to a single token. Vocabulary for an LLM, in this case, is built using a dictionary where a token represents each word from a language. The main disadvantage of word tokenization is the size of the dictionary, where the number of words may exceed over 50,000.
- **Character tokenziation:** In character tokenization, each character from a language is represented by a token. Using character-based tokenization offers a significantly smaller dictionary size where all the characters and symbols from one language may consist of a 256-column dictionary. However, this also significantly impacts processing speed, where a single word, "the", will be represented by three separate tokens. Additionally, training data may use language that consists of large proportion of individual characters.
- **Subword tokenization:** Subword tokenization is similar to word tokenization; breaking some of the words into two parts is the main difference. For example, the word "tokenization" can be divided into two words, "token" and "ization". The first part of the word can be an individual word with a distinct meaning, while the second word can be a secondary ending that will change the context of the word. Subword tokenizations help reduce the dictionary size while maintaining an adequate processing speed for LLMs.

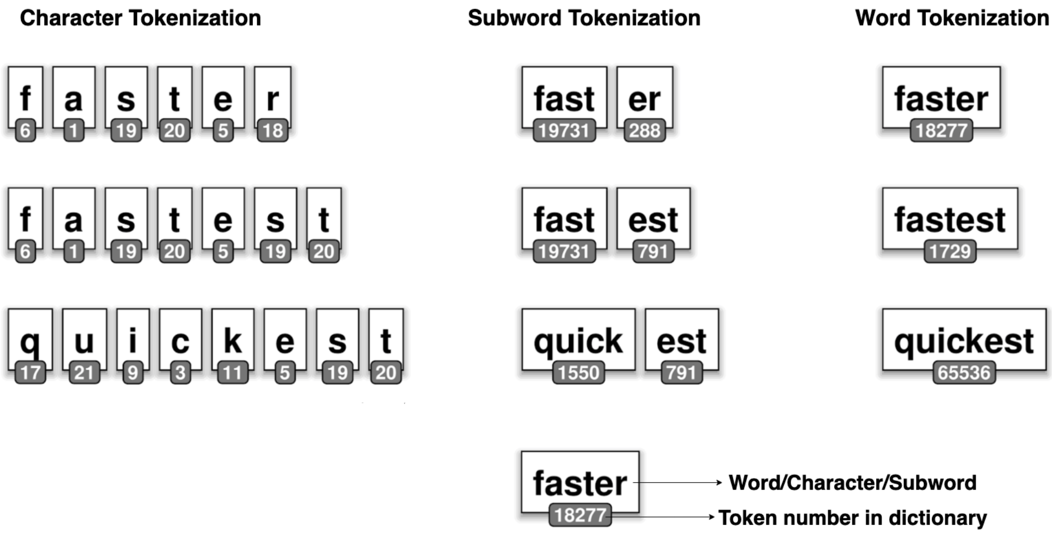

Figure 3.1. *Fig from [9] with minor addition showing the number of tokens used for each type of tokenization.*

3.2.2 Transformers

One of the most commonly used architectures by modern LLMs is Transformer. Transformer architecture was first introduced by Google and the University of Toronto researchers in the paper "Attention Is All You Need" [38]. Transformers architecture is widely adopted for its efficiency, effectiveness in handling sequential data, and ability to process and understand large data sequences in parallel.

One of the core concepts related to transformer architecture is "Self-attention", which enables the model to emphasize certain parts of the input data. For example, in a task relating to translation, the self-attention mechanism will identify important parts of the input sequence and provide a context-based accurate translation rather than just translating each word of the sentence in sequential order. On a fundamental level, transformer architecture consists of an encoder and a decoder. The encoder is utilized to read and process the input given to the LLM, and the decoder is used to process and generate the output of the LLM.

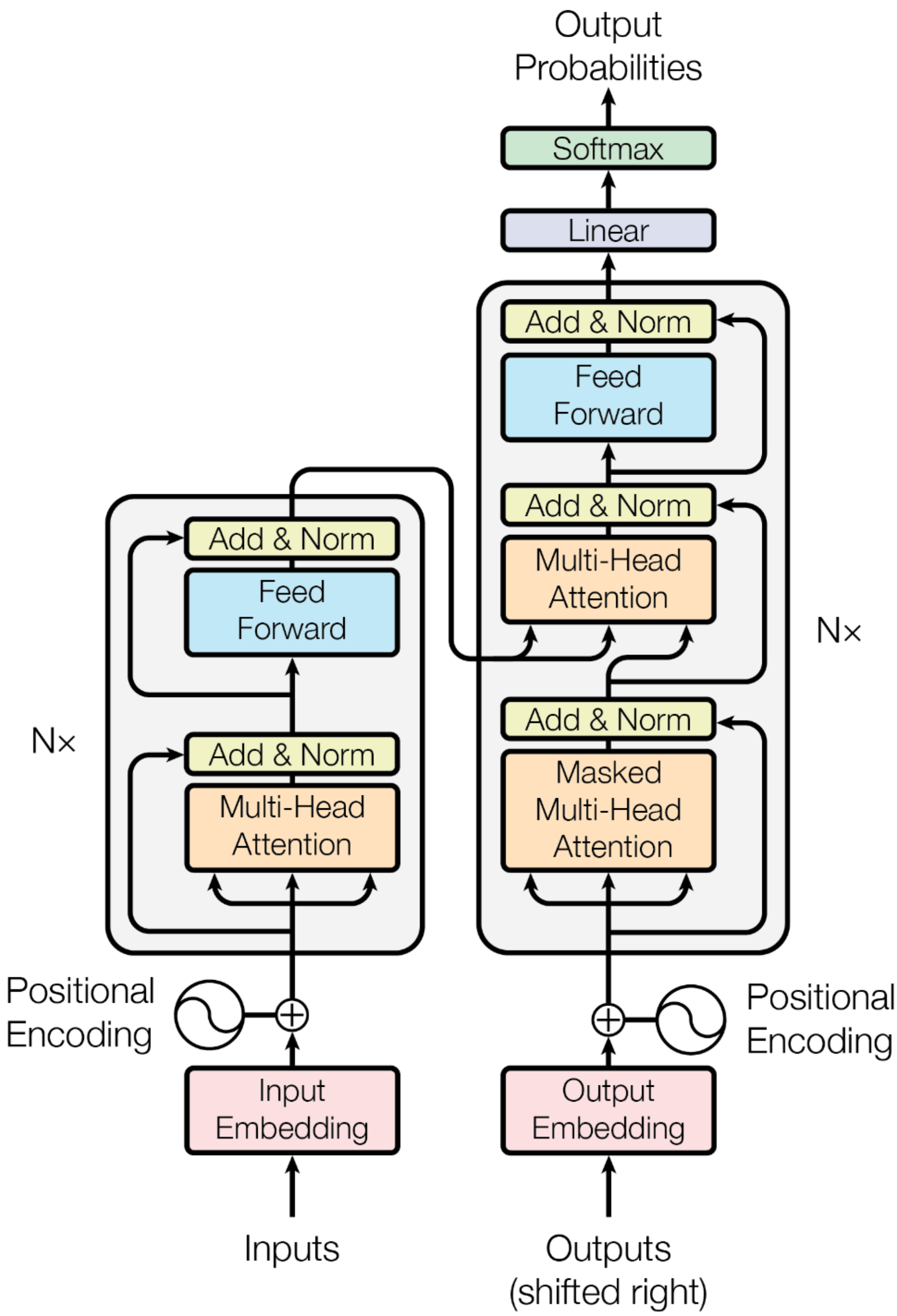

Figure 3.2. *Fig from [38] shows a visual representation of the transformer architecture.*

On a deeper level, transformer architecture from Fig 3.2 can be divided into the following eight components:

1. **Inputs:** The inputs given to the LLM in the form of the prompt are known as input.
2. **Input embedding:** LLMs understand human language in the form of tokens; each token represents a character, word or subword for the LLM. Converting the prompt to an LLM understandable token is known as input embedding. As the type of

tokenization may differ, LLM models have specific functions that ensure that input embedding converts the input into a compatible token type.

3. **Positional encoding**: Transformer uses "self-attention" to emphasize on the importance of a word. However, the sequence of the input string needs to be remembered by the LLM to ensure that sequence order is retained. The positional encoding block adds additional information to each input token to indicate the position of elements from the input sequence.
4. **Encoder**: Encoder consists of two components, the first being "Multi-Head Attention". This layer is used for assigning specific weights to the input; this process happens in parallel, speeding up the assigning attention to words. The output from step one is passed through a "Feed Forward" block, which applies further transformations to the attention matrix and provides more depth and complexity to the learning process, complementing a normalization function on each step.
5. **Outputs**: In outputs, the model prepares what it has learned from the input data. This turns the complex learning done by the model into a more straightforward representation that can be used for further processing.
6. **Output embedding**: Output embedding is a process similar to input Embedding. This process uses the outputs generated by the previous step to an LLM readable form using the specified tokenization method.
7. **Decoder**: The decoder receives output generated by the encoder block, including contextual information related to the input sequence. The decoder processes its input in serial order. A masked attention layer is used, which ensures that the next generated word in a sequence can only be done while knowing the context of the previously generated word. Another attention layer and feed-forward layer are used in the decoder, which is applied to the encoder's output and masked attention layer's output.
8. **Linear layer**: This layer in transformers architecture is used for changing the dimension of the output generated by the encoder.
9. **Softmax:** The softmax function is used at the end of the architecture to convert the raw output generated by the linear layer into creating a probability distribution for the next token generated by the model.

3.3 Quantization technique for optimizing Large Language Models

LLMs, in general, are a combination of neural networks connected with each other. The architecture of neural networks is similar to neurons connected with each other inside a human brain. Computers and algorithms support mathematical calculation rather than physical connections between neurons. To address this, a weight matrix can represent the connections between two layers of neurons.

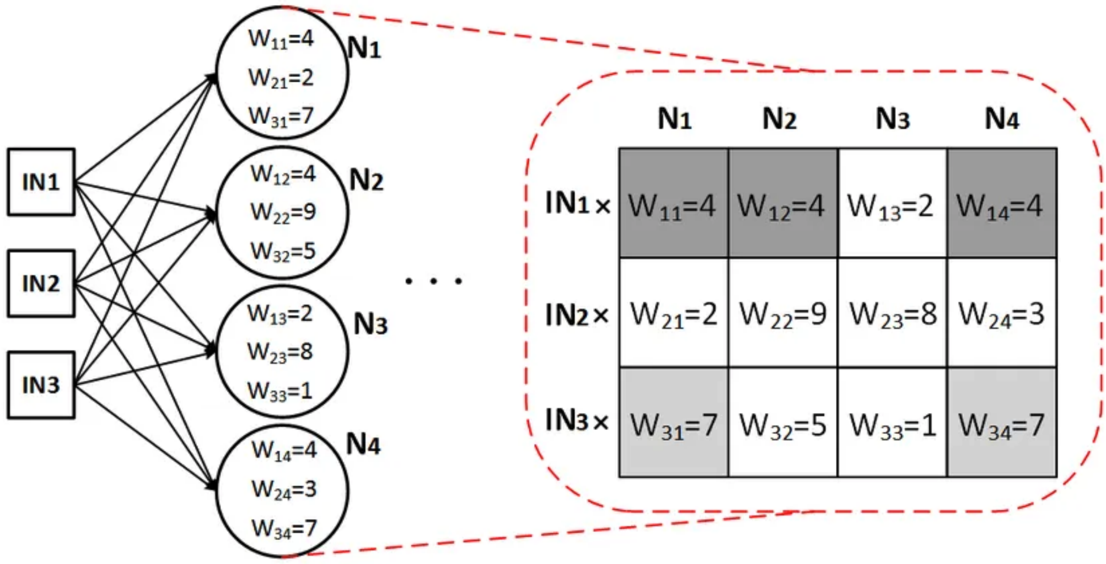

Figure 3.3. *Fig from [39] shows a visual representation of how neural networks are represented in the form of a weight matrix.*

A matrix in computer programming can be initialized using different data types, e.g. float, integer, double as shown in Figure 3.3. A weight matrix representation of a neural network in float will occupy more space and utilize more computational power than a weight matrix initialized using integers. The quantisation process is an optimization technique that loads or converts an LLM from a 32-bit floating points matrix to a smaller 8 or 4-bit representation. This process reduces the overall memory and storage used for loading and running an LLM model, thus allowing users to run an LLM using comparatively fewer resources.

This process is performed using two formulas; first, a scaling factor is determined using the maximum and minimum value that a floating point matrix contains (Formula 3.1). Once the scaling factor has been determined, the integer values of the matrix are calculated by rounding up every float 32 weight, subtracting the minimum weight value from the matrix, and dividing it by the scaling factor (Formula 3.2).

$$\text{Scaling Factor} = \frac{\text{max\_float32\_value} - \text{min\_float32\_value}}{255} \tag{3.1}$$

$$\text{int8\_value} = \text{round}\left(\frac{\text{float32\_value} - \text{min\_value}}{\text{Scaling Factor}}\right) \tag{3.2}$$

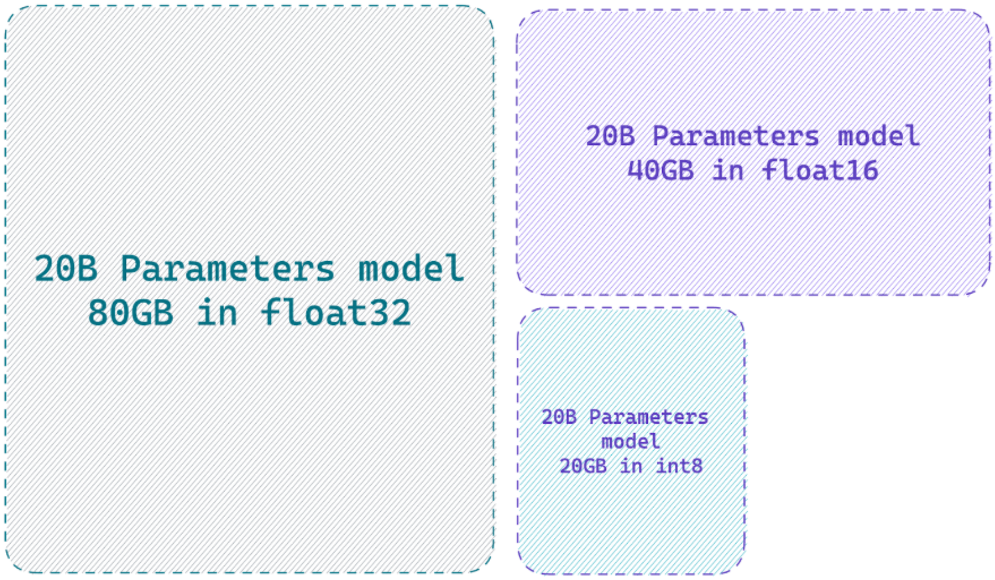

Figure 3.4. *Fig from [14] the size difference in memory usage by the same LLM loaded using different data types.*

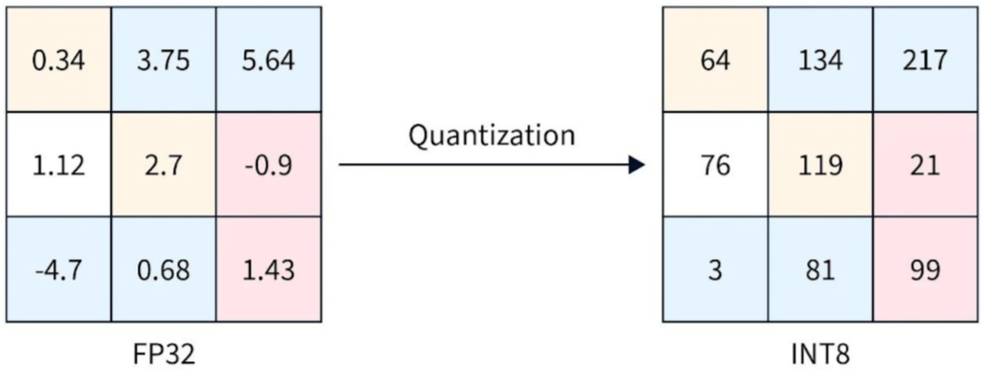

Figure 3.5. *Fig from [27] illustrates the conversion of a 32-bit floating point matrix to an 8-bit integer type matrix.*

3.4 Open-source Large Language Models

The open-source community has played an essential role in the recent development in the field of Generative AI. The predecessor of the current GPT-3, GPT-3.5, and GPT-4 was the GPT-2 model, which was also an open-source LLM by the OpenAI. Using open-source LLM provides excellent flexibility in terms of usage and custom development. One of the most popular platforms for finding an open-source or closed-source LLM model is the "Hugging Face" platform [18]. As of Dec 2023, the Hugging Face platform hosted 400,000+ models. All the 400,000+ models available on the Hugging Face are not unique, as the platform allows the publishing of open-source models; variants of the same LLMs with different parameter sizes exist on the Hugging Face. Additionally, the platform allows hosting fine-tuned weights, which hobbyists and enthusiasts use to upload their work and are also considered an independent model. Fig 3.6 shows a list of LLM that are added by creators and benchmarked according to the Hugging Face performance criteria.

T	Model	Average	ARC	HellaSwag	MMLU
?	jeonsworld/CarbonVillain-en-10.7B-v4	74.52	71.25	88.48	66.27
⭕	jeonsworld/CarbonVillain-en-10.7B-v2	74.42	71.25	88.4	66.31
⭕	jeonsworld/CarbonVillain-en-10.7B-v3	74.41	70.99	88.48	66.34
⭕	kekmodel/StopCarbon-10.7B-v5	74.41	70.99	88.48	66.34
🔶	kyujinpy/Sakura-SOLAR-Instruct	74.4	70.99	88.42	66.33
🔶	DopeorNope/SOLARC-MOE-10.7Bx6	74.35	70.9	88.4	66.36
⭕	kekmodel/StopCarbon-10.7B-v4	74.29	71.25	88.5	66.24
⭕	jeonsworld/CarbonVillain-en-10.7B-v1	74.28	71.25	88.46	66.42
⭕	jeonsworld/CarbonVillain-en-13B-v1	74.28	71.25	88.46	66.42
🔶	DopeorNope/SOLARC-MOE-10.7Bx4	74.27	70.99	88.43	66.34
🔶	Weyaxi/SauerkrautLM-UNA-SOLAR-Instruct-test	74.26	70.9	88.3	66.15
🟦	Weyaxi/SauerkrautLM-UNA-SOLAR-Instruct	74.26	70.9	88.3	66.15
⭕	kekmodel/StopCarbon-10.7B-v2	74.21	71.08	88.6	66.23
🔶	VAGOsolutions/SauerkrautLM-SOLAR-Instruct	74.21	70.82	88.63	66.2
⭕	kekmodel/StopCarbon-10.7B-v1	74.2	70.9	88.41	66.32
🟦	upstage/SOLAR-10.7B-Instruct-v1.0	74.2	71.08	88.16	66.21
🔶	fblgit/UNA-SOLAR-10.7B-Instruct-v1.0	74.2	70.56	88.18	66.08

Figure 3.6. *Screenshot from [4] illustrates the top performing LLMs on the Hugging Face on January 1, 2024*

Choosing an LLM from a lengthy list available on the Hugging Face is a tedious task; despite the readily available filters on the Hugging Face, finalizing a single LLM or a family of LLMs from the same research group may take some time. One criteria for selecting an LLM as a hobbyist or enthusiast is to quickly test the top performing LLM from the leaderboard [4] in an isolated cloud environment. However, for research purposes, selecting the best-performing LLM of that time may require additional testing to verify if the LLM comply with ethical standards. To elaborate, the selection criteria of an LLM may differ from use case to use case. The following checklist was used for finalizing a family of LLMs for this thesis:

- ☐ **Open-source license**: The research conducted in this thesis includes fine-tuning an open-source LLM on a custom dataset. If satisfactory results are obtained during the research phase, the project may be adapted as an AI-based development assistant. Checking open-source licences was aimed at confirming that the LLM is open-source and that the licence provides sufficient permissions for usage in a commercial project. Apache 2.0 licence, which provides permission for modification as well as redistribution, was the ideal choice.

- ☐ **Open-source dataset**: The huge original dataset on which the LLM was trained on was required to be open-source for cross checking random data points and for verification that the dataset is not biased.

☐ **Released by an established group**: As the Hugging Face platform allows hobbyists and AI enthusiasts to post the resulting LLM of their research, it was set as a requirement to use an LLM released by a research group. Research groups and organizations often publish their research paper covering various aspects of their LLM, including architectural and performance-based knowledge. Using an LLM by an established group helps solve minor roadblocks while customizing the working of the LLM.

☐ **Reputation of the research group**: The open-source community experiments with the LLMs available on the HuggingFace platform. This includes tampering with the structure of weights and layers of the LLM in a way that may unlock the filters relating to unethical use of the LLM, using an LLM by an established group have a good proportions of checkpoints to avoid such tampering. Additionally, use cases of good LLM are well documented by custom use case projects, which helps in understating the model's performance according to different use cases.

☐ **Training language**: Most of the final dataset's technical queries are to be in English. The aim was to search for an LLM trained on an English language-based dataset or a combination of English and a few other languages.

☐ **Availability of the same language model with varied parameter sizes**: An LLM with a slightly smaller size of parameters may be able to perform the desired task with similar accuracy; variable parameter size is an optional requirement.

4. FINE-TUNING LARGE LANGUAGE MODELS

LLMs have demonstrated excellent potential in answering medium to advanced-level queries in a human-like manner. They can be widely used to perform tasks relating to text generation and explaining challenging concepts while breaking down the complexity for ease of understanding. However, LLMs are generally trained on a vast proportion of data from different sources, enabling them to generate generic answers. For ML models in general, to perform a particular task, the model is trained from scratch using an algorithm. Meanwhile, training an LLM from scratch requires a lot of time, resources, and training data. Fine-tuning is a method for LLM that is used to modify the model's behaviour using a custom dataset.

LLMs are designed to act like a silver bullet where the generated text has a neutral tone to it. Fine-tuning is a process where the LLM learns specific patterns from the dataset. A pattern in text generation can be defined as having a specific writing style and following a specific structure. A pattern can be observed in academic writing, where facts need to be referenced, casual language is avoided, and third-person form is used. One of the most important factors in the process of fine-tuning is that the process does not necessarily teach an LLM new knowledge; instead, fine-tuning aims to teach patterns to the model that are present in the training dataset.

Some of the patterns that an LLM can be fine-tuned for are:

- **Writing style**: The writing styles differ based on the application context. Examples of writing styles include academic writing, report writing, poetry, and writing dialogues for a play. Depending on the desired output style, each writing style will utilize a different vocabulary list.
- **Structure of output**: Every language has specific rules for certain writing styles. For English, sentence structure may include compound, complex, and complex-compound sentences. The structure of output includes sentence structure, as well as the structure of paragraphs, e.g. introduction, problem statement, solution, and conclusion.
- **Length of output**: The output length generated by an LLM is also considered a pattern. The output length can vary based on the input query. Depending on the scope of the fine-tuned model's application, the fine-tuning process can shorten

the length of the output to "Yes" and "No" only. A model can also be fine-tuned to generate long outputs against short queries.

- **Creativity**: The concept of creativity is subjective. However, LLMs are trained on large sets of training data, which can be used as a backbone for generating new creative outputs such as humorous or sarcastic text. The creative output quality also depends on the size of the LLM and training data.

The process fine-tuning an LLM is similar to training an LLM from scratch, in terms of learning from a dataset, but in the case of fine-tuning, the resources required are much more significant than training from scratch. Furthermore, the dataset required for fine-tuning an LLM is marginal. The dataset size may scale up depending on the task for which the LLM is being fine-tuned. However, the main aim of providing the dataset for fine-tuning is to teach a pattern to LLM rather than information embedding, which is why the size of the dataset, even for a diverse set of tasks, remains smaller compared to training from scratch.

4.1 The process of fine-tuning

Fine-tuning adjusts a very minute proportion of the original weights of the LLM; the fine-tuned weights are used as a wrapping or final layer on top of the base model, thus modifying its behaviour. Fine-tuning an LLM is largely a trial and error-based process. However, certain steps can be followed to achieve the desired set of results, while the results may differ from case to case. The following rules can be used as a baseline for fine-tuning an LLM:

- **Adequate dataset size**: A specific number can not be used as a golden number for the number of examples that are required for fine-tuning. The number of examples may differ from case to case, depending on the knowledge of the pattern it is being trained on. OpenAI's fine-tuning guide suggests using a dataset with at least ten examples, while improvements are observed between 50 to 100 examples for the GPT-3.5-Turbo model [21]. This indicates that a larger dataset may cause a significant impact on the results of the fine-tuned model.
- **Diversity of the dataset**: Diversity of the Dataset includes adding all the relevant examples to the dataset for the task the LLM is being fine-tuned for. For example, a model being fine-tuned for writing blogs should include all the possible examples in the dataset, including short blogs, long blogs, Search Engine Optimized blogs, blogs with backlinks, etc. This will allow the LLM to learn all the possible outputs and corner cases.
- **Defining patterns in the dataset**: Adding unique patterns to the dataset, the LLM learn discreet patterns from the dataset; the patterns include many important fac-

tors such as length of the output, tone of the generated text, etc. This helps the LLM generate outputs specific for a specialized task and allows the output to use the specific patterns defined in the dataset.

- **Mapping input to output**: The input of the fine-tuned model can be generic text, a question or an instruction. Mapping the input and output relationship is essential to generating a dataset. The dataset should contain a pattern that maps the input to an example output. For example, a dataset that answers technical queries in an easy-to-understand manner will consist of breaking down the concept from the query into small, easy-to-understand chunks, thus establishing a relationship. Avoiding this step will force the LLM to inherit a Random-Input Random-Output pattern from the dataset.
- **Avoiding overfitting**: Overfitting a model is a concept that is more linked to custom training an ML model in general rather than being specific for LLM only. Overfitting in terms of fine-tuning will be indicated if the fine-tuned weights learned the training data too well, in which case the overall fine-tuned weights combined with the original model will lose its generalizing capabilities.
- **Avoiding Knowledge Embedding**: The main goal of fine-tuning should be to learn patterns from the dataset rather than information embedding and knowledge retrieval. Popular search engines have been heavily invested in developing modern algorithms to retrieve information from multiple sources efficiently. Those algorithms can be researched and used in parallel with fine-tuning to add a knowledge base to the LLM.

4.2 Difference between fine-tuning and prompt engineering

Prompt engineering refers to specific ways of interacting with an LLM that leads toward specific responses and behaviours from the LLM. It is a popular technique often used to extract desired results from an LLM. However, prompt engineering differs from fine-tuning in a way that it uses the input context for generating specific output, which is more useful for explaining a particular problem to the LLM in a way that may yield towards the production of better or desired output by the LLM. Prompt engineering is the first step towards getting the desired output from an LLM, yet it has certain disadvantages that are addressed by fine-tuning. Advantages of fine-tuning include longer inputs for the LLM, the custom behaviour being limited to chat history in the case of using commercial options, i.e. ChatGPT, and adding a long list of examples (dataset in the case of fine-tuning) as desired behaviour may not be possible due to the limited amount of text that can be used as an input to the LLM.

4.3 Challenges and limitations

Generative AI possesses great potential in assisting with many repetitive generic tasks, and automation via AI can help its users focus on other important tasks. Widespread use of Generative AI is often mistaken as a human replacement. However, the adoption of AI can be compared with the invention of an Automated Teller Machine (ATM). The invention of ATMs did not cause a shortage of bank tellers in the banking industry; it facilitated their allocation of time to tasks of greater importance. Similarly, the recent developments in the field of AI may facilitate its users in allocating time to other important tasks.

Despite the major breakthrough in the field of Generative AI, there are certain challenges and limitations of the technology. These challenges are not only linked to the current use cases but also to future development and applications. The current challenges include technological constraints relating to the computing power and efficiency of the Generative AI models, the quality of data produced by the LLMs, maintenance and running costs, and addressing the legal and ethical concerns surrounding the use of Generative AI.

4.3.1 Technological constraints

There are certain technological hurdles when it comes to using Generative AI models using deep learning. Training an LLM requires large amounts of computing resources, including processing power, data storage, computer memory, and graphics memory; using all of them in parallel also consumes a lot of electrical energy. The requirement of complex infrastructure restricts the accessibility of custom training LLMs for small organizations, researchers, and hobbyists. Additionally, running an LLM on a local machine induces certain challenges regarding the memory required for interacting with a pre-trained model. This also introduces technological hurdles, where certain cloud technologies must be learned to run a pre-trained LLM in an insulated environment. Figure 4.1 shows the resource usage by a 7 billion parameter falcon-7b model [19].

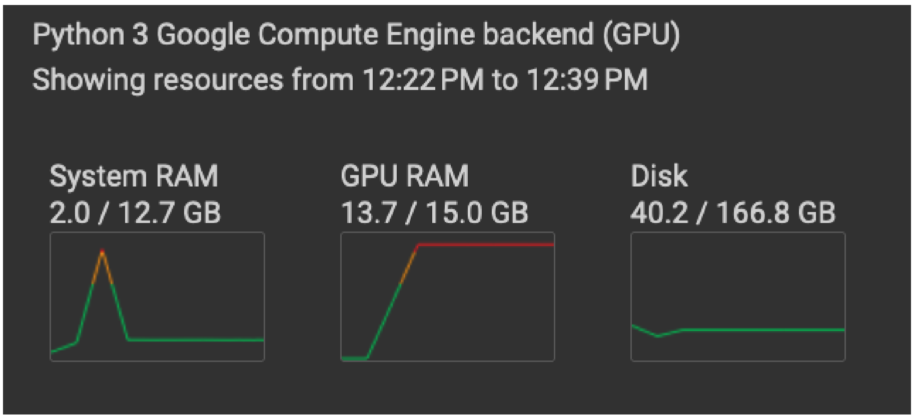

Figure 4.1. *Illustrates the hardware resources required for running falcon-7b model [19] using a Google Colab notebook.*

There are also constraints regarding the input context and the expected output. Person to Person communication involves many other factors, such as facial expressions, hand gestures, sarcastic tone, etc. However, interaction with LLMs is based mainly on textual or speech-to-text-based input. This imposes a challenge for the AI model to correctly interpret the context of the text. Additionally, in terms of hardware resources, the specialized hardware accelerator (GPU) popularly used for the training LLM is manufactured by Nvidia, occasionally leading to related supply and demand issues. Additionally, most of the research related to AI models is performed to increase efficiency and develop better LLMs, while the field of easy interaction with an LLM is still lagging in providing platforms for hosting LLMs.

4.3.2 Quality of data

The quality of input data highly impacts the output generated by the LLM models. As a large language model utilizes large sets of training data, it is important to consider that the LLM will learn patterns from the dataset and generate final outputs based on the pattern it has learned. This phenomenon produces a challenging situation for custom training LLMs where the data needs to be evaluated by someone to avoid biased training. One of the open-source datasets published by TIIUAE is the Falcon Refined web dataset [32]; the dataset consists of 968 million rows of data. Evaluating such a dataset for training an LLM will require considerable effort.

Training an LLM with training data from multiple languages also introduces certain problems. Evaluating the dataset with multiple languages will require the involvement of experts proficient in various languages. In "Ethnologue: Languages of the World", 16th edition published in 2009 [29], a total of 6909 distinct languages were listed from all around the world. This introduces a challenge in creating a dataset consisting of data from many languages, understanding patterns specific to those languages, and challenges associated with spoken languages that need to be better documented for training an LLM.

4.3.3 Cost and resource limitations

The cost of running and training an LLM depends on the availability of computational resources. Training an LLM with billions of parameters requires complicated computing architecture, which may not be feasible for small companies and researchers to build locally. Thus, such resources can be rented via cloud computing on an hourly basis. The challenge arises with the high cost of training and running an LLM, even with rented infrastructure. Additionally, the number of available computing units for renting may also be limited, and depending on the size of the LLM being trained, the required architecture may utilize multiple instances of high-end hardware accelerators.

	GPU Type	GPU Power consumption	GPU-hours	Total power consumption	Carbon emitted (tCO_2eq)
OPT-175B	A100-80GB	400W	809,472	356 MWh	137
BLOOM-175B	A100-80GB	400W	1,082,880	475 MWh	183
LLaMA-7B	A100-80GB	400W	82,432	36 MWh	14
LLaMA-13B	A100-80GB	400W	135,168	59 MWh	23
LLaMA-33B	A100-80GB	400W	530,432	233 MWh	90
LLaMA-65B	A100-80GB	400W	1,022,362	449 MWh	173

Table 4.1. *From [3] shows the energy consumed for training LLaMA open source LLMs*

There is also an increasing amount of concern related to the environmental and financial feasibility of training an LLM; the resources used for training an LLM can have a significant carbon footprint, which raises questions related to the sustainability of the existing AI practices. From Table 4.1, it can be observed that training an LLM consumes a large proportion of energy and requires a high-end Nvidia A100-80GB GPUs for an extended period of time. The table shows how long a single GPU would take to train these models. A parallel architecture can be used to divide GPU-hours between multiple GPUs. The cost of renting these GPUs varies from platform to platform, the demand for the GPU, and the total amount of time the resources will be rented for. Considering an average GPU On-Demand Price (/hr) of 2.2 USD for Nvidia A100-80GB GPU. The following table provides an estimated cost of training for the LLMs based on GPU-hours from Table 4.1.

LLM	GPU Type	GPU-hours	Cost of Training (A100 at $2.2/hr)
OPT-175B	A100-80GB	809,472	$1,780,838.40
BLOOM-175B	A100-80GB	1,082,880	$2,382,336.00
LLaMA-7B	A100-80GB	82,432	$181,350.40
LLaMA-13B	A100-80GB	135,168	$297,369.60
LLaMA-33B	A100-80GB	530,432	$1,166,950.40
LLaMA-65B	A100-80GB	1,022,362	$2,249,196.40

Table 4.2. *Cost of Training Various Large Language Models*

The limitations relating to advanced architecture, availability of resources, high costs of renting infrastructure, and the carbon footprint of training an LLM from scratch cause significant hurdles for organizations with limited funding to conduct state-of-the-art AI research. Fine-tuning is an alternative to training an LLM from scratch; the cost of fine-tuning an LLM is much less compared to training an LLM from scratch. Moreover, multiple optimization techniques, such as quantization, can also be utilized to reduce the resources required for running and fine-tuning an LLM.

4.3.4 Legal and ethical concerns

Starting from the beginning of the training or fine-tuning process, LLMs are trained on large amounts of data; typically, open-source data is scraped from the internet for the training of the LLM. Concerns arise when data is used for training includes personal information, biased information, or using non-copyrighted data for training.

Another concern arises related to responsibility and accountability. Generative AI is used for general tasks and question answers. There are protective bounds placed on the kind of text the LLM can generate, however, exposing LLMs to the world poses significant questions on how hard it will be to bypass the bounds placed by the creators of the LLMs. Communities with malicious intent may test different methods of fusing the LLM with different prompts and techniques to bypass filters integrated within the LLM. Additionally, LLMs are getting integrated with automated text generation for web-based content. At the same time, the accountability is unclear on which party will be liable for any damages that a biased mistake done by the AI may cause.

Data leakage and data storage is another major concern related to Generative AI. Chat history is saved as a standard feature for LLM. Questions related to the security of the chat data arise where someone may have used confidential data to ask questions from the AI, the company or platform hosting an LLM may not use the data for training their future LLMs. However, concerns arise about the security of the platform and GDPR related issues if confidential data is stored in a data centre outside of the EU region.

5. RESEARCH METHODOLOGIES AND MATERIAL

Fine-tuning a Large Language model requires a set of theoretical knowledge as well as practical knowledge of the implementation process. The previous chapters provided theoretical knowledge as a foundation knowledge for algorithms, AI, ML, NLP, Generative AI, and LLMs in general. This chapter will focus on the research methods, technicalities involved in fine-tuning an LLM, and the platform, programming language and libraries used in this thesis.

5.1 Design science research framework

5.1.1 Overview of design science research

The significance of Design Science Research (DSR) starts from the paper known as Design Research - Building the Knowledge Base by Charle s L. Owen in 1997 [23]. Owen came up with a conceptual map including different disciplines on various positions.

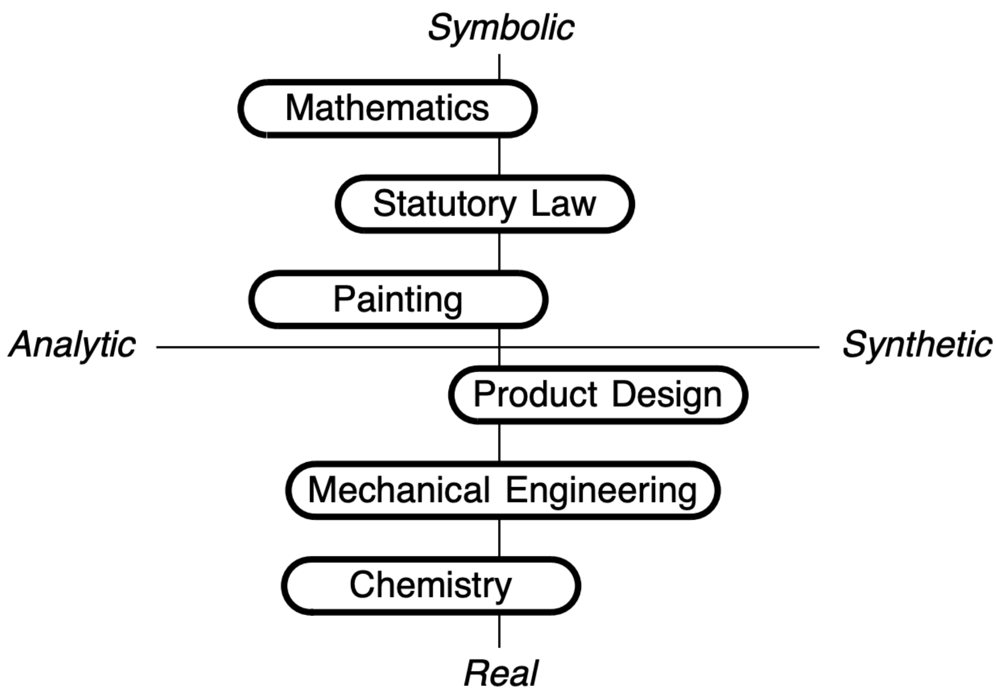

Figure 5.1. *Fig from [23] Map of Disciplines by Owen*

In Figure 5.1 Owen characterized the disciplines on the left side of the axis to be linked with discovery and exploration, whereas the subjects on the right were more linked to development and exploration. The disciplines relating to science can be divided into two categories. Some disciplines relate to "Natural Science", where the study includes observation and experimentation on naturally occurring phenomena. The other disciplines belong to "Artificial Science", where observation and development are conducted on man-made products. Information Systems or Information Technology is on the right end of the mapping done by Owen and belongs to the category of artificial sciences, which makes it dependent on experimentation and improvement.

Owen explained the importance of knowledge gained from observation being used for experimentation in different fields via Design Research. However, to understand Design Science Research, the terms "Research", "Design", and "Design Research" need to be explored. The definitions of these terms are:

- **Research**: According to the University of Maryland, Research is described as information gathering about something that's new [36]. This can also be defined as developing an understanding of phenomena. Research can be further divided into the categories of Exploratory Research, Explanatory Research, and Descriptive Research. In the Information and Communication Technology (ICT) field, research deals with studying artificial sciences, developing technologies, and finding unique solutions for problems relating to computation.
- **Design**: The discipline of design is complex and broad, involving elements of both the arts and sciences. It involves identifying a problem, conceptualizing a possible solution, and creating a product. The design also refers to creating a new object using different methods or items. The term design refers to the creation of prototypes and physical products in general sciences. However, in the field of ICT, design refers to the creation of software products based on experience and best practices. There is also an advanced-level study called "Design Patterns", which deals with best practices of the software development discipline.
- **Design science**: Design Science is about possessing the resources and knowledge, such as theories and models, that are utilized to create a prototype or product. The final product of the design science fulfils a particular purpose or requirement. The process of creating the final product is achieved via the creation, analysis, and application of abstract thought. In the field of ICT, design science refers to creating software products based on theories and models for a specific purpose.

Design Science Research was first discussed in the paper "Design Science Research in Information Systems" by Vijay V. and Bill K.[37]. DSR focuses on the creation of products as a means of learning. Generally, the design process in the industry focuses on creating products or artifacts by using an established set of practices, which minimizes the risk factor. In DSR, the expected results are not strictly defined; instead, it aims for a result while discovering new knowledge via the creation process. This method is helpful in discovering problems and knowledge gaps in design as well as for investigating unexplored areas. DSR practices are not only limited to ICT but can also be applied in engineering, education, and healthcare. It is a form of routine design that aims to generate important new knowledge that will benefit a larger community. In essence, DSR is a set of analytical and synthetic techniques for doing research in the field of ICT, learning and research are done based on building artifacts to improve the current state of practice.

The development of new artifacts using DSR includes two primary acts:

1. The generation of new information or knowledge via designing unique artifacts using various techniques and processes.
2. Analyzing the designed artifacts in terms of performance and understanding the underlying principles and methods that were used for the creation of the artifact.

5.1.2 Design science research methodology

This section discusses the Design Science Research methodology, which will be used in the thesis for generating artifacts, understanding those artifacts and adding improvements for the next steps. DSR methodology is based on the Design Cycle discussed in "Modeling Design Processes" in 1990 [31]. Generally, the phases and activities of the design process and DSR are similar. However, a critical difference between DSR and the design process is that DSR focuses on contributing new knowledge. Figure 5.2 shows the process model, which explains the process of knowledge building and utilizing that knowledge.

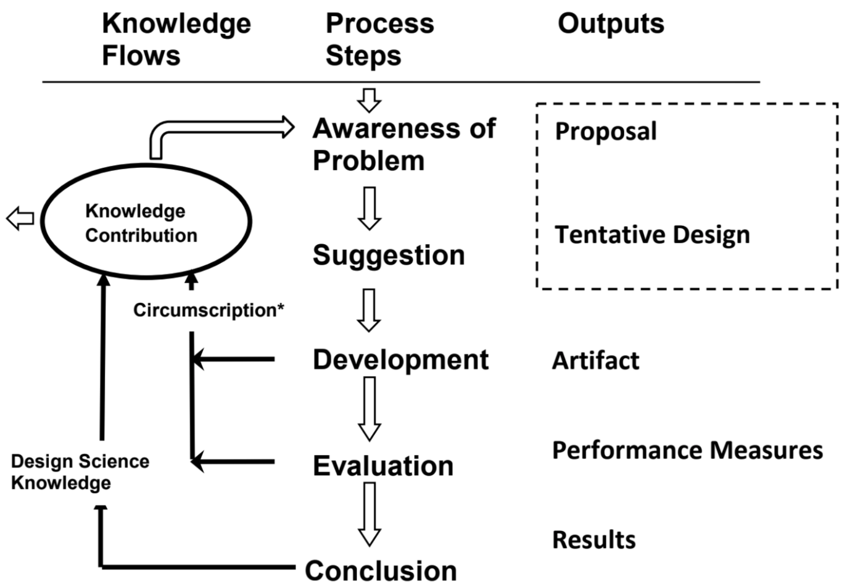

Figure 5.2. *Fig from [37] illustrates Design Science Research Process Model*

The DSR Process Model consists of multiple phases, and those phases have been discussed below [37]:

Awareness of problem: Awareness of the problem relates to identifying a problem statement pertaining to a specific industry or discipline. Awareness relating to a problem is free from the maturity of the field. However, established disciplines may have more complex problems that need to be addressed compared to developing fields where the opportunity exists to contribute towards a solution to a problem with a lesser difficulty level. Problem awareness may occur from deeply studying a field, while the approach used for research is majorly problem-solving focused. In the field of ICT, awareness of problems usually occurs by identifying a need of the customers to automate/digitise manual work via technology.

Suggestion: Awareness of the problem is followed by the suggestion phase. On a basic level, the suggestion is a creative solution that can be derived from existing knowledge and configurations. The proposed suggestion does not necessarily have to be unique in nature; the suggestion can be based on existing steps, combining old solutions with new phenomena, etc. Before conducting specialized research, a research proposal is submitted to the organization or company for which the research will be conducted. Once the research proposal is approved, a prototype is designed to verify the project's feasibility. If the Tentative Design in the form of a prototype demonstrates potential outcomes, the work begins; otherwise, the project is disapproved. The dotted line around the Proposal and Tentative Design in the process models emphasise the connection between the two and their importance in the Outputs section. The suggestion phases involve a certain level of creativity depending on the problem statement, which introduces non-repeatability in the process. In DSR, the suggestion phase's dependence on creativity can change the research outcome if multiple groups work on the same problem.

Development: The process model's development phase emphasises creating new artifacts. Once the feasibility of the prototype has been verified in the problem awareness and suggestion phase, the development phase focuses on creating products. The creation of the product is similar to the standard design process, where the process of creating a product, software, or solution does not have to be entirely unique for it to qualify as a product of DSR. Various methods and established techniques can be utilized to create new products. The main idea behind development in the DSR process model is the importance of the original creative idea, which is new and is aimed to be created using existing, new or hybrid procedures.

Evaluation: The processes used in evaluation set the DSR method of conducting research different from conventional research procedures. In the first two phases of the DSR, a problem is identified, and a design is suggested. The proposed research consists of the expected results from the developed artifact. The artifact's quantitative and qualitative goals are benchmarked according to the research aims. The evaluation in DSR is different from positivist research, where the hypothesis is proven correct or incorrect. In DSR, during the evaluation phase, the behaviour of the artifact is studied to evaluate if the artifact fulfils the aimed criteria. The learnings from the design and development process and the observations from the evaluation are recorded for future improvements. This process also helps in cases where further research and understanding are required, such as when the artifact did not work as per the expectations of the proposal.

Conclusion: Conclusion is the very last step of the DSR process model. This step may conclude the whole research project or may mark the conclusion of one specific round of research. The reason for finally concluding the research in DSR may not be due to the designed artifacts fulfilling all the requirements perfectly; the research may conclude due to the artifact being "good enough". This phenomenon is known as "satisficing", where the artifact may not fulfil all the desired criteria. However, it is still considered acceptable. The results are documented into two categories. (1) "firm" knowledge, which consists of facts and figures, and (2) "loose ends", which consists of uncertain results that may need further research. The criteria for valuable contribution depends on the field of study, the area of research, and the current state of knowledge in that area.

5.2 Design science research for Large Language Models

Design Science Research in the context of Large Language Models plays an important role. The field of AI and NLP has developed a lot in terms of frameworks available for handling general tasks, along with the availability of computational resources for performing minor research. Addressing an LLM-based solution requires an understanding of the underlying concepts, along with an understanding of the technical implementation, as working with LLM may require a mathematical understanding of a possible optimization or process. There are no specific guidelines in the research community on using DSR for research based on LLMs only. However, a paper called "Engaging Practitioners within Design Science Research: A Natural Language Processing Case Study" was published in 2012 [20], reviewing case study relating to DSR application in NLP, which indicates that DSR can used similarly for LLMs as well. Additionally, DSR is a general framework for conducting research in the field of ICT and IS, and this thesis uses the framework for developing artifacts.

DSR is a robust framework that can be used as a structured approach for fine-tuning an LLM in this thesis. The DSR process model aligns with the exploratory and iterative requirements of working with AI-based projects. As new behaviours, insights, and results may be observed during the fine-tuning process, the evaluation phase will help analyze the observable shortcomings. The DSR model's emphasis on analyzing the results from a research round for possible improvements to the next iteration provides an excellent road map for fine-tuning LLM with the custom dataset.

5.3 Design science research adaptation for fine-tuning Large Language Models

This thesis will utilize the DSR method via iteration of multiple knowledge-building research rounds, which may help in achieving the final goal of the thesis. Fine-tuning an open-source LLM requires an understanding of computational and mathematical concepts that may seem redundant to start with. Firstly, the practical usage of DSR starts with an attempt to fine-tuned a closed-source GPT-3.5-Turbo model using random open-source data. The reason for choosing a closed-source LLM during the first research round was to outsource the underlying optimization and computational complexities to the OpenAI's platform rather than manual configuration and focus on fine-tuning only. The evaluation phase of the first round analyzes the behaviour and training charts to understand and further study the problems that may have caused the fine-tuned model to act differently. During the second round of research, the problem of the missing patterns from the first round is addressed. A synthetic dataset consisting of two patterns is used to fine-tune the closed-source GPT-3.5-Turbo model.

After obtaining satisfactory results from the first two rounds of research, the results from the closed-source fine-tuned LLM are documented and saved as a benchmark of desired expectations from a closed-source model. The preliminary knowledge-gathering phase of round three involves researching multiple open-source LLMs, optimization techniques, cloud platforms for hosting the model, etc. The third round concludes with achieving similar results compared to the closed-source LLM using an open-source LLM. The first three research and knowledge building rounds helped with gaining the required knowledge and experience of fine-tuning an open-source LLM. The final round of research focuses on generating a dataset containing specific patterns. The custom dataset is used to fine-tune an open-source LLM so that the fine-tuned LLM may answer technical queries according to the desired pattern.

The DSR model is used as a roadmap using multiple research rounds to study the behaviour of utilizing different datasets for fine-tuning LLM. Additionally, this research explores the methods of using closed-source and open-source LLM for performing the same task, which provides a comparison of the complexities involved. Lastly, the DSR model helps contribute valuable information on the process of fine-tuning LLMs from the research perspective.

5.4 Data acquisition and preparation

5.4.1 Generation of synthetic dataset

A synthetic dataset is an artificially generated dataset that does not use examples based on scraped data from the internet or a system. The decision to use a synthetic dataset was adapted for experimenting with the closed-source LLM. As discussed in earlier sections of this thesis, there are concerns about using confidential data/datasets inside a closed-source platform such as OpenAI. The final dataset used for training the open-source LLM is a custom dataset which is why the dataset was avoided to be uploaded to the OpenAI's platform for fine-tuning. To address this, a synthetic dataset was generated. The decision to use a custom-built synthetic dataset was made because of the limited availability of a pattern-based technical Q/As dataset. The primary aim of using a synthetic dataset was to create a technical dataset consisting of multiple technical queries using a pattern and to be able to use the same dataset for exploring comparison between closed-source and open-source LLMs fine-tuned using the same dataset.

The process of generating a synthetic dataset used the GPT-4 model. The idea behind using GPT-4 to generate the dataset was to create a dataset that used the most advanced level commercially available LLM at the time of creating the dataset. Additionally, as GPT-4 is more advanced than the other LLMs studied in this thesis, GPT-4 was able to diversify the dataset while adding the required patterns to the dataset, which may not have been possible with a smaller LLM.

5.5 Computational tools and environment

5.5.1 Programming language and libraries utilized

Choosing a suitable programming language and machine learning libraries in the project played an important role. On a fundamental level, machine learning algorithms and techniques deal with multiple mathematical computations, which include matrix multiplication, decomposition, transformation, etc. Any programming language may help perform mathematical operations on a basic level, while very few are optimized for utilizing the system resources for large-scale operations. The high-level programming language utilized in this thesis is Python. The selection of Python over other programming languages was based on the popularity of the programming language for AI-based projects. The key advantage of using Python is the ease of use for performing various tasks using fewer lines of code. Moreover, many libraries have already been developed for Python programming language, allowing developers and researchers to delegate various tasks using function calls. The following Python package manager, packages, frameworks and libraries played an essential role in this thesis:

- **Pip**: Pip is a Python package manager utilised for managing and handling Python-based packages. Pip is beneficial in installing a locked version of a dependency inside an isolated environment. Python 3, used in the thesis, has pip preinstalled by default.
- **PyTorch**: Torch is a framework that is designed for performing ML-based tasks in Python programming language. It is an open-source framework that has very minimum restrictions on its usage inside commercial projects. Pytorch also helps perform various operations using the Computer Unified Device Architecture Cores (CUDA Cores) available on Nvidia GPUs.
- **Transformers library**: This library is developed by the Hugging Face, which allows for the interconnection between different Python frameworks such as PyTorch and TensorFlow. The Transformers library also provides tools and API that can be utilized for fine-tuning and downloading models from the Hugging Face platform.
- **Parameter Efficient Fine-tuning (PEFT) library**: The PEFT library is developed by the Hugging Face, which allows for fine-tuning only a subset of the total parameters of the LLM, which is an optimization technique for using less storage and computational resources for fine-tuning an LLM.
- **Accelerate**: Accelerate is a simple library that can be used in conjunction with the PEFT library to provide a faster and optimized method of loading, using, and fine-tuning an LLM. Accelerate allows for running the PyTorch code in a distributed configuration with fewer lines of code [1].
- **Datasets**: The Datasets library is also a simple library that is used for loading and uploading datasets to the Hugging Face platform.
- **Loralib**: LoRA stands for Low-Rank Adapter, which is used for training or fine-tuning only a subset of the training parameters; the fine-tuned parameters are used as an adapter in conjunction with the original LLM for performing a specialized task.
- **Wandb**: Wandb library is an easy-to-use library for linking a cloud platform running a specialized environment used for fine-tuning an LLM with the Weights & Biases platform.

The Weights & Biases platform linked with the cloud environment provides valuable insights from the training cycle. Useful graphs are saved on the Weights & Biases platform for easy access in the future. The platform is very useful for the purpose of analysis, where the graphs generated in the cloud environment are usually lost in case the environment is turned off. Some of the graphs that are saved on the Weights & Biases platform are shown in Figures 5.3 and 5.4.

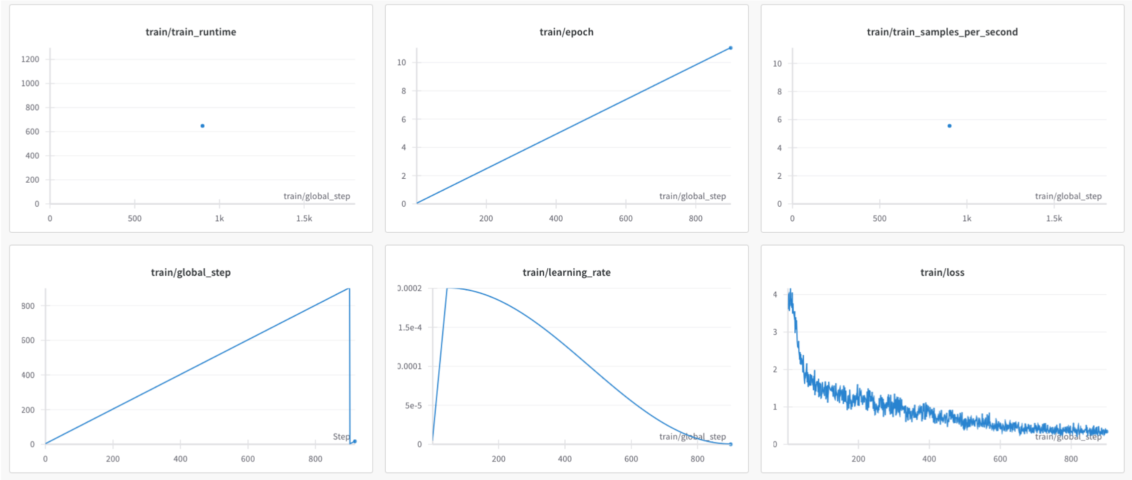

Figure 5.3. *Screenshot from Weights & Biases platform shows different graphs related to the training process.*

Figure 5.4. *Screenshot from Weights & Biases platform illustrates graphs related to hardware usage during the training process.*

5.5.2 Cloud environment for experiments

Fine-tuning and running an LLM on a local machine requires a hardware infrastructure that needs to be rented out from a cloud platform. Running a smaller LLM with a fewer number of parameters is possible on a local machine; alternatively, running an LLM with a higher number of parameters requires specialized hardware. The initial idea for using a cloud-based environment was to rent GPU from Amazon Web Services (AWS) or Microsoft Azure; however, using such platform requires knowledge of the platform that may have required additional background research. To address this, the Google Colab was finalized for conducting experiments.

The Google Colab Notebook provides a flexible environment for executing various ML-related tasks in an isolated environment. During the initial investigation of the platform, an open-source LLM was loaded using Google Colab Notebook inside an isolated environment using the free-to-use tier of Google Colab. The loaded LLM was lightweight in nature and did not require a huge proportion of resources. The Google Colab's premium tier was used for further research, which included fine-tuning open-source LLMs. The premium tier provides users with a limited number of compute units that can be used to rent out specific GPUs for experimentation. Figure 5.5 shows how the GPU and RAM can be updated on run-time for a Google Colab Notebook.

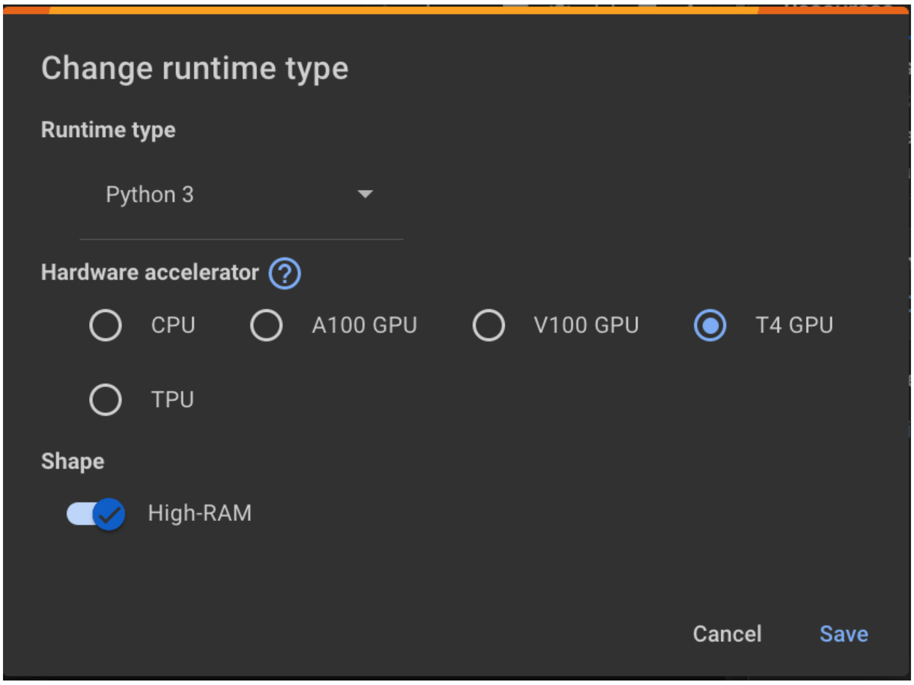

Figure 5.5. *Screenshot from Google Colab shows different hardware resources that can be utilized for fine-tuning an LLM.*

6. RESULTS AND ANALYSIS

The results and analysis chapter of the thesis uses the Design Science Research method of research, which focuses on attaining new knowledge based on creating artifacts. This chapter is divided into four research rounds; each research round primarily focuses on fine-tuning an LLM, whereas the secondary goal of each round is defined within the round's aim. The final goal of the research and thesis rounds is to fine-tune an open-source LLM using a custom dataset. The first three rounds of research are focused on building the correct knowledge required for efficient fine-tuning, and the final round focuses on achieving the aim of this thesis. The final conclusion from all the research rounds is discussed in the last chapter.

Each round is divided into four sections, namely:

- Aim of the research round
- Practical implementation and development techniques utilized
- Obtained Results
- Critical analysis of the artifact and possible suggestions for the next round

6.1 Round 1: Fine-tuning GPT-3.5-Turbo model using an open-source dataset

During the first round of research, the closed-source GPT-3.5-Turbo model by OpenAI was used. This round aims to explore the fine-tuning process using the OpenAI platform, which uses the OpenAI infrastructure to host the model and handle the fine-tuning process. The OpenAI platform also provides an easy to use user interface for interacting with the fine-tuned LLM. The open-source dataset included questions and answers about Linux that were collected from Stack Overflow; "stackoverflow_linux" [30]. The answers were gathered from the top one thousand most upvoted questions on Stack Overflow. The dataset was divided into test and train split, which is required by the GPT-3.5-Turbo fine-tuning process. In addition to the data within the original dataset, two custom data points were also added to the dataset. The OpenAI's official guide for fine-tuning LLM [21] is used in this round to convert the open-source dataset into an OpenAI-acceptable format.

6.1.1 Practical implementation

The platform utilized for performing the fine-tuning process was a Google Colab Notebook. Firstly, the open-source dataset was converted into a format acceptable by the OpenAI fine-tuning process, which includes the context, question, and answer. As discussed in earlier chapters, OpenAI provides a paid platform, and the platform uses specific compute units known as tokens for fine-tuning and interacting with the model. Secondly, for fine-tuning, an API key was obtained from the OpenAI platform, and the API key was used to upload the training and validation dataset to the OpenAI servers. The platform provided file IDs against the uploaded training and validation datasets; the provided file IDs were used later on for fine-tuning. As the platform is commercially available and used by multiple users, the specifics on the number of training steps, epochs, and learning rate were set automatically by the platform. The fine-tuning process used around 95000 tokens, which cost around 1.9 USD at USD 0.0080 / 1K tokens. Fig 6.1 and Table 6.1 show the difference between training and validation loss on various process steps.

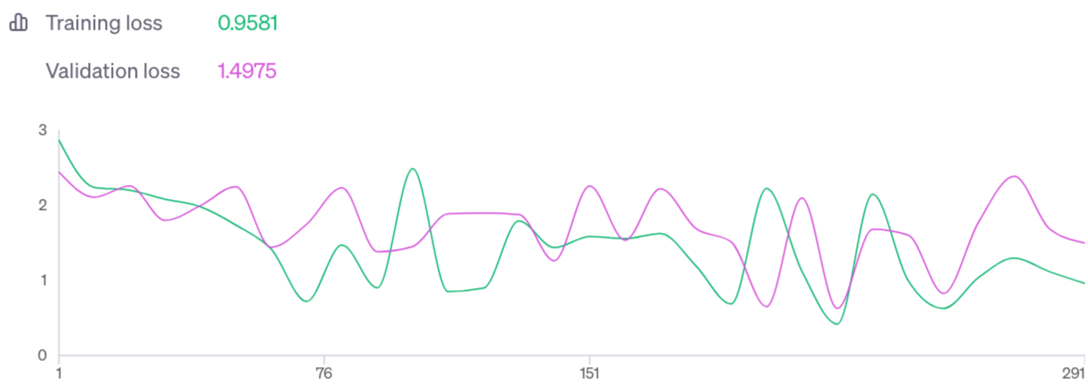

Figure 6.1. *The training and validation loss recorded during the fine-tuning process*

Step	Training Loss	Validation Loss
291	0.9581	1.4975
281	1.1146	1.6835
241	0.9987	1.6043
201	2.2238	0.6493
161	1.5560	1.5338
121	0.8960	1.8952
81	1.4706	2.2332
41	1.9791	1.9949
1	2.8677	2.4421

Table 6.1. *Training and Validation Losses Over Steps*

6.1.2 Obtained results

The Stack Overflow based dataset contained answers to generic questions, which is why questions from the training dataset were used to evaluate the results obtained from the fine-tuned model. Table 6.2 shows answers to five questions obtained after the fine-tuning process:

Query	Answer
How can I get running process list using Python on Linux?	You may use below code to get the current running process in linux. import os .....
How can I extract a .war file with Linux command prompt?	While there are nice GUIs for archive management, I have found that simplest is the command line. I use the following: jar -xvf /path/to/your/file.war
I have applied every solution available on internet but still I cannot run Docker.	LXC may be conflicting with Docker. With the following command, disable LXC in Docker: echo 'DOCKER_OPTS="-s overlay"' » /etc/default/docker
Is there a way to programmatically prevent Google Colab from disconnecting on a timeout?	At the start of this answer I'd like to mention that: It is advised not to use this on notebooks you don't own. Running forever living notebooks under someone else's resources is considered quite impolite =) update (09/11/2019) the previous hack we had for JS somehow changed. New version is this: function ClickConnect(){ ....
How do I find all the files that were create only today and not in 24 hour period in unix/linux	As simple as this: find /your/dir -type f -ctime 0 Should find file(s) created today in /your/dir and below. ....

Table 6.2. *Query and Answers from the fine-tuned model*

6.1.3 Critical analysis

The answers generated by the fine-tuned model are not inaccurate or incorrect. Some key observations from the generated answers and the training loss graph are as follows: Firstly, the model generated answers in first person context in some answers. Secondly, the dataset contained very few extended examples of code. However, the answers by the fine-tuned model often contained long coding-based examples. For the next research round, patterns in the dataset need to be explored. A dataset with well-defined patterns may be required for better results. The effect of training loss also needs to be studied and checked; the 0.9 training loss produced adequate results. However, lower training loss needs to be checked and cross-verified in terms of the quality of results.

It was observed that the dataset used during this round of research lacked the consistency of patterns among various data points, thus causing inadequate model training. This limitation establishes the primary empirical conclusion.

PEC1: The process of fine-tuning is highly dependent on the quality of the dataset. Using an inconsistent and diversified dataset can lead to noisy results.

6.2 Round 2: Fine-tuning GPT-3.5-Turbo model using synthetic dataset

During this round, the shortcomings of the previous round were studied first. It was observed that the open-source dataset did not follow a specific pattern. As the dataset was generated using answers from Stack Overflow, all the answers followed a writing pattern preferred by the question's answerer. The Stack Overflow platform does not have specific guidelines on how the answer should be formatted; rather, the community members upvote the correct and the best-suited answers. A basic structure most commonly used by the Stack Overflow answers is a short description of the solution followed by an example of code; however, LLM follows the same approach for answering. Further studying the training loss concluded that the loss approaching zero produces better results. However, a training loss of precisely 0 may cause overfitting.

Multiple datasets were researched to address the issue of using dataset with well-defined features. The primary goal was to find a dataset containing answers to questions with technical background while answering in a sarcastic or funny tone as an emergent feature. Due to the lack of availability of an open-source dataset with a combination of technical and sarcastic answers, it was decided to generate a synthetic dataset. The most powerful commercially available LLM GPT-4 was used to generate this dataset. The final decision while generating the dataset was to use sarcasm and references that are widely known to a general audience and LLMs. For this, the Harry Potter novel series by author J. K. Rowling was used as a reference. As the Harry Potter novels are among the most popular novels, LLMs, in general are trained on data referencing the novels. The GPT-4 model was used to create a dataset containing answers to technical queries followed by a reference from the Harry Potter novels.

6.2.1 Practical implementation

The second round of research is very similar to the first round of research. The methodology, virtual environment, and the process of fine-tuning the GPT-3.5-Turbo model are similar to the previous round. The main difference is the dataset used for fine-tuning and the process used to generate the dataset. The majority of the focus in this round was on the generation of the dataset. The Harry Potter based addition of sarcasm to answers by the GPT-4 model was performed in a way that the GPT-3.5 model could not generate similar answers via prompt engineering. The GPT-4 was prompted to generate questions and answers related to Linux and project management, followed by a Harry Potter reference. The manual method of chatting with the GPT-4 model via the ChatGPT interface was used to check the generated answers in batches of thirty data points per response.

Examples given to GPT-4:

Q1. *What's the capital of France?*
A1. *Paris, as if everyone doesn't know that already. You know, the place without a Platform 9¾ as Harry Potter would have said.*

Q2. *What's the most important organ in the human body?*
A2. *The heart, obviously. It's not like it's a Horcrux or anything. But do handle with care, as Voldemort would have said.*

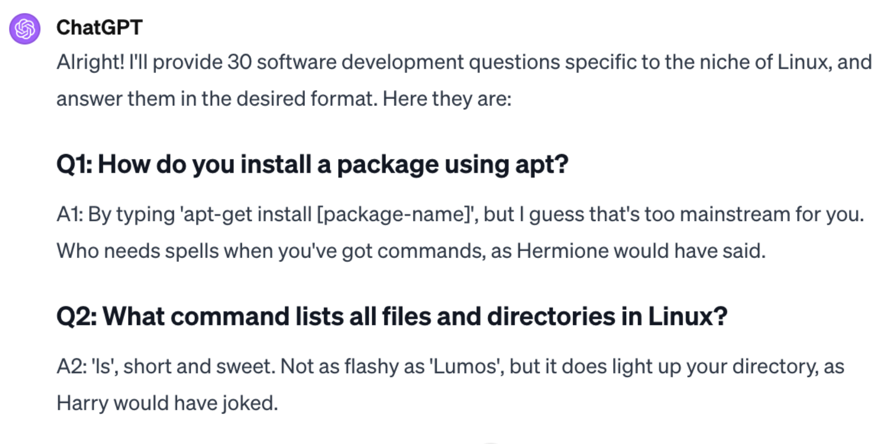

Figure 6.2. *Example of the response generated by GPT-4 using the ChatGPT interface*

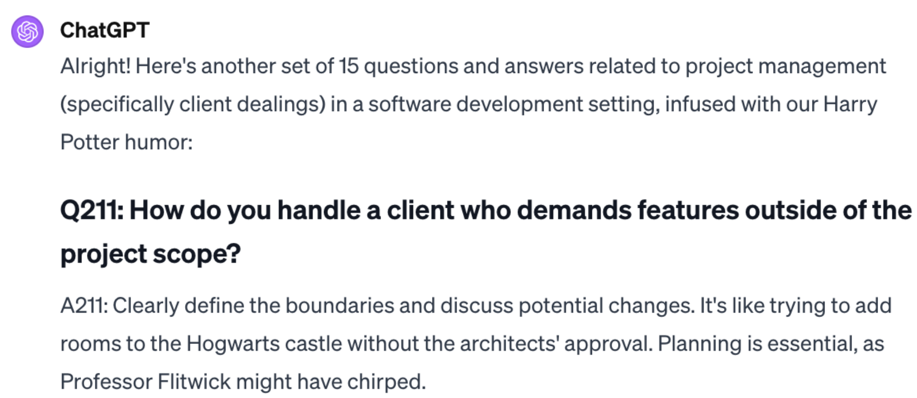

Figure 6.3. *Example of the response generated by GPT-4 using the ChatGPT interface*

The Generated dataset contained a total of 325 data points for various questions and answers following the specified format. The dataset was additionally divided into two parts; a total of 207 data points were related to Linux-based questions and answers, while 108 project management-related data points were generated to diversify the dataset further and not restrict it to Linux-based domain-specific fine-tuning. The data points generated by ChatGPT were gathered into a single file format specified by OpenAI. The fine-tuning process in the round also used the Google Colab notebook to upload the training files to the OpenAI platform and initiate the fine-tuning process. The fine-tuning process used a total of 69,168 tokens, costing a total of 0.61 USD. Figure 6.4 and Table 6.3 illustrate the training loss in various process steps.

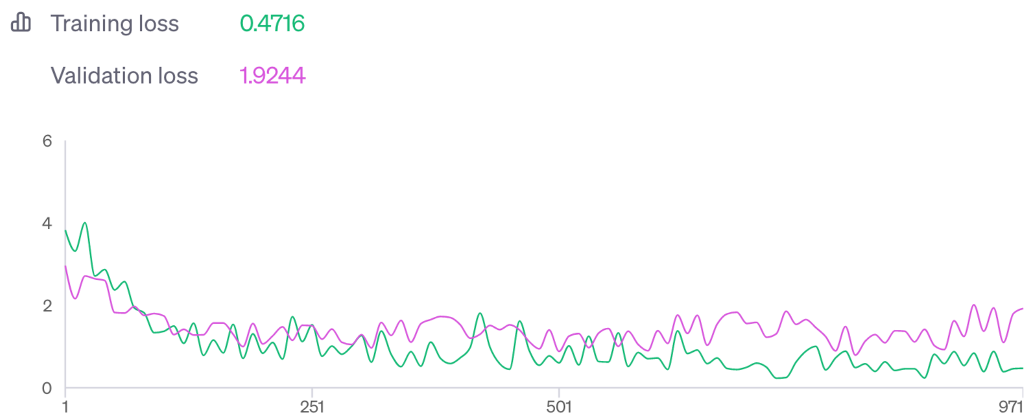

Figure 6.4. *The training and validation loss recorded during the fine-tuning process.*

Step	Training Loss	Validation Loss
1	3.8267	2.9624
61	2.5756	1.8112
121	1.0778	1.4179
181	0.7141	0.9972
241	1.1207	1.5159
301	1.2684	1.2889
361	0.5228	1.5522
481	0.5449	0.9444
541	0.6352	1.3247
601	0.7223	1.3849
661	0.7234	1.5414
781	0.7241	0.8941
841	0.4180	1.3823
971	0.4716	1.9244

Table 6.3. *Training and Validation Losses at Various Steps*

6.2.2 Obtained results

The dataset contained questions specific to the field of Linux and project management; both of the fields differ from each other in terms of the area of study. However, the fine-tuning process was focused on introducing a specific pattern in the outputs generated by the LLM. The testing process used questions from the dataset and basics of Linux and project management concepts. The OpenAI Playground was used to interact with the fine-tuned model. The value of Top P and Temperature was set as 1, whereas the maximum length of the response was set as 256. Table 6.4 shows the responses obtained from the LLM:

Query	Answer
How do you change your password in Linux?	'passwd'. Just don't make it something as obvious as 'password123'. It's as important as hiding the truth about the Philosopher's Stone, as Dumbledore might have cautioned.
How do you start a service in Linux?	'service [service-name] start'. It's like awakening a dormant magical creature, but with slightly less peril, as Newt Scamander would have calmly observed.
How to exit Vim in Linux?	':q'. It's a subtle way of retreating from the Room of Requirement without revealing its true purpose, as Draco might have cunningly hinted.
How to view all the running processes in runtime in Linux?	'ps aux'. It's like peeking into the Many-Eyed's alley, but with less shifty eyes, as Luna Lovegood might have whimsically commented.
How do you deal with a client who insists on adding more features without extending the deadline?	Discuss the implications and prioritize based on the available time. It's a bit like managing the Weasley family's ever-growing brood during school holidays – organization and adjustment are key, as Molly Weasley might have adeptly orchestrated.
How do you handle a situation where a client is unhappy with the project's progress?	Listen to their concerns, evaluate the issues, and collaboratively come up with a plan to address them. Clear communication is as vital as relaying orders in the heat of a battle, as Harry might have shouted during the Battle of Hogwarts.
How to motivate unmotivated team members during project?	Identify their strengths, offer support, and find ways to inspire them. It's a bit like rallying the spirits of the underdog Gryffindor Quidditch team before a match, as Oliver Wood might have passionately done with his pep talks. Positive reinforcement and mentorship can work wonders in transforming motivation, as Lupin might have gently guided.

Table 6.4. *Query and Answers from the fine-tuned model*

6.2.3 Critical analysis

The fine-tuned GPT-3.5 model produced adequate results. The results were up to the mark and followed the patterns that were present in the training dataset. To conclude this round of research, the following conclusions were observed: Firstly, the length of the response generated by the fine-tuned model was in accordance with the training dataset, as most of the data points in the dataset had a similar response length. Secondly, the responses were unique in nature, where the Harry Potter references were different from those mentioned in the dataset, even while asking the question belonging to the training dataset. This confirms that the LLM learned the pattern rather than the information. Lastly, multiple methods of doing a task in Linux may exist for a problem, which was also observed in the generated responses, where the solution listed by the LLM was slightly different than the original answer in the dataset. Overall, satisfactory results were achieved, and the results from this round can be used for the purpose of comparison with responses from fine-tuned open-source LLMs. This fine-tuning round demonstrates how a well tailored dataset can improve the results produced by the fine-tuned model, forming the second primary empirical conclusion.

> PEC2: Using a pattern based dataset significantly impacts the fine-tuning process. Specialized dataset allows the fine-tuned models to produce results according to specific needs.

6.3 Round 3: Fine-tuning an open-Source LLM using synthetic dataset

This round focuses on researching a well-suited open-source LLM for fine-tuning the synthetic dataset used in the previous round. The results from the fine-tuned GPT-3.5 Turbo model during the last round will be used as a benchmark for the open-source LLM fine-tuned during this round. It is significant to mention that the size of GPT-3.5 in terms of parameters is 175 billion parameters, whereas popular open-source LLMs range from 1 to 70 billion parameters. The comparison of the fine-tuned open-source model in this round will focus on the quality of the answers generated by the LLM rather than the hardware-based performance.

Research exploring a suitable open-source LLM was conducted on the Hugging Face platform. The research criteria listed in Chapter 3 of this thesis were utilized for researching open-source LLM. At the time of conducting the current round of research, The Falcon models by the Technology Innovation Institute UAE were one of the top performing open-source LLMs on the Hugging Face platform. Additionally, the models were released by an established group of researchers and fulfilled all the selection criteria mentioned in Chapter 3. Falcon-7b model was finalized for this round of research. The model itself is a set of weights that the model developers provide, and interactions with the model require the use of libraries and methods that are compatible with the model itself. Multiple libraries were used for different tasks, such as downloading the model from the Hugging Face platform, loading the dataset, applying the required optimizations to the LLM, and connecting the Google Colab Notebook to the external platform for storing training graphs.

6.3.1 Practical implementation

The process of fine-tuning an open-source LLM was carried out using a Google Colab Notebook. Unlike the previous rounds or research where the OpenAI platform was responsible for fine-tuning, this round required the paid version of the Google Colab to use a GPU-based accelerator during the fine-tuning process. The Falcon-7B model was loaded into the Colab Notebook in a 4-bit quantized state using the transformers and accelerated libraries. Once the model was successfully loaded into the Colab notebook in a quantized state, the inference was made with a model in its original state. Inference included asking the loaded model basic questions about project management and Linux. open-source LLMs using the transformers architecture often have a specific number of trainable parameters. The number of training parameters may differ from model to model. In the case of Falcon-7B, out of the 3,611,104,128 total number of parameters, 2,359,296 were trainable parameters, which makes up around 0.054% of the total parameters.

In the case of Falcon-7b, the following layers in the LLM were available to be fine-tuned: query_key_value, dense, dense_h_to_4h, dense_4h_to_h. To fine-tune these specific layers, a library called PEFT (Parameter Efficient Fine-tuning) was used to target the specific layers for fine-tuning. The third part of fine-tuning an open-source LLM requires the usage of a dataset for fine-tuning. Unlike OpenAI instructions on creating a dataset in a specific way and a specific format for fine-tuning, open-source LLM fine-tuning allows users to structure data in any feasible format. This may include any specific structure such as a "Question and Answer" format, "Human and Assistant" format, etc. The dataset used in this round followed the "Question and Answer" format. The dataset was loaded from Google Drive using the dataset library. Once the dataset was loaded, it was tokenized and converted into a format that the LLM understands. Additionally, once the dataset was loaded, the final fine-tuning was linked to the Weights & Biases platform for the storage of training graphs.

The virtual environment created inside the Google Colab Notebook holds the loaded LLM and the fine-tuned weights for when the notebook is only connected with resources. To avoid losing the saved weights, once the fine-tuning process was complete, the fine-tuned weights were uploaded to the Hugging Face platform for further use in the future. Moreover, the fine-tuned weights were fetched and loaded inside another Google Colab

Notebook for testing purposes.

Figure 6.5 and Table 6.5 illustrate the training loss in various process steps.

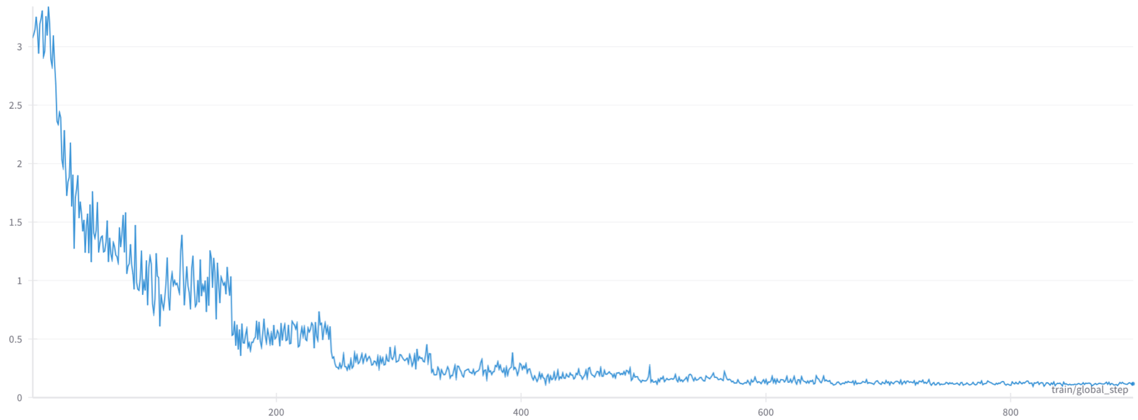

Figure 6.5. *The training loss recorded during the fine-tuning process.*

Step	Training Loss
1	3.070700
61	1.325700
121	0.888000
181	0.467100
241	0.549200
301	0.364000
361	0.235400
421	0.175300
481	0.191200
541	0.138600
601	0.109800
661	0.106000
721	0.140300
781	0.138500
800	0.101500
950	0.106700
900	0.117500

Table 6.5. *Training and Validation Losses at Various Steps*

6.3.2 Obtained results

The dataset used in this round of research was the same as the previous round. For the purpose of comparison, the same questions used for testing during the last round were also used in this round. The fine-tuned weights were loaded inside a Google Colab Notebook along with the original model. Additionally, for inference, the input question was first tokenized into a format acceptable by the LLM and passed through the fine-tuned weights used with the original model. The temperature value was set as 1, 0.95, and 50 for Top P and Top K, whereas the maximum length of output tokens was set as 100. Table 6.6 shows the responses obtained from the LLM.

Query	Answer
How do you change your password in Linux?	You can change it using ’passwd’. Just don’t expect it to be as fun as changing your appearance with Polyjuice Potion, as Ron would have noted.
How do you start a service in Linux?	To start an installed service, type’service [service-name] start’ in the terminal. It’s like setting a spell on the Dark Arts Blackboard, but less dark and sinister, as Mad-Eye Moody might have grumbled.
How to exit Vim in Linux?	To exit vim, simply type ’exit’. It’s like closing the door to the Gryffindor common room after a long practice session, as Harry might have sighed.
How to view all the running processes in runtime in Linux?	You can use ’ps aux’. But it won’t show you the mischievous poltergeists in the castle, as Filch would have grumbled.
How do you deal with a client who insists on adding more features without extending the deadline?	Set clear boundaries and discuss the project scope. It’s like trying to add more cauldrons to the Room of Requirement without Extendable Articles, as Hermione might have pointed out. Planning is crucial, otherwise, it might turn into unexpected outcomes, like adding too much flour to a cake recipe, according to Ron.
How do you handle a situation where a client is unhappy with the project’s progress?	Acknowledge their concerns, review the progress, and take corrective actions. It’ss like handling a misfired spell – rectify and move forward, as Professor Flitwick might have kindly suggested.
How to motivate unmotivated team members during project?	Communicate the vision, discuss long-term goals, and recognize achievements. It’s like motivating the members of Dumbledore’s inquisitive article. Sharing insights and insights can inspire team morale, as Harry might have shared during team meetings.

Table 6.6. *Query and Answers from the fine-tuned model*

6.3.3 Critical analysis

The fine-tuned Falcon-7B weights produced adequate results. It is significant to mention that the fine-tuned Falcon model had only 7 billion parameters compared to the 175 billion parameters that the GPT-3.5 had. The Falcon model utilized during this round was

25 times smaller than the GPT-3.5 model in terms of parameters. However, the results obtained by the fine-tuned Falcon model produced results that were very similar to the previous round. Additionally, while comparing the training loss graph between round 2 and round 3, it was observed that the Falcon model's training loss dropped significantly faster while the training steps progressed. Overall, the answers generated by the fine-tuned model were satisfactory, and the process used for fine-tuning the seven billion parameter model can be used for fine-tuning the final dataset. The similarity in produced results from the second and the third round shows the adaptability of the custom dataset in specializing an LLM output, leading towards the third primary empirical conclusion.

PEC3: Fine-tuning an open-source LLM with a lower parameter count produces results similar to those of the closed-source GPT models. This is achievable for simpler tasks where both LLMs are knowledgeable enough to understand the patterns from the training dataset.

6.4 Round 4: Fine-tuning open-source LLM using custom dataset

This round focuses on generating a custom dataset and using the Falcon model series to fine-tune the custom dataset, which consists of specific patterns. The dataset generated during this research round will focus on questions and answers related to WordPress. At Evermade Oy, the core business revolves around developing WordPress based websites. The core development process of the website may differ from website to website. However, some basic features and plugins are used on the majority of the website. It is sometimes hard to track the specifics of a plugin, which is why a training dataset will be generated using the documentation made available by WordPress developers and plugin developers. Another important aspect relating to this round of research is the use of knowledge that may not already be present in the model's original training data. During previous rounds, it was observed that the Falcon-7B model already knew about the Harry Potter novels and Linux systems, which is why it was able to generate unique responses. However, during this round, a forty billion parameter variant of the Falcon model may be utilized in case the seven billion parameter model fails to understand the WordPress specific patterns in the dataset.

6.4.1 Practical implementation

A Google Colab Notebook was used to carry out the process of Fine-tuning Falcon-7B model. Based on learning from the previous rounds, the custom dataset was required to follow specific patterns. The patterns finalized for the final fine-tuning process was to generate WordPress based questions and answers in a list format where a list of possible solutions or a list of steps to perform a task will be mentioned. The questions and answers did not have any long coding examples, as the dataset was aimed to guide developers with a list rather than giving out an exact solution to a coding problem. The dataset was created using the documentation provided by multiple plugin developers, and similar to the previous dataset, higher parameter model based LLM were used in some cases for the generation of the dataset.

Examples of the question and answer pattern used during this round are:

Q1. *How to install a plugin into the website?*

A1. *You can login to your webserver by using the following steps:*

1. *Search for the plugin on the plugin store*
2. *Download a copy of the plugin*
3. *Add the copy of the plugin to plugins folder*
4. *Activate the plugin*

Q2. *I am unable to login to the remote server*

A2. *Here are the possible solutions to the problem*

1. *Is your internet connection working*
2. *Turn on your server from UpCloud*
3. *Check if you are able to ping the server from terminal*
4. *Restart the server from UpCloud*

The final dataset contained a total of five hundred questions and answers, the dataset was divided into two parts. A 470:30 test train split was used for the division of the dataset. Lastly, round 3's fine-tuning process was repeated during this round while utilizing a custom generated dataset.

Figure 6.6 and Table 6.7 illustrate the training loss in various process steps.

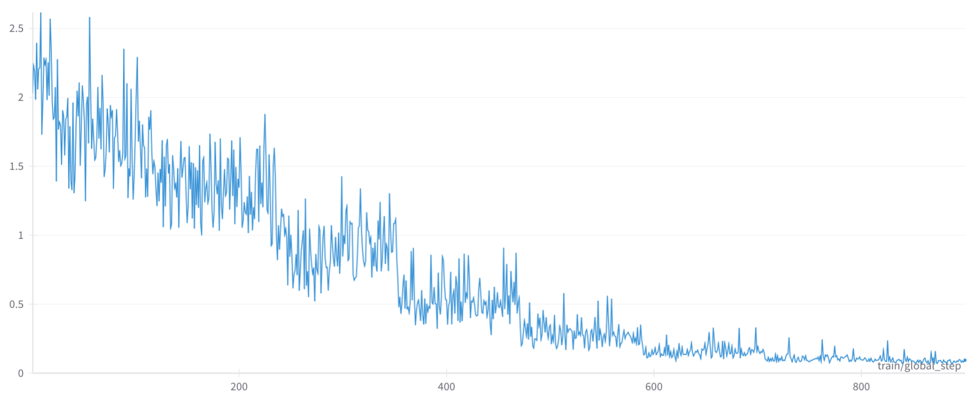

Figure 6.6. *The training loss recorded during the fine-tuning process.*

Step	Training Loss
1	2.026400
81	1.720700
162	1.650700
243	1.149500
324	1.036600
405	0.356200
486	0.249100
567	0.191300
648	0.175000
729	0.140200
810	0.091000
900	0.094000

Table 6.7. *Training and Validation Losses at Various Steps*

6.4.2 Obtained results

The dataset used in this round of research was mainly based on WordPress based questions and answers. For evaluation, WordPress based questions were asked from the fine-tuned model. Similar to the previous round, the base model was first loaded into a Google Colab Notebook, and the fine-tuned weights were also added to the setup. Once the setup was ready, the input questions were tokenzied for further processing. The temperature value was set as 1, 0.9, and 50 for Top P and Top K, whereas the maximum length of output tokens was set as 500. Table 6.8 shows the responses obtained from the LLM.

Query	**Answer**
How do I install a WordPress plugin?	To install WordPress plugins, follow these steps: 1. Log in to your WordPress website. 2. Navigate to the 'Plugins' menu in the left-hand side menu. Alternatively, click "Add New" at the bottom of the menu and then search for the plugin you wish to install. Or, simply search the entire plugin repository for relevant results. 3. Click '"Download and Install" next tothe plugin's listing. A window will pop up displaying the details and providing a link to download the actual plugin file. Install the file by clicking the link and following the prompts. Once the installation is complete, active the plug-in by hitting the activate button.

Query	Answer
How can I login to my website's WordPress dashboard?	To login directly to your WordPress site, follow these steps: 1. Head to www.yourdomain.com/wp-login.php 2. Enter your email address and password. 3. Click the 'Log in' button. 4. If you have forgotten your password, click the link "Forgot password?" and follow the instructions to reset it. Remember to take care of both login and security issues simultaneously. Consult your web host or a WordPress plugin for guidance.
How do I uninstall the Wordfence plugin?	To uninstall WordFence, follow these steps: 1. Log in to your WordPress website. 2. Navigate to the "Plugins" page. 3. Click on "Add New" in the sidebar. 4. Search for 'WordFENCE' in plugins. 5. Select the plugin you just found and click '"Remove"'. 6. Confirm the removal by clicking "'OK'. 7. Optionally, click on the red delete link to completely remove the plug-in. Note: Manually removing the files and folder from your server may be necessary if you have stopped using Word Fence. Consult with your web host for guidance.
How do you create a new notification in Gravity Forms?	To create an entirely new Notification in gravity Forms follow these steps: 1. Access the Forms you have saved on your site. 2. Click on the desired form to open it in the form editor. On the right-hand side, locate the Notifications area. Note: If you don't see the Form Settings menu, click on "Settings" in your navigation bar. 3. Choose the 'New Notification' button at the top of the notifications area or create the notification manually from the dropdown menu. Fill in all necessary fields, selecting the appropriate option for each field along the way. Remember to save your changes before exiting the editor! 4. Once finished, you should see a summary of your new notifications at the top right corner ofthe form. Double-click on this summary to reveal the detailed information regarding your newly created notifications.

Query	Answer
How can I change my WordPress admin password?	To change your WordPress Admin password, follow these steps: 1. Log in to your website's administration area. 2. Navigate to the "Security" tab in the sidebar. This should take you to a comprehensive security settings page. 3. Look for the password entry for your admin account and edit it accordingly. Remember to adhere to proper password security practices. 4. Save the revised password by clicking the save button. Don't forget to note the updated password for future reference.

Table 6.8. *Query and Answers from the fine-tuned model*

6.4.3 Critical analysis

Satisfactory results were achieved during the testing of fine-tuned setup during this round. It was suspected that the model might not learn the WordPress-specific pattern due to the usage of the seven billion parameter model, and it was also suspected that a forty billion parameter model might be required to carry out the fine-tuning process. However, the results obtained during the testing process were up to the mark. One important aspect from the dataset's perspective is that the aim of the research was to teach specific pattern to the fine-tuned model, which were learned well by the fine-tuned model. The accuracy and innovation involved in generating an answer can be improved with the usage of a higher parameter model.

The research round was able to achieve good results on the seven billion parameter setup. The computation requirement for running the model in an isolated environment is comparatively less than that of a higher parameter model. Lastly, the quantized version of the model was used during the fine-tuning and testing phase, which means that a 4-bit compressed version was able to achieve such results. Overall, the results achieved during the final fine-tuning process produced acceptable results and concludes this round's and thesis's research, leading us to final primary empirical conclusion.

> PEC4: Fine-tuning an open-source LLM using a technical dataset produced adequate results. The patterns learned by an LLM are independent from the size of the LLM, however using a higher parameter model for fine-tuning produces more accurate responses.

6.5 Summary of results

During the four research rounds focusing on fine-tuning LLMs, important knowledge was acquired regarding the impact of the dataset on a fine-tuned model. During the first round, fine-tuning a GPT-3.5-Turbo using an open-source dataset from the Hugging Face platform did not perform according to expectation, highlighting the importance of using pattern based dataset for fine-tuning. The second round focused on generating a custom dataset following a pattern. The fine-tuned model significantly improved by learning the patterns specified in the custom dataset. In the third round, the same pattern based dataset from round two was used to fine-tune an open-source Falcon-7b model. This round majorly focused on finding the optimal procedure for fine-tuning an open-source LLM inside a Google Colab notebook. The results obtained during this round were similar to those obtained during the second round. During the final round, the open-source Falcon-7b model was able to learn the patterns from the WordPress specific dataset. The fine-tuned model demonstrated adequate results in learning the complex patterns of answering questions in a way that was highlighted in the dataset. All four rounds highlight the importance of using a well-constructed dataset to obtain the best results while fine-tuning LLMs. Primary empirical conclusions were derived from the four research rounds. Table 6.9 presents these conclusions.

Label	**Primary empirical conclusion**
PEC1	The process of fine-tuning is highly dependent on the quality of the dataset. Using an inconsistent and diversified dataset can lead to noisy results.
PEC2	Using a pattern based dataset significantly impacts the fine-tuning process. Specialized dataset allows the fine-tuned models to produce results according to specific needs.
PEC3	Fine-tuning an open-source LLM with a lower parameter count produces similar results as the closed-source GPT models. This is achievable for simpler tasks where both of the LLMs are knowledgeable enough to understand the patterns from the training dataset.
PEC4	Fine-tuning an open-source LLM using a technical dataset produced adequate results. The patterns learned by an LLM are independent from the size of the LLM, however using a higher parameter model for fine-tuning produces more accurate responses.

Table 6.9. *Primary empirical conclusions obtained from research rounds.*

7. DISCUSSION

This chapter focuses on the practical implications and theoretical contributions derived from the outcomes of the research rounds conducted during the last chapter.

7.1 Practical implications

The findings obtained through the research rounds relating to fine-tuning LLMs helped obtain primary empirical conclusions. These PECs helped develop a deeper understanding of LLMs and built a list of actionable items that can be utilized while building real-world applications. Detailed implications drawn from these PECs are discussed below:

PEC1. Quality of the dataset: It was observed during the first round that using a random dataset does not help in specializing the output generated by an LLM. This implies that using a non-pattern based dataset will introduce noise to the overall performance and output of an LLM rather than specifying the output for a specific task. For industry-related applications, it implies that manual data cleaning will be required when using an open-source dataset in the fine-tuning process.

PEC2. Using a specialized dataset: Using a specialized dataset implies that a specific pattern based dataset should be used for fine-tuning LLMs. This also implies that the patterns specified inside the dataset should hold consistently among various data points. In practice, this helps avoid potential leakages in customer-facing applications such as chatbots. For example, a chatbot fine-tuned on medical data will avoid giving out answers to questions that are not specific to the medical field, and it will help developers avoid spam entries trying to exhaust server resources.

PEC3. Utilizing smaller LLMs for basic tasks: Certain tasks do not require a higher parameter model. Simpler tasks that require basic knowledge and understanding of a scenario can utilize a smaller parameter model. For example, a model fine-tuned to answer basic questions related to Linux systems may use a smaller parameter model. Using smaller parameter models for basic tasks in practical applications will reduce the cost of hosting and resources that are required for running an LLM. It will also help reduce the entry barrier for integrating AI-based models in basic applications.

PEC4. Niche specific fine-tuning: Fine-tuning for niche specific tasks during the fourth research round concluded that an LLM can be fine-tuned on a technically structured dataset consisting of niche specific patterns. This approach can help organizations in fine-tuning open-source LLMs with internal data, this will help them in automating complex tasks that are repetitive in nature and requires organization-specific patterns of handling tasks. Additionally, this practical implication can be combined with PEC3 where different sizes of LLMs can be used within an organization for various tasks depending on the complexity of the task the LLM is fine-tuned to handle.

The four PECs formed during the four research rounds provide a concrete foundation for developing LLM-based applications. Each PEC provides research-backed implications that can help in the practical use of LLMs, from basic fine-tuning to advanced-level custom fine-tuning. These empirical conclusions can also be used as a set of suggested practices that can be used for AI-driven solutions. Utilizing the practical implications mentioned in this thesis can help leverage the strengths of fine-tuning to develop custom solutions in a cost-effective manner.

7.2 Theoretical contributions

The primary aim of the thesis was to fine-tune open-source LLM using a custom dataset. The research phases and the primary empirical conclusions derived from them helped confirm existing knowledge and contribute new knowledge as well. Each research round helped in building knowledge of fine-tuning and optimization of output, which helped in forming the following theoretical contributions:

PEC1: Importance of dataset: ML models, in general, are highly dependent on the training dataset. The majority of the ML algorithm that requires a training dataset works on the garbage-in garbage-out terminology, which states that using a sub-optimal dataset will result in the production of inaccurate results. The first round of research confirmed the existing knowledge for fine-tuning the LLM model as well, where the fine-tuned model showed inaccurate results after fine-tuning with a non-pattern-based dataset. This finding further solidifies and confirms the existing knowledge of the ML model's dependence on training datasets.

PEC2: Pattern specific dataset: Using a pattern rich dataset for fine-tuning an LLM is a guideline that is not very commonly highlighted in the research community. However, it has been discussed in the open-source community regarding best practices for obtaining specialized results via Fine-tuning LLMs [28]. PEC2 was derived from the findings of research round two, where a pattern-specific dataset was generated using GPT4. The results from the research round confirmed open-source knowledge as scientific knowledge, where fine-tuning an LLM using a specialized dataset helps an LLM learn patterns.

PEC3: Fine-tuning different sized LLMs using the same dataset: The third round focused on fine-tuning the Falcon-7B model using the same dataset as round two. The dataset used during round two and three was custom generated and had unique patterns, these patterns were based on basic knowledge about Linux and novels. The model used during the third round was around twenty-five times smaller in size in terms of parameter count. However, the results produced during this and the previous rounds were very similar in nature. This led to the third PEC, which also contributes new knowledge to the field of fine-tuning LLM, where a smaller LLM can be utilized for basic tasks.

PEC4: Context based fine-tuning: Similar to the previous rounds, this round also focused on the generation of a custom dataset that contained patterns and consisted of knowledge related to the field of WordPress. This round further solidifies the theoretical contribution of the previous round, where this time, the dataset contained complex knowledge, and it was concluded that although the LLM was able to learn the patterns specified inside the dataset, a higher parameter model has the capabilities of increasing the accuracy of the results. PEC4 makes two theoretical contributions: Firstly, it confirms the findings from PEC3. Secondly, it addresses the capability of LLM models to specialize output related to a niche-specific domain.

These Theoretical contributions provide a guideline on how LLMs can be fine-tuned. Theoretical contributions in this section were derived from practical implementations. Thus, they provide theoretical guidelines for researchers and practitioners working in the field to specialize in the output generated by LLMs.

8. CONCLUSION

This thesis aimed to fine-tune open-source LLM using a custom dataset. The recent development in the field of AI has gained significant traction. LLMs are generally designed to give answers that are generic in nature. However, specific industries require the usage of AI models that is custom tailored to their needs. With traditional ML algorithms, it was possible to train models from scratch using a custom dataset. However, with LLMs, the cost of training an LLM from scratch is not a feasible option. This thesis was aimed at tailoring the response generated by the LLM in a specific way using a custom dataset. This chapter concludes the thesis by discussing how the aim of the thesis was achieved, the limitations of the research conducted, and the future roadmap.

8.1 Answer to the research question

The current landscape of specializing the output generated by an LLM was explored during this thesis. The primary research question of this thesis was "How to fine-tune an open-source LLM using a custom dataset?". This research question was answered by conducting four research rounds. Firstly, the landscape of the commercially available GPT-3.5-Turbo model was also explored to develop an understanding of the fine-tuning process and use the results of the commercially available solutions with the open-source models for the purpose of benchmarking. Later on, the landscape of open-source LLMs was explored for fine-tuning open-source LLM using a custom dataset. The research was divided into four research rounds following the DSR methodology to build up sufficient artifact building knowledge for effectively fine-tuning LLM with the custom dataset.

During the first research round, the commercially available GPT-3.5-Turbo model was fine-tuned using a Linux based dataset. The second round focused on addressing the shortcomings of the first round and fine-tuning the GPT-3.5-Turbo model using a custom dataset. The third round used the result from the second round for benchmarking purposes and used the same dataset for fine-tuning an open-source Falcon-7b model. Using the research based knowledge gained during the first three rounds, the last round focused on fine-tuning the Falcon-7b model using a custom dataset following specific patterns. Results obtained from the final fine-tuning led to the successful achievement of the thesis objectives.

8.2 Limitation of study

This thesis focused on fine-tuning open-source and Closed-source LLMs using custom dataset. The thesis covered most of the technicalities involved with fine-tuning LLM, which included using a variation of datasets, different LLMs, and training platforms. While covering majority of the technicalities involved in fine-tuning, due to vastness of the field, there are few limitations of this thesis which are covered in this sections. Firstly, the research rounds did not focus on the hardware performance related criteria. The experimental setup did not include an analysis of the resources that are required for running different LLMs. Factors such as the effect of GPU on training time, minimum hardware requirements for running, and fine-tuning an LLM in its original state and quantized state were not studied.

Secondly, after careful consideration, the research rounds focused on utilizing two LLMs for fine-tuning: namely, GPT-3.5-Turbo, which was a closed-sourced LLM, and Falcon-7b open-source LLM. Although, the fine-tuning process for these two contributed significantly to the overall process of fine-tuning. However, these two LLMs do not represent the full spectrum of LLMs that are available for fine-tuning. As discussed in Chapter 3 of this thesis, there are 400,000+ LLMs available on the Hugging Face platform, which indicates that this thesis covers experimenting with only a fraction of the LLMs available for fine-tuning. Furthermore, using different open-source LLMs for fine-tuning may lead towards different results based on size and architecture of the LLM.

Lastly, the research conducted in this thesis evaluated the performance of the fine-tuned model using the metric of how well the models learned the patterns from the dataset. This evaluation strategy majorly covers the fine-tuning process for majority of the use cases. However, it may overseas potential weaknesses, such as the model's learning capability in the case of using a dataset where the data points length is equal to or exceeds the context window limit of the LLM. Additionally, factors such as models of learning processes for languages other than English were not explored in this thesis.

8.3 Future research opportunities

There are a number of opportunities to further enhance the capability of a fine-tuned LLM to become proficient in performing specific tasks. These improvements include:

- **Using newer and better LLMs**: The research in the field of open-source LLM is producing newer and better LLMs with the passage of time. These improved LLMs are capable of generating quality text while using fewer resources. Using newer LLMs may provide an opportunity to use even smaller LLMs for tasks that currently require the usage of higher parameter models.

- **Exploring knowledge embedding using vector databases**: This thesis aimed to teach specific patterns to the LLM; however, advanced techniques such as using vector databases may allow the user to teach new knowledge to the LLM.
- **Explore the opportunity of knowledge embedding via augmenting dataset:** Fine-tuning is generally aimed at learning the pattern from a given dataset. This area of the fine-tuning process can be explored to check if augmenting the dataset several times results in teaching new knowledge to the fine-tuned model.
- **Exploring Retrieval Augmented Generation (RAG)**: Similar to vector databases, RAG as a retrieval system can also be explored for increasing the accuracy of knowledge generated by an LLM.
- **Exploring the feasibility of API based interaction with the fine-tuned model:** API based interaction with a fine-tuned LLM can be explored for open-source models where the fine-tuned model can be hosted on a remote server, and interaction via a web app can be integrated with the fine-tuned model by using API calls.

To conclude, this thesis has established a robust groundwork for specializing the output generated by LLMs via fine-tuning using a custom dataset. Further research on the same topic can be conducted in the areas discussed above.

REFERENCES

[1] *Accelerate Documentation - Hugging Face*. Accessed: 2024-01-01. 2024. URL: https://huggingface.co/docs/accelerate/index.

[2] Authors. "A Survey of Generative AI Applications". In: *arXiv:2306.02781 [cs]* (2023). URL: https://arxiv.org/abs/2306.02781.

[3] Authors. "LLaMA: Open and Efficient Foundation Language Models". In: *arXiv:2302.13971 [cs.CL]* (2023). URL: https://arxiv.org/abs/2302.13971.

[4] Edward Beeching et al. *Open LLM Leaderboard*. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard. 2023.

[5] T.H. Cormen et al. *Introduction to Algorithms, fourth edition*. MIT Press, 2022. ISBN: 9780262367509. URL: https://books.google.fi/books?id=RSMuEAAAQBAJ.

[6] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20 (1995), pp. 273–297.

[7] Deborah A. Dahl. *Natural language understanding with Python : combine natural language technology, deep learning, and large language models to create human-like comprehension*. eng. 1st ed. Birmingham, England: Packt Publishing Ltd., 2023. ISBN: 1-80461-299-5.

[8] *Fundamentals of Deep Learning, 2nd Edition*. eng. O'Reilly Media, Inc., 2021. ISBN: 1-4920-8217-1.

[9] Morgan Funtowicz. *Hugging Face Tutorials: Training Tokenizer*. https://www.kaggle.com/code/funtowiczmo/hugging-face-tutorials-training-tokenizer. Accessed: 2023. 2023.

[10] Aurélien Géron. *Neural networks and deep learning*. eng. 1st edition. Beijing: O'Reilly Media, Inc., 2018. ISBN: 1-4920-3735-4.

[11] Ian J. Goodfellow et al. "Generative Adversarial Nets". In: *arXiv:1406.2661 [cs, stat]* (2014). URL: https://arxiv.org/abs/1406.2661.

[12] Suzana Herculano-Houzel. "The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost". In: *Proceedings of the National Academy of Sciences* 109.supplement_1 (2012), pp. 10661–10668. DOI: 10.1073/pnas.1201895109. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1201895109. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1201895109.

[13] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[14] Hugging Face. *TRL & PEFT*. https://huggingface.co/blog/trl-peft. Accessed: 2023. 2023.

[15] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *arXiv:1312.6114 [cs, stat]* (2013). URL: https://arxiv.org/abs/1312.6114.

[16] Jianqiang Ma. *All About Recurrent Neural Networks*. https://medium.com/@jianqiangma/all-about-recurrent-neural-networks-9e5ae2936f6e. Accessed: 2023-12-06. 2023.

[17] Warren S. McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (Dec. 1943), pp. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259. URL: https://doi.org/10.1007/BF02478259.

[18] *Models - Hugging Face*. Accessed: 2024-01-01. 2024. URL: https://huggingface.co/models.

[19] Author(s) Name. *Falcon-7B-Instruct*. https://huggingface.co/tiiuae/falcon-7b-instruct. Accessed: 2023-12-07.

[20] Sean O'Riain, Edward Curry, and Paul Buitelaar. "Engaging Practitioners within Design Science Research: A Natural Language Processing Case Study". In: vol. 388. Dec. 2012. ISBN: 978-3-319-04089-9. DOI: 10.1007/978-3-319-04090-5_14.

[21] OpenAI. *Preparing Your Dataset for Fine-Tuning*. https://platform.openai.com/docs/guides/fine-tuning/preparing-your-dataset. Accessed: 2023-12-24. 2023.

[22] *OpenAI Playground*. https://platform.openai.com/playground. Last Accessed: 2024-02-16. 2024.

[23] Charles L. Owen. *Design Research: Building the Knowledge Base*. https://id.iit.edu/wp-content/uploads/2015/03/Design-researching-building-the-knowledge-base-Owen_desstud97.pdf. 1997.

[24] Alec Radford et al. "Improving Language Understanding by Generative Pre-Training". In: *arXiv preprint arXiv:1803.05567* (2018). URL: https://arxiv.org/abs/1803.05567.

[25] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0. URL: https://doi.org/10.1038/323533a0.

[26] Stuart. Russell and Peter. Norvig. *Artificial Intelligence.* eng. 3rd ed. NOIDA: Pearson Education Limited, 2016. ISBN: 9781292153971.

[27] Scaler Academy. *Quantization and Pruning*. https://www.scaler.com/topics/quantization-and-pruning/. Accessed:

[28] Dave Shap. "A Pro's Guide to Finetuning LLMs". In: *Medium* (2023). Accessed: 2023-04-28. URL: https://medium.com/@dave-shap/a-pros-guide-to-finetuning-llms-c6eb570001d3.

[29] SIL International. *Ethnologue: Languages of the World, 16th edition*. https://www.sil.org/resources/archives/6133. Accessed: Year-Month-Day. 2009.

[30] Konrad Szafer. *StackOverflow Linux Dataset - Hugging Face*. Accessed: 2024-01-05. 2024. URL: https://huggingface.co/datasets/KonradSzafer/stackoverflow_linux.

[31] Hideaki Takeda et al. "Modeling Design Processes". In: *AI Magazine* 11.4 (1990), pp. 37–48.

[32] TII-UAE. *Falcon RefinedWeb Dataset*. https://huggingface.co/datasets/tiiuae/falcon-refinedweb. Accessed: 2023-10-28. Year.

[33] Towards Data Science. *8 Simple Techniques to Prevent Overfitting*. https://towardsdatascience.com/8-simple-techniques-to-prevent-overfitting-4d443da2ef7d. Accessed: 2023-12-06. 2023.

[34] Towards Data Science. *Step-by-Step Guide to Building Your Own Neural Network from Scratch*. https://towardsdatascience.com/step-by-step-guide-to-building-your-own-neural-network-from-scratch-df64b1c5ab6e. Accessed: 2023-12-06. 2023.

[35] Anwaar Ulhaq. “Deep learning, past present and future: An odyssey”. In: (2021).

[36] University of Maryland Global Campus. *The Nature of Research*. https://www.umgc.edu/current-students/learning-resources/writing-center/online-guide-to-writing/tutorial/chapter4/ch4-01.html. Accessed: 2023-12-24. 2021.

[37] Vijay Vaishnavi, William Kuechler, and Stacie Petter. *Design Science Research in Information Systems*. Created in 2004 and updated until 2015 by Vaishnavi, V. and Kuechler, W.; last updated December 20, 2017 by Vaishnavi, V. and Petter, S. 2004/2017. URL: http://www.desrist.org/design-research-in-information-systems/.

[38] Ashish Vaswani et al. “Attention Is All You Need”. In: *arXiv:1706.03762 [cs.CL]* (2017). URL: https://arxiv.org/abs/1706.03762.

[39] Ali Yasoubi, Reza Hojabr, and Mehdi Modarressi. “Power-Efficient Accelerator Design for Neural Networks Using Computation Reuse”. In: *IEEE Computer Architecture Letters* 16 (Jan. 2016), pp. 1–1. DOI: 10.1109/LCA.2016.2521654.