Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

LARGE LANGUAGE MODELS FOR EDUCATION

Relatore:
Chiar.mo Prof.
SAVERIO GIALLORENZO

Correlatore:
Chiar.mo Prof.
MAURIZIO GABBRIELLI

Presentata da:
BIANCA RAIMONDI

Sessione II
Anno Accademico 2022/2023

E, andandomene via, dovetti concludere meco stesso che veramente di cotest'uomo ero piú sapiente io: in questo senso, che l'uno e l'altro di noi due poteva pur darsi non sapesse niente né di buono né di bello; ma costui credeva sapere e non sapeva, io invece, come non sapevo, neanche credevo sapere; e mi parve insomma che almeno per una piccola cosa io fossi piú sapiente di lui, per questa che io, quel che non so, neanche credo saperlo.

(Platone, Apologia di Socrate)

Abstract

This thesis explores the use of large language models (LLMs) in specialized domains, with a particular focus on educational content found in LaTeX-formatted textbooks. The introduction highlights the importance of fine-tuning LLMs for domain-specific tasks and emphasizes the significance of preprocessing data from specialized sources like textbooks.
One of the key objectives is to demonstrate how fine-tuned LLMs, tailored to educational content, can effectively address the limitations of LLMs pre-trained on general text, especially in responding to single-choice questions. Additionally, the thesis examines the practical aspects of deploying LLMs, comparing the resource consumption of large pretrained models to smaller fine-tuned ones, offering insights into performance-efficiency trade-offs.
In summary, this thesis aims to contribute to the field of natural language processing by exploring the adaptation of LLMs to educational content and addressing their limitations. The research is structured into three chapters, each focusing on distinct aspects, and concludes with reflections on future directions in this evolving field.

Introduction

In the ever-evolving landscape of natural language processing, the quest to enhance the capabilities of large language models (LLMs) has led to groundbreaking advancements in various applications, from chatbots to text generation. One compelling avenue of research is the adaptation of LLMs to specialized domains, such as educational content found in LaTeX-formatted textbooks. This thesis endeavors to delve into this intriguing realm, shedding light on the utilization of LaTeX-formatted textbook data for fine-tuning LLMs. This research underscores the critical importance of domain-specific data preprocessing when confronted with the intricacies of specialized content like textbooks, with their unique formatting and structures.

One of the primary objectives of this work is to showcase the adaptability of fine-tuned LLMs, specifically tailored to the nuances of educational content. This adaptation addresses a substantial limitation in the capabilities of LLMs pretrained on general text, which often struggle to provide accurate responses to single-choice questions. By fine-tuning LLMs with domain-specific data, this research aims to demonstrate significant improvements in their ability to comprehend and respond to such questions effectively.

Furthermore, this thesis delves into the practical considerations surrounding the deployment of fine-tuned LLMs, with a focus on resource consumption. It compares the resource-intensive nature of employing a large, pretrained LLM versus the more efficient utilization of a fine-tuned, smaller LLM. This exploration provides valuable insights into the trade-offs between model size and computational cost, offering guidance for researchers and practitioners seeking to strike a balance between performance and efficiency in specialized NLP applications.

In summary, this thesis embarks on a multifaceted journey that encompasses the fine-tuning of LLMs using LaTeX-formatted textbook data, the optimization of their performance in addressing single-choice questions, and a critical evaluation of resource consumption. Through these investigations, we aim to contribute valuable knowledge to the field of natural language processing, fostering a deeper understanding of how LLMs can be harnessed to excel in

the domain of educational content and beyond.
This thesis is structured into three distinct chapters, each addressing a crucial aspect of the research work. The first chapter 1 introduces Large Language Models (LLMs), elucidating their core concepts, utilization in various applications, and the process of fine-tuning to adapt them to specific tasks.
Moving forward, the second chapter 2 delves into the historical evolution and existing literature on the application of LLMs in the domain of education. This chapter provides valuable context for the subsequent experiments by tracing the evolution of LLMs in educational contexts and highlighting relevant prior work.
The third and pivotal chapter 3 details the fine-tuning experiment conducted in this research. It encompasses the methodology, dataset, and results, showcasing how fine-tuned LLMs enhance their performance in addressing single-choice questions. Additionally, this chapter presents a rigorous cost-consuming comparison, shedding light on the resource efficiency of smaller fine-tuned LLMs compared to larger, pretrained counterparts.
This well-structured thesis aims to comprehensively explore the potential of fine-tuned LLMs in educational content and contribute valuable insights to both the NLP and education communities.

Contents

List of Figures

List of Tables

Chapter 1

Large Language Models

In this chapter, the focus lies on Large Language Models (LLM), their history, and the main techniques employed in present times. The first part will cover Generative AI and Language Modeling, providing essential insights to comprehend the nature of LLMs. Following that, the chapter will introduce the aforementioned techniques.

1.1 Generative AI

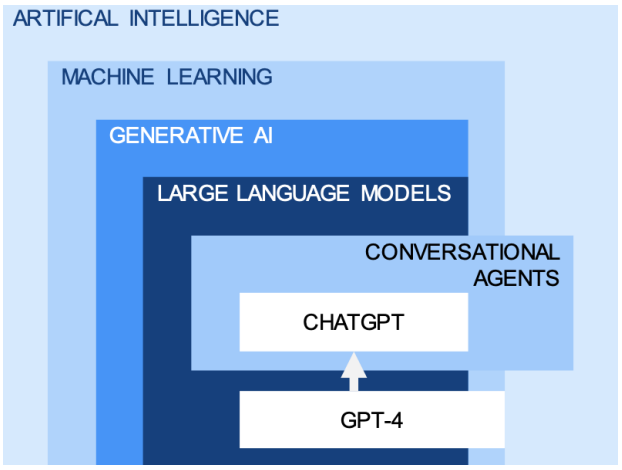

Figure 1.1: Relationship between AI, ML, Generative AI, and LLMs

Generative AI is a type of artificial intelligence that is capable of generating new content or data that resembles existing data or follows a certain pattern. It uses machine learning algorithms to learn from a dataset and then create

new content based on that learning. Figure 1.1 from [1] shows the relationship between Artificial Intelligence, Machine Learning, Generative AI and LLMs in terms of nested sets.
Generative AI can be used in a variety of applications, such as natural language processing, image generation, and music composition. For example, a generative language model could be trained on a large dataset of text and then be used to generate new text that resembles the original dataset. Similarly, a generative image model could be trained on a large dataset of images and then be used to generate new images that resemble the original dataset.

1.2 Language Modeling

Human beings possess a significant ability to express and communicate through language, which starts developing in early childhood and continues to evolve throughout their lives. However, machines lack the innate capacity to understand and communicate using human language, unless equipped with powerful Artificial Intelligence algorithms. This has been a long-standing research challenge: enabling machines to read, write, and communicate like humans.
Language modeling (LM) is a major approach used to enhance the language intelligence of machines. Generally, LM focuses on modeling the likelihood of generating word sequences to predict the probabilities of future or missing tokens. Over time, LM research has gone through four major development stages:

1. Statistical Language Models (SLM): the fundamental concept of this stage, also known as n-gram language models, involves constructing a word prediction model using the Markov assumption. In other words, it predicts the next word by considering only the most recent context. In figure 1.2 from [2] an example of SLM.

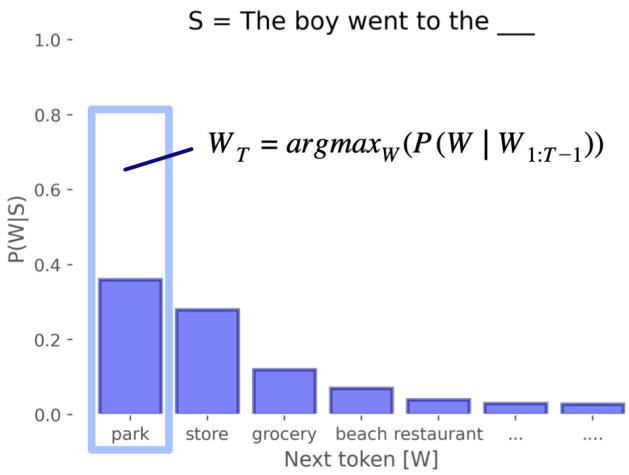

Figure 1.2: An example of Statistical Language Model

2. Neural Language Models (NLM): during this stage, the probability of word sequences is characterized using neural networks, particularly recurrent neural networks (RNNs). These RNNs are utilized to model the likelihood of different word sequences. In figure 1.3 from [3] an example of NLM.

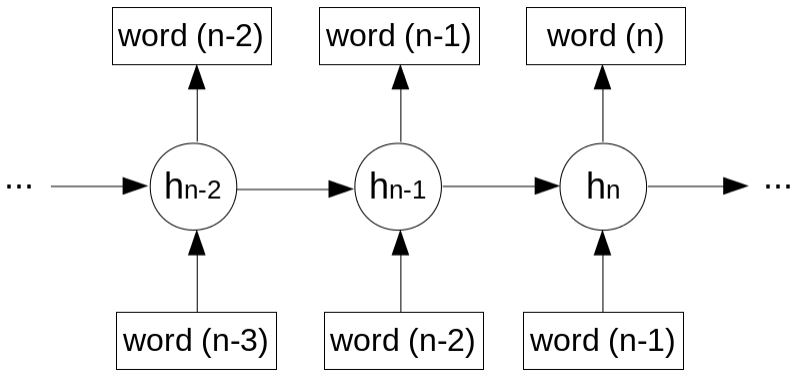

Figure 1.3: An example of Neural Language Model

3. Pre-trained language models (PLM): in this stage, the approach was introduced to capture context-aware word representations by initially pre-training a bidirectional LSTM (biLSTM) network. Instead of learning fixed word representations, the biLSTM network is fine-tuned specifically for downstream tasks to enhance its ability to understand con-

text. In next paragraph will be explained main concepts of these models.

4. Large language models (LLM): the term *Large Language Models (LLMs)* is used by the research community to refer to PLMs that are of significant size in terms of parameters number inside the network. As it will be illustrated in next sections, a remarkable example of LLMs is ChatGPT [4], which leverages the advancements of the GPT series for dialogue applications. ChatGPT showcases an impressive ability to engage in amazing conversations with humans.

Each of these stages has received considerable attention in the literature, but at the moment, LLMs have attracted the attention of a vast area of research due to their capabilities in content generation. In this work, the attention will be focused on LLMs only. Before introducing LLMs, it is necessary to understand what Transformer architecture is and how it is used in LLMs. So the next section introduces Transformer and how it relates to LLMs.

1.3 Transformer

In 2017, Vaswani et al. [5] presented the Transformer architecture, which marked a breakthrough in natural language processing by effectively overcoming the limitations associated with recurrent neural networks (RNNs).

Figure 1.4: RNN reference window

RNNs have a limited context window (as shown in Figure 1.4), which limits their ability to retrieve words produced earlier in the sequence.

Figure 1.5: LSTM reference window

LSTMs have a larger reference window compared to RNN (Fig. 1.5).

Figure 1.6: Transformer reference window

Transformers, on the other hand, have an infinite reference window (Fig. 1.6). This allows them to take full advantage of the context while generating text.
At its core, the Transformer uses a novel attention mechanism that allows it to process input data in parallel, making it highly efficient for sequential tasks. The architecture consists of an encoder and a decoder, each of which consists of several layers. At a high level, the encoder maps an input sequence into an abstract continuous representation that holds all the learned information of that input to the decoder. The decoder then takes these continuous representations and incrementally generates a single output, while also feeding back to previous outputs.
The encoder processes the input sequence by applying self-attention mechanisms to capture dependencies between different words, regardless of their position in the sequence. The self-attention mechanism computes weighted

representations of all words in the sequence, allowing the model to consider the context of each word in relation to all other words. This is particularly useful for understanding the nuanced relationships between words, even if they are far apart in the sequence.

The breakthrough feature of the Transformer architecture, its attention mechanism, is a key element that has facilitated the development of LLMs. LLMs, such as OpenAI's GPT series, use the Transformer architecture to learn and generate human-like text. By pre-training on massive amounts of text data, these models learn complex linguistic patterns and structures. This pre-trained model can then be fine-tuned for specific tasks such as language translation, question answering and even creative writing.

The relationship between the Transformer architecture and LLMs lies in their synergy. LLMs use the powerful attention mechanisms of the Transformer to understand and generate coherent and contextually relevant text. The model's self-attention allows it to capture the nuances of language, such as semantics, syntax and long-range dependencies, and to produce human-like text that is contextually accurate. In addition, LLMs can be primed with prompts to trigger specific types of responses, providing a remarkable degree of control over the content generated.

The innovative attention mechanism of the Transformer architecture, combined with the massive scale of training data and advances in optimisation techniques, has propelled the development of LLMs to the forefront of natural language understanding and generation. These models have demonstrated exceptional performance in a wide range of applications, from automated translation to content creation, and have played a pivotal role in reshaping how machines interact with and produce human language. The next subsection provides a better overview of the architecture.

Transformer Architecture

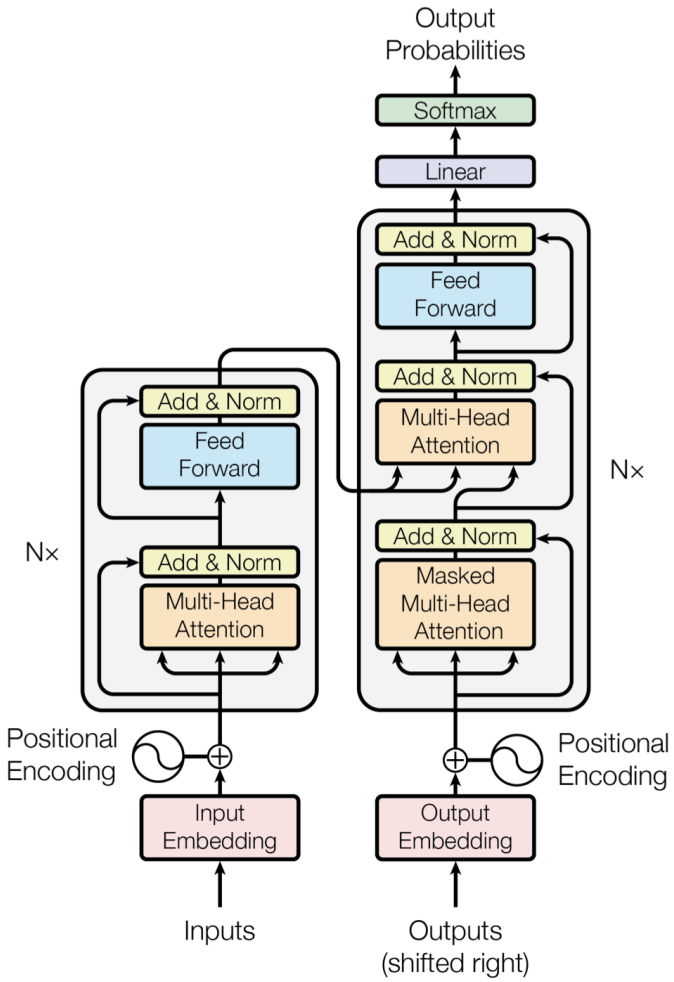

Figure 1.7: Transformer Architecture

The transformer architecture, as shown in Figure 1.7, has a distinctive structure that includes stacked layers of self-attention and pointwise fully connected layers in both its encoder and decoder components, shown on the left and right sides of Figure 1.7, respectively. Within the encoder stack, a crucial aspect is the presence of N = 6 identical layers. Each of these layers consists of two sub-layers. The first sub-layer implements a multi-head self-attention mechanism, while the second sub-layer uses a simple positionally fully connected feed-forward network. Residual connections are integrated around each of these sub-layers, followed by layer normalisation. Similarly, the decoder stack, which mirrors the structure of the encoder, also consists

of N = 6 identical layers. In addition to the two sub-layers present in each encoder layer, the decoder introduces a third sub-layer. This third sub-layer performs multi-head attention on the output generated by the encoder stack. As in the encoder, residual connections are included around each sub-layer, followed by layer normalisation. A notable modification to the decoder's self-attention sub-layer is to prevent positions from attending to subsequent positions, effectively creating a masking effect. This, combined with the property of shifting output embeddings by one position, ensures that predictions for position i rely solely on previously known outputs at positions preceding i. This strategic design enhances the decoder's ability to effectively handle sequential information and produce coherent output sequences.

Encoder

The first step of the encoder is the Input Embedding.

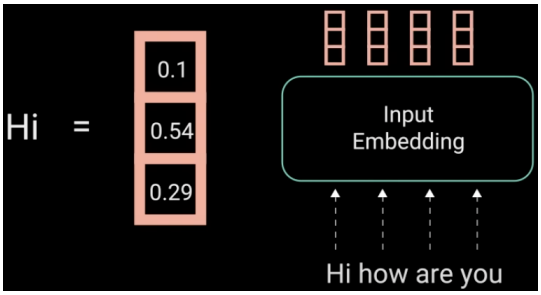

Figure 1.8: Encoder: Input Embedding

The input is passed through a word embedding layer, which acts as a kind of reference table. This layer effectively captures a learned representation factor for each individual word. Consequently, each word corresponds to a vector of continuous values, as shown in Figure 1.8.

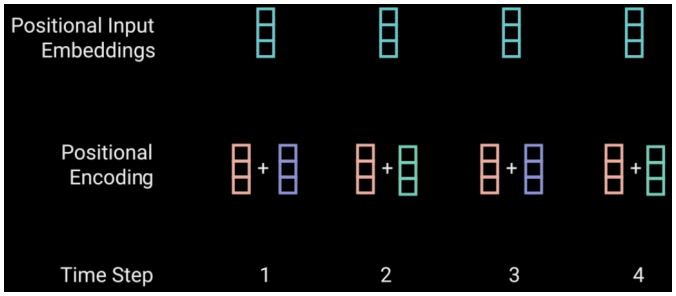

Figure 1.9: Encoder: Positional Embedding

The next stage is to inject positional data into the embeddings, as shown in Figure 1.9. These positional embeddings are combined with the input embeddings to form the positional input embeddings used in the subsequent stages.

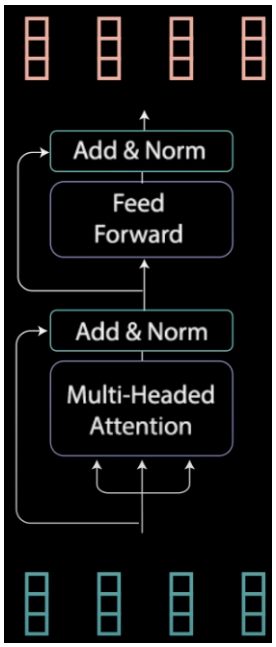

Figure 1.10: Encoder

The primary objective of the encoder is to transform the entire input sequence into a continuous and abstract representation that captures the knowledge acquired throughout the sequence. This component consists of two sub-modules, as shown in Figure 1.10: first, the multi-head attention mechanism is applied, followed by a fully connected network. In addition, residual connections surround each of these two sub-modules, and then a layer normalisation step, referred to as "Add & Norm" in Figure 1.7, is applied.

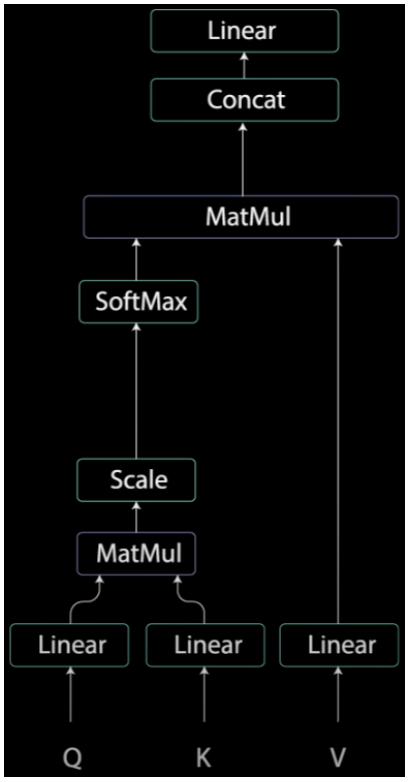

Figure 1.11: Encoder: Multi Head Attention

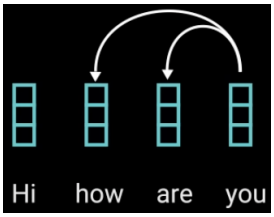

Figure 1.12: Encoder: Example of Multi Head Attention

Multi-head attention uses a specialised attention mechanism known as self-attention, as shown in Figure 1.11. Self-Attention allows the model to make associations between each individual word in the input and other words in the same input sequence. For example, as shown in figure 1.12, the word "you" can be associated with the words "how" and "are". This mechanism makes it easier to capture natural language structures.

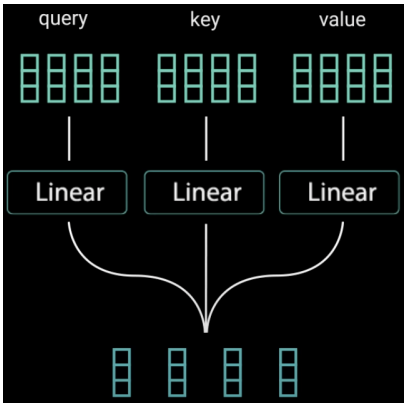

Figure 1.13: Encoder: Query, Key, Value

Self-attention is achieved by processing the input through three separate, fully connected layers, generating the query, key and value vectors. These terms come from the context of retrieval systems.

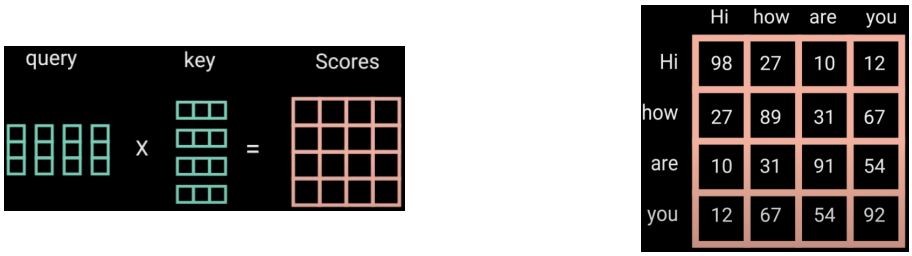

Figure 1.14: Encoder: Query, Key, Value - Score Matrix

Matrix multiplication is performed on the queries and keys to generate a score matrix, as shown in Figure 1.14 within the MatMul operation shown in Figure 1.11. This score matrix plays a key role in determining the amount of attention each word should be given to other words: higher scores correspond to more attention. Subsequently, as shown in figure 1.11 under the label "Scale", these scores are scaled. This scaling step is essential to stabilise the gradients, as direct multiplication of the values can lead to undesirable exploding effects.

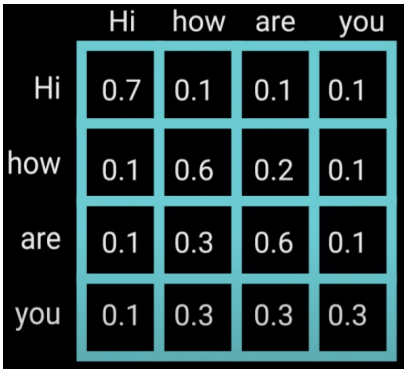

Figure 1.15: Encoder: Multi Head Attention - Softmax

Then, as shown in Figure 1.15 in the context of the Softmax operation, the scaled scores are subjected to a transformation. This transformation is crucial in the context of the multi-head attention mechanism shown in Figure 1.11. It involves the application of the softmax function, which is used to convert the scaled scores into attention weights. These attention weights represent probabilities that range from 0 to 1. The softmax operation plays a critical role in increasing the significance of higher scores while decreasing the influence of lower scores. Consequently, this process helps the model to achieve greater confidence in selecting which words to attend to during its computation.

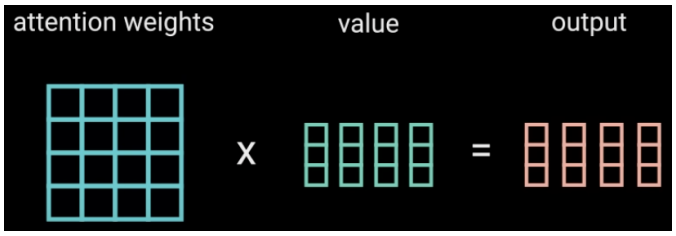

Figure 1.16: Encoder: Multi Head Attention - Matrix Multiplication

In the context of multi-head attention within the encoder architecture, as shown in Figure 1.11, the process involves taking the attention weights, as shown in Figure 1.16, and using them to perform a matrix multiplication operation on the value vector. This operation produces an output vector. Importantly, the softmax scores associated with the attention weights play a crucial role in this operation. In particular, higher softmax scores correspond to words that the model considers more important, effectively preserving their contribution in the output vector. Conversely, lower scores have the effect of

reducing the importance of less relevant words, essentially attenuating their influence in the final output.

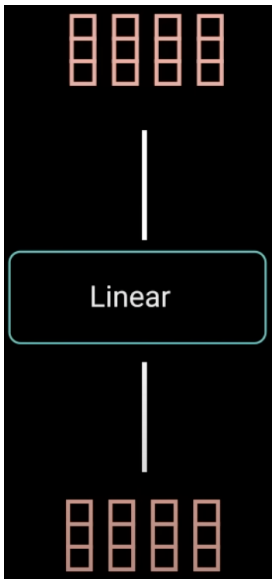

Figure 1.17: Encoder: Multi Head Attention - Linear Layer

Then, as shown in Figure 1.11, the output vector from Figure 1.17 is fed into a linear layer for further processing.

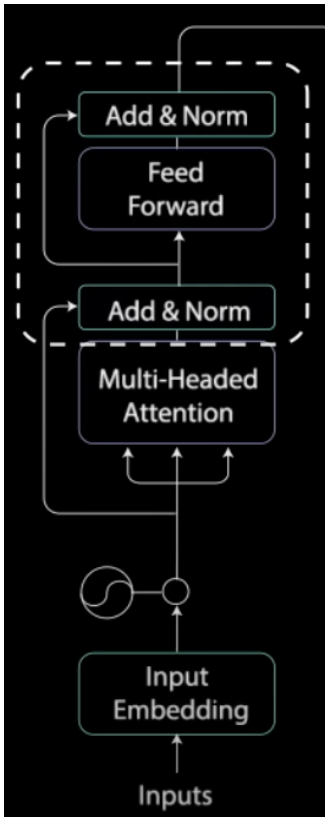

Figure 1.18: Encoder: Feed Forward

Looking again at the encoder architecture, the next step after the multi-head attention phase is a feed-forward operation (as shown in Fig. 1.18). Prior to this feed forward step, the architecture incorporates a crucial element known as a residual connection, whereby the output vector from the multi-head attention mechanism is added to the original vector. This combined output is then subjected to a layer normalisation process. The resulting normalised residual output is then fed into a pointwise feed-forward network for additional computation. The resulting output from this feed-forward network is again combined with the input to the feed-forward network and subjected to another round of normalisation.
The purpose of these residual connections is to facilitate efficient training of the network by allowing gradients to propagate directly through the architecture. Meanwhile, the layer normalisation steps serve to stabilise the network, promoting consistent training performance over time. The pointwise feed-forward layer plays a key role in further enhancing the attentional output, potentially giving it a more expressive and informative representation.

Decoder

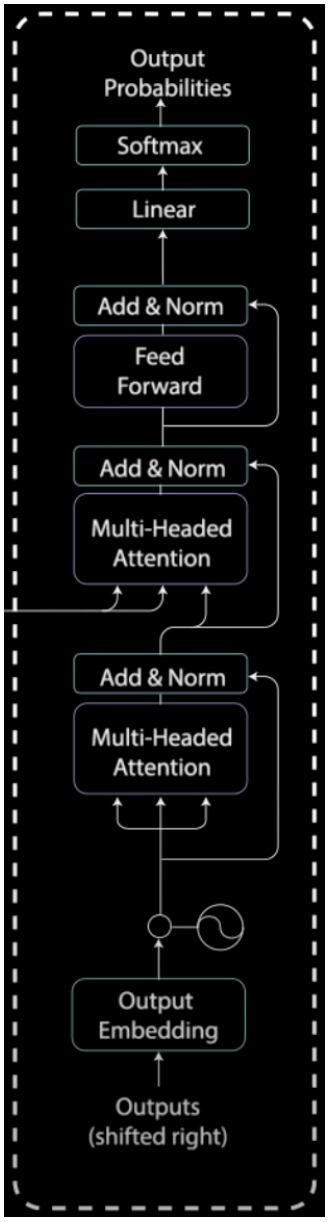

Figure 1.19: Decoder

The encoder's representation helps the decoder to focus its attention on the relevant input words during the decoding process. The primary function of the decoder is to produce text sequences. As shown in Figure 1.19, the decoder consists of sub-layers analogous to those found in the encoder. Specif-

ically, it consists of two multi-headed attention layers and a point-wise feed-forward layer. These sub-layers behave similarly to the layers in the encoder, although each Multi-Headed Attention layer serves a different purpose. The final Softmax layer is introduced to compute word probabilities.

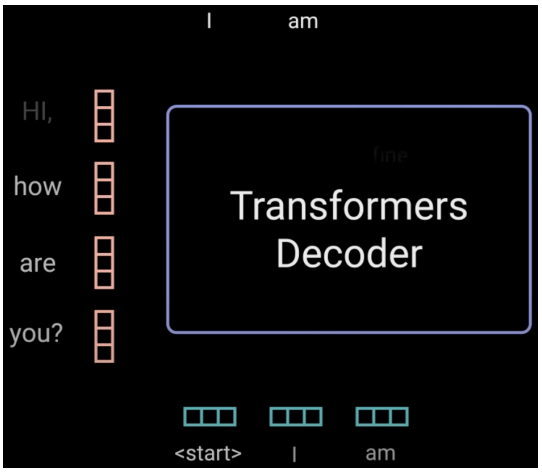

Figure 1.20: Decoder - Auto regressive

In the context of the decoder, a critical aspect to highlight is its autoregressive nature, as illustrated in Figure 1.20. This means that the decoder's operation relies on two key inputs: the sequence of previous outputs and the encoder outputs, which encapsulate important attentional information derived from the input data. It is worth noting that the decoder's decoding process continues until it produces an 'end' token as an output, signalling the completion of its task.

1.4 LLMs - Large Language Models

LLMs [6] are a type of generative AI model that uses deep learning techniques to generate human-like text. There are significant distinctions between LLMs and smaller Pre-trained Language Models (PLMs). Firstly, LLMs demonstrate surprising emergent abilities that were not observable in previous smaller PLMs. These newfound abilities are crucial for enhancing the performance of language models on complex tasks, resulting in unprecedented power and effectiveness for AI algorithms. Secondly, LLMs have the potential to revolutionize the way humans develop and utilize AI algorithms. Unlike small PLMs, LLMs are primarily accessed through a prompting interface, such as the GPT-4 API [7]. Users need to comprehend how LLMs function and structure their tasks in a manner that allows the LLMs to comprehend and respond effectively. This shift in approach introduces new possibilities for human-AI interactions.

Corpora	Size	Source	Latest Update Time
BookCorpus [122]	5GB	Books	Dec-2015
Gutenberg [123]	-	Books	Dec-2021
C4 [73]	800GB	CommonCrawl	Apr-2019
CC-Stories-R [124]	31GB	CommonCrawl	Sep-2019
CC-NEWS [27]	78GB	CommonCrawl	Feb-2019
REALNEWs [125]	120GB	CommonCrawl	Apr-2019
OpenWebText [126]	38GB	Reddit links	Mar-2023
Pushift.io [127]	2TB	Reddit links	Mar-2023
Wikipedia [128]	21GB	Wikipedia	Mar-2023
BigQuery [129]	-	Codes	Mar-2023
the Pile [130]	800GB	Other	Dec-2020
ROOTS [131]	1.6TB	Other	Jun-2022

Figure 1.21: LLMs Corpora

LLMs are pre-trained on massive amounts of text data, such as books, articles, and websites, and use this data to learn the patterns and structures of language. The term "massive amounts of text data" refers to hundreds of billions of words. This is one of the main features that make LLMs different from simple PLMs. To fulfill this requirement, various accessible training datasets have been made available for research purposes. Figure 1.21 from [6] provides a concise overview of several widely used corpora for LLM training.
The most well-known example of LLMs is the GPT (Generative Pre-trained

Transformer) series developed by OpenAI [8]. The GPT models use the Transformer architecture and are trained on a large corpus of text data using unsupervised learning techniques. The GPT-3 model [9], for example, has been trained on a dataset of over 570GB of text data and has approximately 175 billion parameters, making it one of the largest LLMs.
LLMs are capable of a wide range of language tasks, including language translation, text summarization, question-answering, and language generation. They are often used in natural language processing (NLP) applications to assist with tasks such as customer service chatbots, content creation, and language translation.

$$
\begin{aligned}
L(N) &= \left(\frac{N_c}{N}\right)^{\alpha_N}, \quad \alpha_N \sim 0.076, N_c \sim 8.8 \times 10^{13} \\
L(D) &= \left(\frac{D_c}{D}\right)^{\alpha_D}, \quad \alpha_D \sim 0.095, D_c \sim 5.4 \times 10^{13} \\
L(C) &= \left(\frac{C_c}{C}\right)^{\alpha_C}, \quad \alpha_C \sim 0.050, C_c \sim 3.1 \times 10^{8}
\end{aligned}
$$

Figure 1.22: Scaling Law

LLMs achieve significant improvements in model capacity through the scaling of model size, data size, and total compute. Therefore, it becomes essential to establish a quantitative method for assessing the impact of scaling. In this context, the figure 1.22 from [10] represents a scaling law for Transformer-based LLMs, where L(.) stay for cross entropy loss. The term "scaling law" in the context of LLMs, refers to the observed relationship between the performance and the size (scale) of the model. Scaling laws describe how various metrics, such as model capacity, computational requirements, and performance, change as the size of the LM increases. In figure 1.22 N stays for model size, D for dataset size and C for the amount of training compute. The three laws were established through the process of fitting model performance across a range of diverse data sizes (ranging from 22 million to 23 billion tokens), varying model sizes (from 768 million to 1.5 billion non-embedding parameters), and different levels of training compute. These laws were derived based on certain assumptions, such as ensuring that the analysis of one factor is not constrained by the simultaneous influence of the other two factors. The research revealed a substantial interdependency between model performance and these three influencing factors.

A recent problem about the use of LLMs is the alignment. Alignment tuning is a crucial process for LLMs as they are trained on diverse data, which can lead to the generation of toxic, biased, or harmful content. To ensure LLMs adhere to human values, such as being helpful, honest, and benign, an effective tuning approach is necessary.

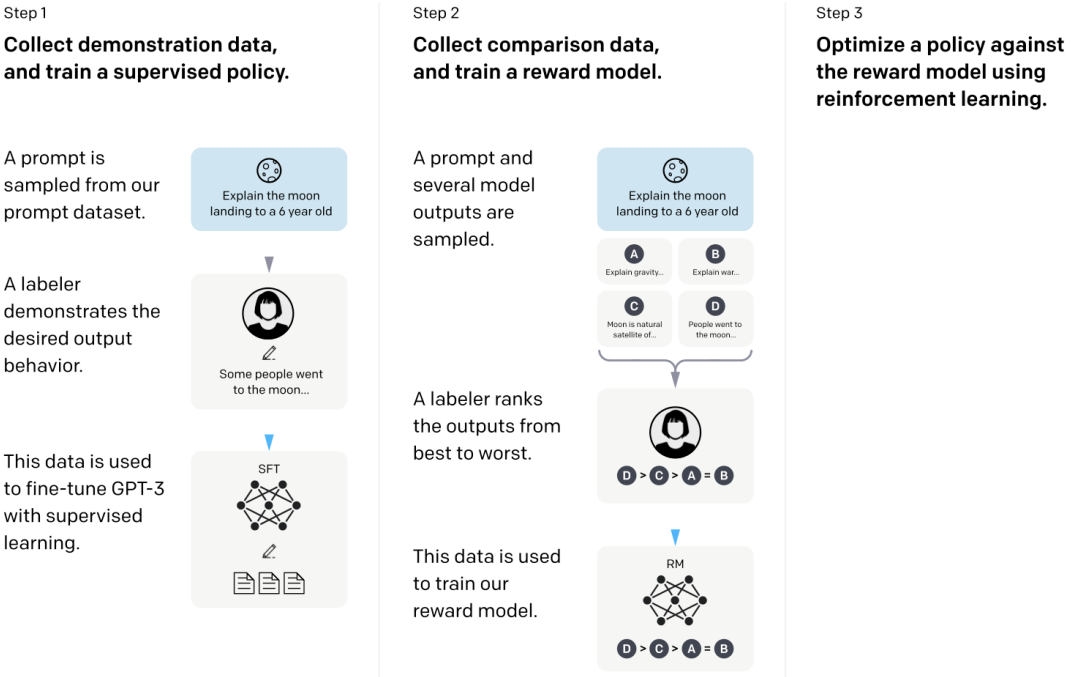

Figure 1.23: InstructGPT

InstructGPT [11] proposes a successful method that uses reinforcement learning with human feedback (illustrated in figure 1.23). This approach involves incorporating human input during training through carefully designed labeling strategies. ChatGPT [4] employs a similar technique to InstructGPT, demonstrating its capability to align with human values by producing high-quality, harmless responses. For instance, it can refuse to answer insulting questions, showcasing its strong alignment capacity.
Other critic aspect is that the computational resources required to train and run these models are significant, making them inaccessible to many researchers and organizations. In this thesis we aim to use pre-trained models to avoid such costs and make inference over Education tasks, as explained in Chapter 3.
Before discussing the use of LLMs in Education (main topic of Chapter 2), it is necessary to understand basic knowledge about the history of LLMs and

about how these models can be pre-trained, fine-tuned, used and evaluated. The following paragraphs explain the above mentioned topics.

1.4.1 A short history of LLMs

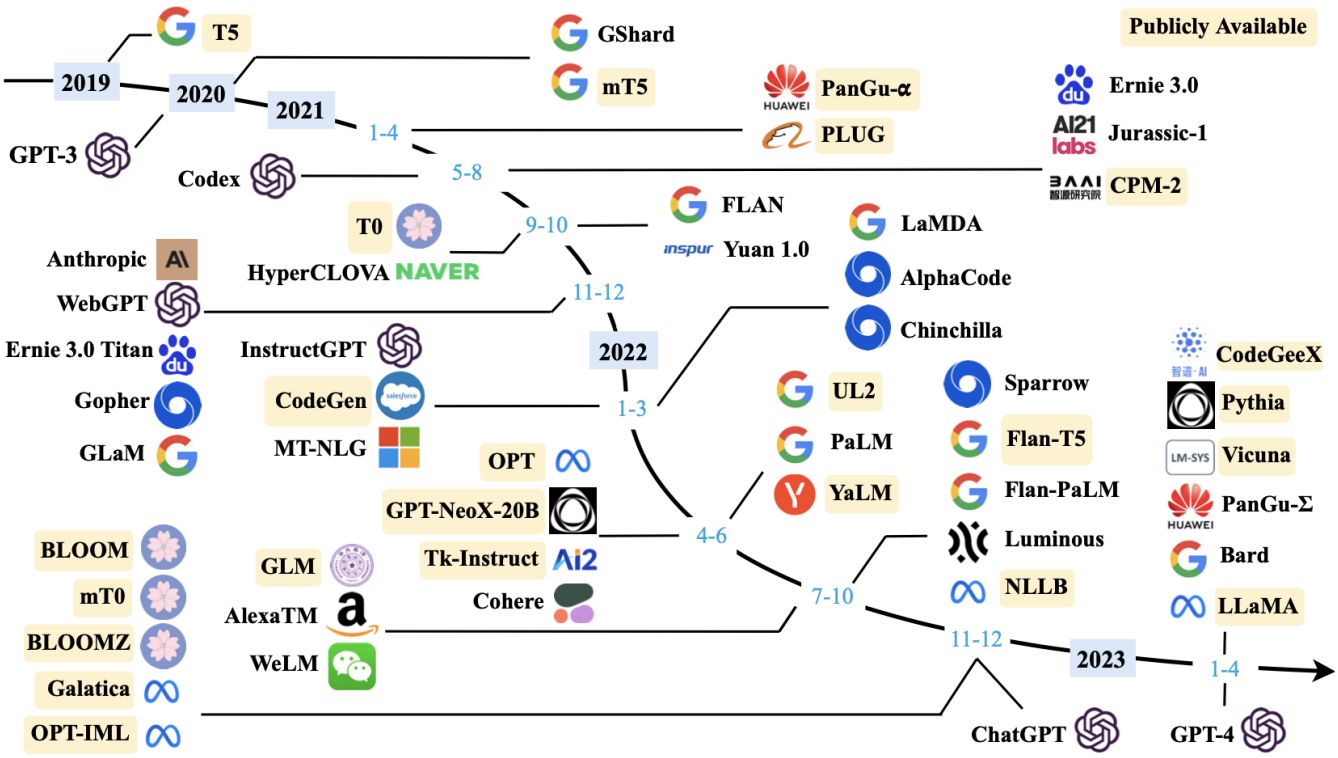

Figure 1.24: LLMs over years

At the end of 2022, OpenAI [8] releases ChatGPT [4], an incredibly powerful conversational LLM. This allowed the world to focus its research on LLMs. In this section, we will briefly discuss the GPT series to illustrate how LLMs take place in years. Figure 1.24 from [6] shows the most important LLMs actually released.
Transformer's advent [5] led to GPT-1 and GPT-2, foundational for subsequent GPT-3 and GPT-4. Figures 1.25 and 1.26 from [12] shows differences between models of GPT series.

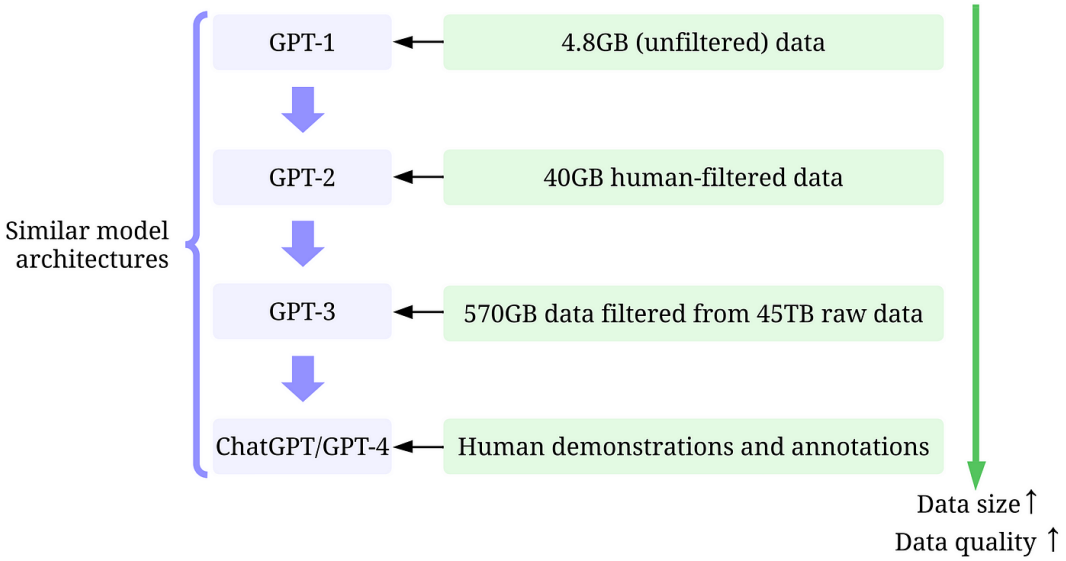

Figure 1.25: GPT series in terms of datasource size

	GPT-1	GPT-2	GPT-3
Parameters	117 million	1.5 billion	175 billion

Figure 1.26: GPT series in terms of number of parameters

- GPT-1, released in 2018, utilized a generative, decoder-only Transformer architecture. It combined unsupervised pre-training and supervised fine-tuning to predict the next word, setting the core for GPT-series models.
- GPT-2, introduced in 2019, scaled to 1.5B parameters and used a large webpage dataset, WebText [13], for unsupervised language modeling. It explored task-solving via probabilistic multi-task approaches, showing the potential of unsupervised models for various tasks.
- GPT-3 (2020) scaled to 175B parameters, introduced in-context learning (ICL) for few-shot or zero-shot tasks. It excelled in NLP tasks and demonstrated exceptional reasoning and domain adaptation capabilities, surpassing basic scaling laws. GPT-3 marks a significant milestone in the evolution from PLMs to LLMs, validating the immense model capacity achieved through neural network scaling.
- OpenAI released ChatGPT [4] in 2022, a conversation model trained similarly to InstructGPT [11], optimized for dialogue with human-

generated conversations. ChatGPT excelled in reasoning, context tracking, and aligning with human values. Later, plugin support extended its capabilities.

- GPT-4, released in 2023, handled multimodal inputs, outperforming GPT-3.5 in complex tasks. It addressed safety concerns through iterative alignment and intervention strategies. Despite progress, limitations persist, including hallucinations and risky responses.

1.4.2 Pre-Training

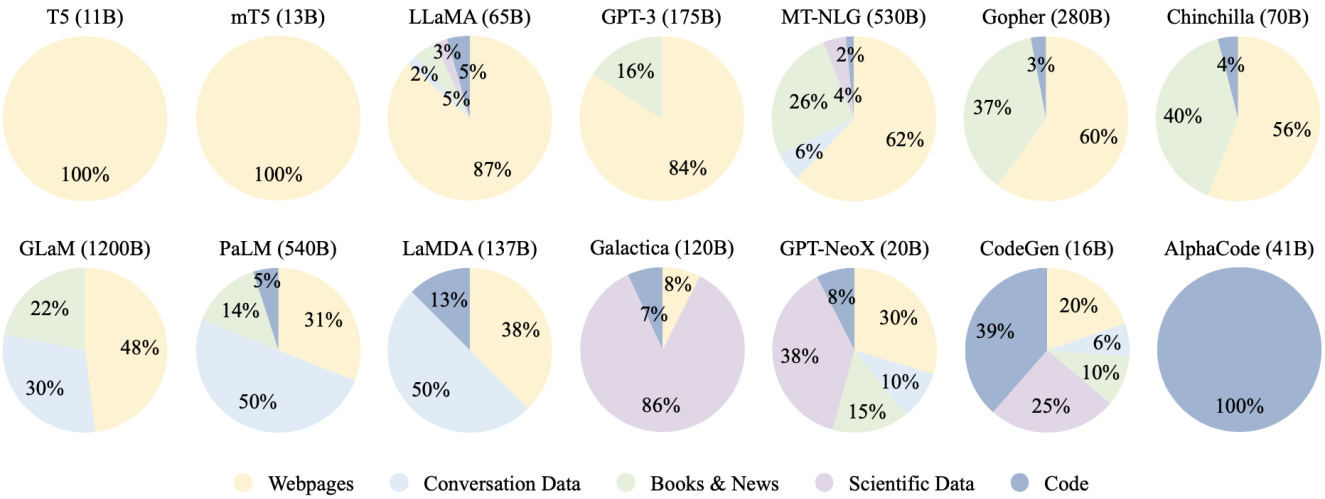

Figure 1.27: LLMs data sources in pre-training

Pre-training is crucial for LLM abilities, involving acquiring language understanding and generation skills through large-scale corpora. The quality and scale of the pre-training data significantly impact LLM capabilities. Effective pre-training requires well-designed model architectures, acceleration methods, and optimization techniques. Data collection and processing play a vital role, and LLMs require high-quality data. Pre-training corpus includes general data like webpages and books, enhancing language modeling and generalization. Specialized data, such as multilingual or scientific data, equips LLMs with task-specific capabilities. Figure 1.27 from [6] illustrates the sources of pre-training data for various LLMs.

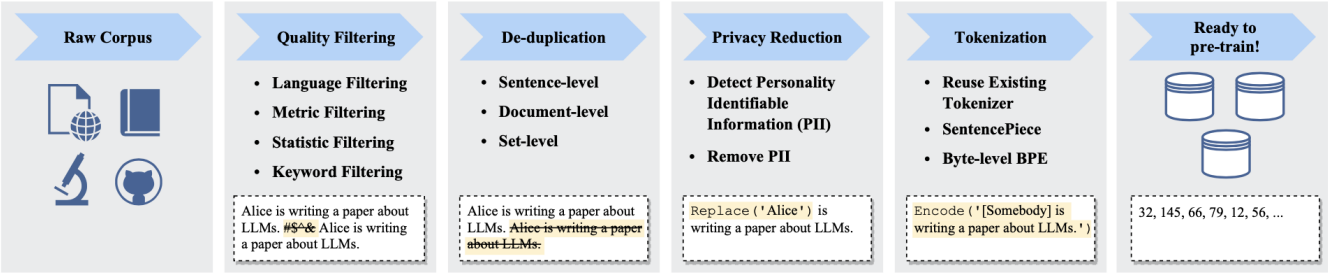

Figure 1.28: LLMs preprocessing

Data preprocessing is a crucial step in constructing the pre-training corpus for LLMs. After gathering a large amount of text data, it is essential to remove noisy, redundant, irrelevant, and potentially toxic data, as these factors can significantly impact the capacity and performance of LLMs. Figure 1.28 from [6] presents a typical preprocessing pipeline for LLMs, comprising several phases:

- Quality Filtering: this phase involves filtering out low-quality or unreliable data to ensure that the pre-training corpus consists of high-quality text. Various criteria, such as language accuracy and relevance, are employed to retain valuable data.
- De-duplication: redundant data can introduce biases and unnecessarily increase the corpus size. De-duplication ensures that duplicate content is removed, resulting in a cleaner and more concise dataset.
- Privacy Redaction: to protect sensitive information, privacy redaction techniques are applied. This ensures that personally identifiable information and confidential data are masked or removed, complying with privacy regulations.
- Tokenization: tokenization is the process of breaking down text into smaller units called tokens. This step is essential for language modeling, as it enables the model to understand and generate text effectively. Tokens can be words, subwords, or characters, depending on the chosen tokenization strategy.

By following this preprocessing pipeline, the resulting pre-training corpus is more refined, representative, and better suited for training powerful and capable LLMs. It ensures that the model focuses on learning meaningful patterns and structures from the data, leading to improved language understanding and generation skills.

Model	Category	Size
GPT3 [55]	Causal decoder	175B
PanGU- α [75]	Causal decoder	207B
OPT [81]	Causal decoder	175B
PaLM [56]	Causal decoder	540B
BLOOM [69]	Causal decoder	176B
MT-NLG [97]	Causal decoder	530B
Gopher [59]	Causal decoder	280B
Chinchilla [34]	Causal decoder	70B
Galactica [35]	Causal decoder	120B
LaMDA [63]	Causal decoder	137B
Jurassic-1 [91]	Causal decoder	178B
LLaMA [57]	Causal decoder	65B
GLM-130B [83]	Prefix decoder	130B
T5 [73]	Encoder-decoder	11B

Figure 1.29: LLMs mainstream architectures

The mainstream architectures of LLMs are represented in figure 1.29 from [6]. The Transformer architecture [5] is the backbone of most LLMs due to its parallelizability and capacity. Existing LLMs can be categorized into three main types: encoder-decoder, causal decoder, and prefix decoder. The encoder-decoder architecture utilizes stacked multi-head self-attention layers for encoding and decoding. Causal decoders use unidirectional attention masks, like the GPT-series models, while prefix decoders modify the masking mechanism to enable bidirectional attention over prefix tokens. Notable LLMs based on these architectures include T5 [14], BART [15], GPT-series [8].

1.4.3 Fine-Tuning

Fine-tuning is a machine learning technique used to improve the performance of a pre-trained model by training it further on a specific task or domain. In fine-tuning, a pre-trained model, which has already learned a set of features from a large dataset, is further trained on a smaller dataset that is specific to the task at hand.
The idea behind fine-tuning is to leverage the pre-trained model's learned features and knowledge to accelerate training on the new dataset and improve its performance on the target task. During fine-tuning, the model's weights are updated through backpropagation to minimize the error between its predictions and the ground truth labels in the target dataset.
Fine-tuning can be done using various techniques such as freezing certain layers of the pre-trained model, adjusting the learning rate, and choosing an appropriate optimizer. The success of fine-tuning depends on factors such as the size and quality of the target dataset, the similarity of the pre-trained model's features to the target task, and the fine-tuning hyperparameters.
Moreover fine-tuning requires training on a moderate number of instances. It can be seen as a supervised training process, and its optimization differs from pre-training in several aspects. One major difference lies in the training objective, where instruction tuning utilizes a sequence-to-sequence loss. The optimization configuration is also adjusted for instruction tuning, including the use of smaller batch sizes and learning rates to ensure stability and efficient convergence. In addition to the optimization configurations, a crucial aspect requires consideration during instruction tuning: balancing the data distribution. Instruction tuning involves fine-tuning on a mixture of different tasks. Therefore, it is essential to balance the proportion of data from each task during the fine-tuning process. This ensures that the model is not biased towards any particular task and can effectively generalize across various tasks.
By addressing this aspect during instruction tuning, the resulting fine-tuned model can effectively leverage the knowledge gained during pre-training and specialize in performing well on the specific tasks at hand. This makes instruction tuning a powerful and efficient approach for tailoring language models to various downstream tasks in natural language processing.
As said before, there still remains the alignment problem. Nowadays, to avoid this problem, there exists a technique called reinforcement learning from human feedback (RLHF) [11]. This approach has been used to fine-tune LLMs such as ChatGPT [4].

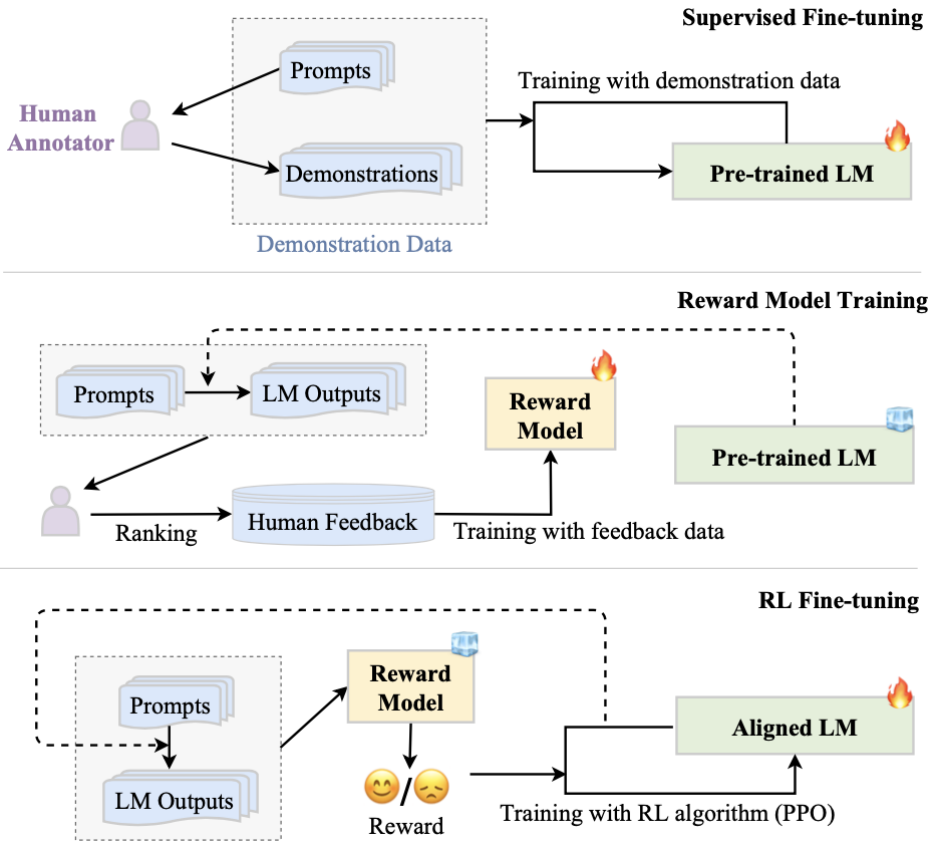

Figure 1.30: RLHF algorithm

The RLHF system comprises three main components: a pre-trained language model (LM) for alignment, a reward model (RM) learning from human feedback, and an reinforcement learning (RL) algorithm for language model training. The pre-trained LM is initialized with existing parameters. The RM provides guidance signals reflecting human preferences, usually as a scalar value. A specific RL algorithm, such as Proximal Policy Optimization (PPO), tunes the pre-trained LM with the reward model signal. Figure 1.30 from [6] represents the RLHF algorithm.

1.4.4 Utilization

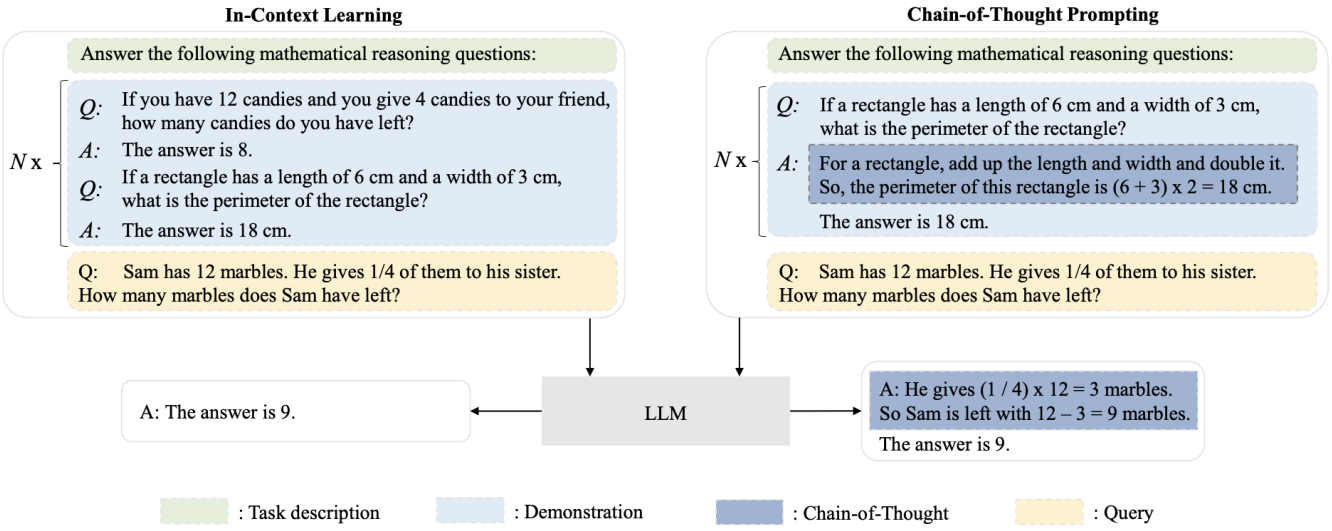

Figure 1.31: Two prompting strategies: In-Context Learning vs Chain-of-Thought prompting

After pre-training or adaptation tuning, one of the major approaches to utilizing Language Models (LLMs) is by designing suitable prompting strategies for solving various tasks. A common method for prompting is in-context learning (ICL). In this approach, the task description and/or task examples are formulated as natural language text prompts. Additionally, chain-of-thought prompting can be employed to enhance in-context learning by involving a series of intermediate reasoning steps into the prompts. The main differences between these two approaches in represented in figure 1.31 from [6].
In in-context learning (ICL), a formatted natural language prompt is used, which typically consists of the task description and/or a few task examples as demonstrations. Firstly, a task description is taken as input, and then a few examples are selected from the task dataset to serve as demonstrations. Finally, the test instance is appended to the demonstrations as input to the LLMs, which then generate the output based on the given prompt. By utilizing task demonstrations, LLMs can recognize and perform a new task without requiring explicit gradient updates. Formally, given the task description I, demonstrations D_k as tuples x_i y_i, and a new input query x_{k+1}, the prediction of the output y_{k+1} generated from LLMs can be formulated as follows:

$$LLM(I, f(x_1, y_1), ..., f(x_k, y_k), f(x_{k+1}, \_\_)) \rightarrow y_{k+1}$$

In this formulation, the actual answer y_{k+1} is left as a blank to be predicted by the LLM.
ICL shares a close connection with instruction tuning, as both approaches utilize natural language to format tasks or instances. However, instruction tuning requires fine-tuning LLMs for adaptation to a specific task, while ICL only prompts LLMs for utilization without requiring explicit fine-tuning.
Overall, the use of in-context learning and chain-of-thought prompting provides a powerful way to leverage the capabilities of LLMs for various tasks, by allowing the model to understand and reason based on natural language instructions and demonstrations. These techniques enable LLMs to perform tasks without the need for explicit updates during inference, making them valuable tools for a wide range of applications. Properly designing the prompts and demonstrations is essential for achieving effective and accurate performance with these approaches.

1.4.5 Evaluation

The effectiveness and superiority of LLMs have been examined through empirical evaluation and analysis using various tasks and benchmarks. Three basic evaluation tasks for language generation and understanding are commonly used:

- Language Modeling: This task involves evaluating the model's ability to predict the next word in a sentence or sequence. Datasets such as PennTreebank [16], WikiText-103 [17], and the Pile [18] are used for evaluation, and the metric of perplexity is commonly employed to assess the model's performance under the zero-shot setting.

- Conditional Text Generation: This task focuses on generating texts based on specific conditions or demands, such as machine translation, text summarization, and question answering. Automatic metrics like Accuracy, BLEU [19], and ROUGE [20] are used to measure the quality of the generated text.

- Code Synthesis: For assessing the code synthesis abilities of LLMs, benchmarks like APPS [21], HumanEval [22], and MBPP [23] are used. These benchmarks include diverse programming problems with text specifications and test cases for correctness checking.

In addition to these basic evaluation tasks, several comprehensive benchmarks have been developed to evaluate LLMs from a more general perspective. Notable examples include:

- MMLU [24]: A versatile benchmark for large-scale evaluation of multi-task knowledge understanding, covering a wide range of knowledge domains from mathematics and computer science to humanities and social sciences.

- BIG-bench [25]: A collaborative benchmark comprising 204 tasks that cover a broad range of topics, intended to probe existing LLMs from various aspects.

- HELM [26]: A comprehensive benchmark that implements a core set of 16 scenarios and 7 categories of metrics.

LLMs have been pre-trained on large-scale corpora, enabling them to capture rich knowledge from the pre-training data. As a result, they are employed as domain experts or specialists for specific areas, and studies have explored their adaptation capacity on domain-specific tasks. In-context learning is used to evaluate the performance of LLMs in domain-specific datasets.
Education is an important application domain where LLMs show promise. Studies have demonstrated that LLMs can achieve student-level performance on standardized tests in subjects like mathematics, physics, and computer science, both in multiple-choice and free-response problems. LLMs have been used as writing or reading assistants for educational purposes, generating logically consistent answers and improving student performance. However, the increasing popularity of LLMs has raised concerns about their rational use, as there are potential risks of cheating on homework and other ethical considerations in education.
In conclusion, the evaluation of LLMs spans various tasks, benchmarks, and application domains, showcasing their versatility and potential impact on different fields. While they exhibit impressive capabilities, it is crucial to carefully assess their usage in educational settings and consider ethical implications for responsible implementation.

Chapter 2

Literature review of LLMs in Education

LLMs can be valuable tools in the field of Education as they provide advanced language processing capabilities and can assist in various educational tasks. Here are some ways LLMs can be used in Education:

- Content Generation: LLMs can generate educational content such as lesson plans, quizzes, worksheets, and study materials [27]. By providing prompts or guidelines, LLMs can produce customized content to meet specific teaching objectives and student needs.

- Automated Feedback and Assessment: LLMs can analyze and evaluate student work, providing automated feedback on assignments, essays, or answers to questions. This can save teachers time and provide students with instant feedback to guide their learning process.

- Natural Language Processing: LLMs can help in developing natural language interfaces for educational software and platforms. This allows students to interact with educational tools using their natural language, making learning more engaging and accessible.

- Adaptive Learning: LLMs can analyze student data and adapt instructional materials to meet individual learning needs. By tracking student progress and understanding their strengths and weaknesses, LLMs can provide personalized learning experiences [28].

- Language Learning and Translation: LLMs can assist in language learning by providing language practice exercises, correcting grammar and

vocabulary errors, and simulating conversations. They can also facilitate translation tasks, allowing students to understand educational materials in their native language.

- Question Answering and Research: LLMs can help teachers and students find answers to questions, conduct research, and explore educational resources. By understanding natural language queries, LLMs can retrieve relevant information from various sources and provide accurate and timely responses.

It's important to note that while LLMs can offer significant support in the field of Education, they should not replace the role of teachers. Educators play a vital role in guiding students, fostering critical thinking, and creating meaningful learning experiences. LLMs can serve as powerful tools to enhance teaching and learning, but human expertise and guidance are crucial for effective education.
In this chapter, we embark on a comprehensive literature review that delves into the integration of LLMs within the realm of education. The review is structured to provide a thorough understanding of LLMs' significance in educational contexts, tracing their evolution from historical milestones to the present day (Section 2.2). We explore the role of LLMs in education, spanning the trajectory from their inception in 1966 to the present, with a specific focus on their application in two critical areas: Intelligent Tutoring Systems (ITS) (Section 2.2.1) and the remarkable impact of ChatGPT (Section 2.2.2). Furthermore, we delve into the methodologies employed to fine-tune LLMs for educational purposes (Section 2.3), offering insights into the strategies that enable these models to cater effectively to educational needs.

2.1 Education

Education is the science or study of teaching and instructional design. It encompasses the principles, methods, and techniques used in education to facilitate effective learning. The field of Education focuses on how to best convey knowledge, skills, and attitudes to learners in various educational settings.

Education involves the development of instructional strategies, curriculum planning, and the design of learning materials and activities. It explores different teaching methods, approaches, and tools that can be employed to optimize the learning process and promote student engagement and achievement.

The goals of Education are to understand how people learn, identify effective teaching methods, and create supportive learning environments.

Education is not limited to a specific educational level or subject area. It encompasses both formal education (schools, universities) and informal learning contexts. It can apply to various subjects, such as mathematics, science, languages, social sciences, and more.

Overall, Education is concerned with understanding and improving the process of teaching and learning, with the aim of enabling learners to acquire knowledge and skills effectively and meaningfully.

2.2 Education: From 1966 to Recent Years

The evolution of chatbots as Education tools began in 1966 with the creation of ELIZA [29], marking the inception of a journey that led to significant technological advancements. These advancements have notably expanded the potential of chatbots as pedagogical tools to support language learning among students.
Over the years, as chatbots have evolved since the days of ELIZA, Intelligent Tutoring Systems (ITS) have emerged as pivotal components, influencing the progress of language education and catalyzing the development of sophisticated language model-powered chatbots like ChatGPT.
The rise of Intelligent Tutoring Systems (ITS) has played a pivotal role in shaping the landscape of language education. These systems aimed to provide personalized and adaptive instruction, mimicking the role of human tutors. Early instances like the PLATO system in the 1970s [30] laid the groundwork for leveraging technology to enhance educational experiences.
The integration of Large Language Model (LLM) technology, exemplified by ChatGPT, represents a natural evolution of Intelligent Tutoring Systems. ChatGPT's capabilities, including contextual memory, understanding user corrections, and filtering inappropriate content, align seamlessly with the objectives of intelligent tutoring. These features empower ChatGPT to engage in contextually relevant interactions with students, akin to personalized human tutoring.
Moreover, ChatGPT's proficiency in sustaining extended goal-oriented conversations positions it as a valuable asset in education. In the domain of language learning, ChatGPT can serve as a versatile partner, enabling students to practice conversational skills, access instructional materials, and aiding teachers in lesson planning. Its adaptability caters to diverse learning styles, promising a personalized educational experience.
The trajectory from early ITS implementations to the advanced capabilities of LLM-powered chatbots like ChatGPT underscores the continual effort to leverage technology in education. By incorporating advanced natural language understanding and generation, these chatbots bridge the gap between traditional teaching methods and modern digital tools. As education evolves, integrating LLM-powered chatbots into the learning process holds the potential for more effective and engaging educational experiences worldwide.
Jeon [31] categorized three distinct types of chatbots beneficial for language learning:

- General-purpose chatbots: Engage in simple daily conversations using Q&A dialogue.

- Specific-purpose chatbots: Designed for language learning, often developed by commercial entities.
- Customized chatbots: Tailored by researchers or educators for specific contexts, built using visual chatbot development platforms.

Recent research [32] consistently validates these chatbot types' effectiveness in language learning. However, limitations exist. Chatbots can play roles beyond mere conversation practice, such as aiding thought organization and furnishing instructional materials, yet these possibilities remain underexplored. Previous chatbots struggled with multi-turn open-ended conversations due to rule-based designs and limited data sources, resulting in contextually inappropriate responses.
The introduction of ChatGPT addresses these limitations. Powered by LLM technology, ChatGPT can generate more human-like responses. LLM integration equips ChatGPT with three vital capabilities [33]:

- Recall of previous user statements.
- Adaptation to user corrections.
- Rejection of inappropriate requests.

These advancements enable ChatGPT to simulate authentic human conversations and respond effectively to diverse user inputs. It excels in sustaining extensive conversational exchanges on specific subjects, suggesting LLM-powered chatbots' potential to serve varied educational functions [34].
In conclusion, integrating LLM-powered chatbots into education, especially language learning, has gained traction. The journey from ELIZA to advanced ChatGPT showcases technology's role in enriching language education through versatile and improved chatbot interactions. This study's primary focus is on ChatGPT due to its prominence in educational literature [4] and its potential to redefine interactions between humans (teachers/professors) and tools.
The subsequent paragraphs will delve into ITS and ChatGPT.

2.2.1 ITS

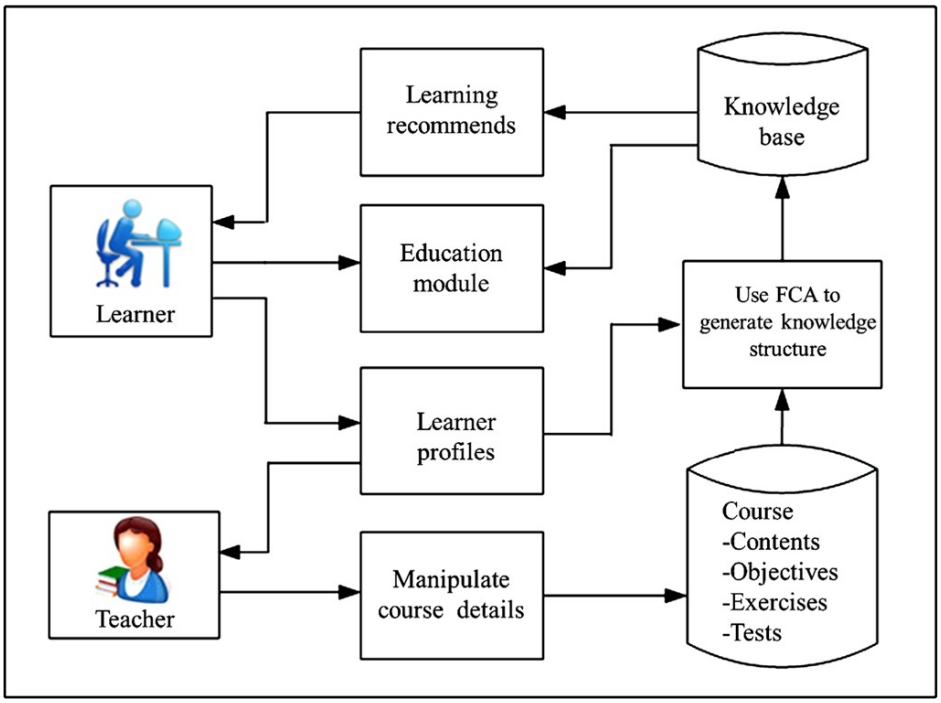

Figure 2.1: ITS

An intelligent tutoring system (ITS) is a computerized platform designed to offer immediate and tailored instruction or feedback to learners, often without requiring human intervention. In figure 2.1 from [35] there is a small representation of ITS. ITSs utilize various computing technologies to facilitate meaningful and efficient learning experiences. These systems find application in formal education as well as professional environments, showcasing their capabilities and limitations. Intelligent tutoring is closely intertwined with cognitive learning theories and design principles, and ongoing research aims to enhance the efficacy of ITS. By aiming to replicate the advantages of personalized, one-on-one tutoring, ITSs address scenarios where students might otherwise receive one-to-many instruction from a single teacher (e.g., classroom lectures) or lack access to a teacher altogether (e.g., online homework). A primary objective of ITS development is to provide all students with access to high-quality education.

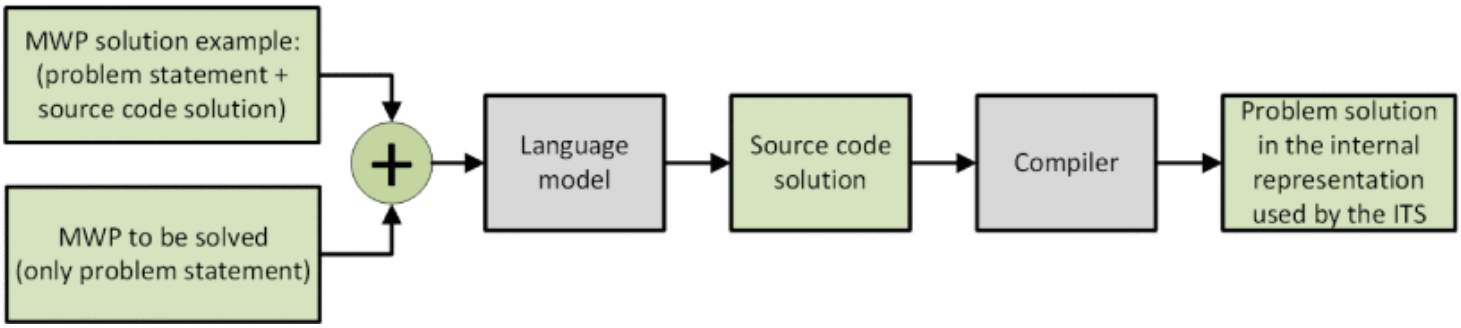

Figure 2.2: Hybrid system: ITS combined with LLM - Architecture

```
""" A book has 3 chapters. The first chapter is 91 pages long
the second chapter is 23 pages long and the third chapter is
25 pages long. How many more pages does the first chapter have
than the second chapter? """
def sol():
    context = dict()
    context['number of chapters'] = 3
    context['number of pages first chapter'] = 91
    context['number of pages sencond chapter'] = 23
    context['number of pages third chapter'] = 25
    context['pages more first chapter'] = (
        context['number of pages first chapter']
        - context['number of pages second chapter']
    )
    return context['pages more first chapter']


""" An industrial machine worked for 5 minutes. It can make 4 shirts a minute.
How many shirts did the machine make? """
def sol():
```

(a) Prompt example and problem to be solved

```
    context = dict()
    context['number of minutes'] = 5
    context['shirts per minute'] = 4
    context['number of shirts'] = (
        context['number of minutes']
        * context['shirts per minute']
    )
    return context['number of shirts']
```

(b) Generated code

Figure 2.3: Hybrid system: ITS combined with LLM - Example

An interesting study [36] for this work proposed an interaction between ITS and LLMs. In figures 2.2 and 2.3 a representation of this study. The researchers introduce an innovative approach to solve mathematical word problems (MWPs) and convert them into the internal representation of Intelligent Tutoring Systems (ITS). The method leverages LLM to generate Python source code capable of solving the problems. A notable advantage of this method is its ability to automatically assign meaningful names to the quantities within the MWP, facilitating conversational interactions between students and the system.
Beyond automated problem-solving and quantity naming, this approach holds potential for automatically translating problem statements into an ITS's internal knowledge representation schema. Such functionality empowers learners to contribute new MWPs to the ITS, enabling practice with a wide range of mathematical problems. Similarly, tutors can efficiently introduce new MWPs at scale. Experimental results reveal the approach's strong performance, particularly with Salesforce's CodeGen model [37], achieving 39.1% accuracy for unknown solutions and 69.1% for known solutions.
This work presents a promising avenue for improving math education within online learning environments. By automating problem-solving, providing automatic quantity naming, and enabling translation into ITS schemas, the approach enhances math skill development and facilitates personalized and scalable learning. The effectiveness of the method hinges on code generation model performance, suggesting potential improvements with more advanced models in the future.
A trade-off exists between model size, generated samples, and performance, with larger models generally performing better. Integrating this system into an ITS remains experimental, where the achieved 69% accuracy might not be sufficient for a final product, but can still simplify MWP encoding and debugging while under human supervision.
However, this approach has limitations. The generation of plain Python code restricts solutions to arithmetic problems, excluding those requiring algebraic equation systems. Additionally, a single solution graph per problem may not cover all possible resolutions. Future work aims to address these issues, explore diverse source code snippets for multiple resolution paths, and integrate the method into an ITS for seamless inclusion of new problems. Evaluation will focus on students' Quality of Experience and the acceptance of the tutoring system.

2.2.2 ChatGPT

ChatGPT [4] is an AI language model developed by OpenAI [8]. It is based on the GPT (Generative Pre-trained Transformer) architecture, specifically GPT-3.5. It is designed to understand and generate human-like text responses based on the given input or prompts.

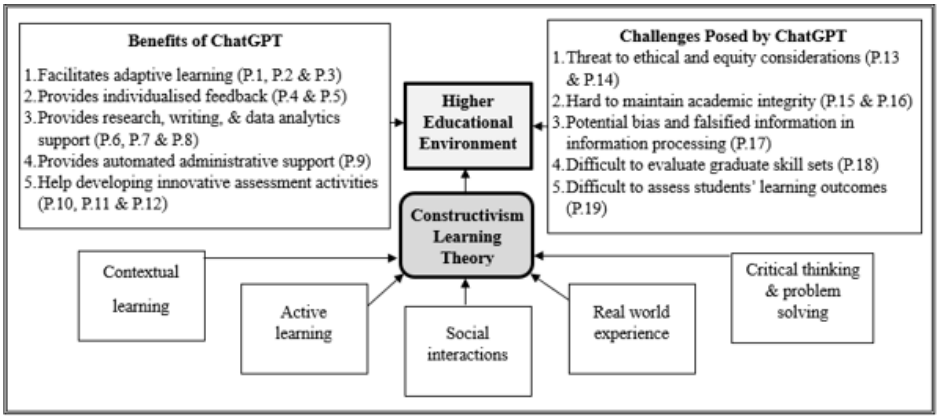

Figure 2.4: Benefits and challenges of ChatGPT an integrated framework

ChatGPT presents both opportunities and challenges in education [38]. In figure 2.4 from [28] there is a small representation of both opportunities and challenges. The "educator's dilemma" arises from the debate between banning or promoting the use of such technologies. However, with proper prompt engineering expertise, ChatGPT can generate high-quality output tailored to specific tasks or goals.
In higher education, ChatGPT can serve as a valuable tool for personalized learning [1]. It can support lecturers in various classroom tasks, create custom exercises and quizzes, offer feedback, and generate educational materials that align with a student's learning style and progress. Additionally, it can assist in developing lecture ideas, drafting seminar plans and module descriptions, and crafting announcement texts. Moreover, ChatGPT can be utilized to assess students' prior knowledge using AI.
A research study [33] examines ChatGPT's educational potential and teachers' evolving roles in response to its implementation. Through chatbot logs and interviews, a symbiotic relationship between teachers and AI emerges. The study reveals that as technology becomes versatile, educators may assume more critical roles to effectively integrate AI, fostering student benefits. The study addresses limitations of prior chatbots by introducing LLM-powered ChatGPT, capable of human-like conversations and diverse pedagogical roles. Teacher-AI collaboration is supported, debunking concerns of

dependency. ChatGPT's adaptability demands teachers' expanded pedagogical decisions, evident in identified roles: resource orchestration, fostering student investigation, and promoting ethical AI use. This study enriches our understanding of AI's integration in education and underscores the importance of teacher-student interactions.
Other works in this direction [39] see ChatGPT as a tool that provide "Personalized tutoring", leveraging this framework to offer tailored instruction to students [40]. This innovative technology enables students to receive timely responses to their queries without waiting for a tutor or teacher, saving them significant time. Accessible around the clock, ChatGPT accommodates students' schedules, simplifying study time integration. Evidence shows personalized tutoring's academic benefits. Bloom's 1984 study revealed superior performance for students with individualized instruction, highlighting AI-powered chatbots' role, offering instant feedback and guidance, central to personalized learning [41] [40].
While the application of ChatGPT in education holds promise, it is important to consider ethical implications, potential biases, and the need for human oversight. Striking a balance between utilizing the capabilities of ChatGPT and maintaining the role of human educators is essential to ensure effective and responsible integration of AI technologies in education.
ChatGPT faces the hurdle of AI feedback acceptance over human instructors, conflicting with constructivist learning's interactive and collaborative nature. To instill trust, universities should combine ChatGPT and human instructors for precise, credible feedback, curbing misinformation [28].
Having said that, the question is whether tools like ChatGPT can be used as they are to create teaching materials, or whether they need to be fine-tuned using some specific data from a particular argument. This is the aim of this study research: try to fine-tune LLMs to obtain better performance helping professors in their work and students during learning phases.

2.3 How to fine-tune LLMs for Educational purposes

In the dynamic realm of natural language processing and artificial intelligence, language models have emerged as powerful tools for understanding and generating human-like text. Among these models, GPT-3 (Generative Pre-trained Transformer 3) stands out as one of the largest and most sophisticated language models, boasting billions of parameters and the ability to generate coherent and contextually relevant text.
The challenge of aligning language models with human intent forms the focal point of this study. Using a fine-tuning approach can refine and enhance LLM's performance [11]. Integrating human-guided refinements into the model's training process aims to bridge the gap between raw generative capacity and nuanced user intent, elevating the practicality and reliability of LLM across diverse tasks. Through careful analysis of labeler-written prompts, user submissions, and human feedback, insights are gained into how the process of fine-tuning contributes to aligning language models with user expectations. Additionally, light is shed on the potential benefits of this approach, including enhanced truthfulness, reduced toxicity, and improved performance on relevant benchmarks. Ultimately, this investigation serves as a stepping stone towards a more user-centric and intention-aligned approach to language model development. By harnessing the power of human feedback, the study strives to unlock the full potential of LLMs, enabling them to not only understand and generate text proficiently but also to serve as effective tools for communication, learning, and problem-solving in a manner that resonates with the diverse needs of human users. The method of [11], called *InstructGPT*, includes some steps that follow the footsteps of [42] [43] in stylistic continuation and summarization domains:

1. Collect Demonstration Data and Train Supervised Policy: Trained human labelers demonstrate the desired behavior on the input prompt distribution. A pretrained model is fine-tuned using supervised learning based on this demonstration data.

2. Collect Comparison Data and Train Reward Model: A dataset of output comparisons is assembled, where labelers express their preferences for given inputs. A reward model is then trained to predict human-preferred outputs.

3. Optimize Policy with PPO using Reward Model: The reward model's output serves as a scalar reward. The supervised policy is fine-tuned

using the Proximal Policy Optimization (PPO) algorithm to optimize this reward.

Steps 2 and 3 can be repeated iteratively. This cyclic process enhances the alignment of the language model with human intent.

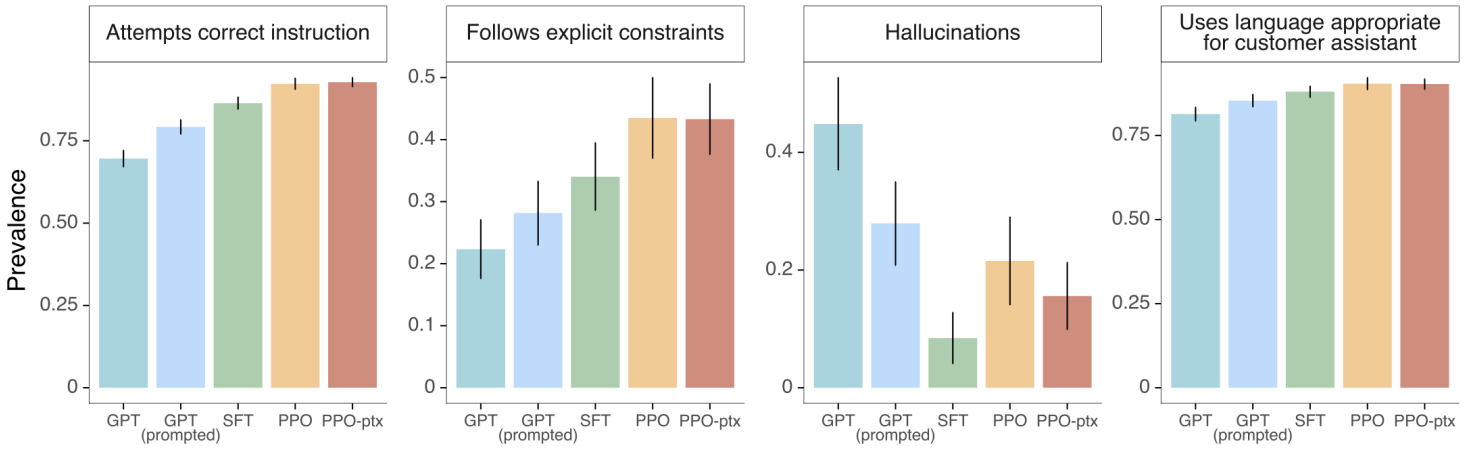

Figure 2.5: InstructGPT (PPO-ptx) vs other models

As results of this technique, in figure 2.5 is illustrated the outcome of human evaluations conducted on diverse models, utilizing the API prompt distribution. Evaluation criteria were based on the frequency of model outputs being preferred over the 175B supervised fine-tuning (SFT) model's outputs. Notably, InstructGPT models (PPO-ptx), along with a pretraining mix-excluded variant (PPO), exhibit significant performance enhancements in contrast to GPT-3 baselines (GPT, GPT prompted). Specifically, the outputs of the 1.3B PPO-ptx model are consistently favored over the 175B GPT-3 model's outputs.
This paper provides a robust depiction of result variability. The insights conveyed by figure 2.5 underscore the effectiveness of InstructGPT models in terms of user preference and alignment with human intent, signifying a notable stride in the field.

Chapter 3

Fine-Tuning LLaMA 7B for precise quiz-question answers

LLMs like GPT-3 have demonstrated remarkable abilities in understanding and generating human-like text. When faced with single-choice questions regarding university courses, LLMs utilize their extensive training on diverse text sources to generate responses. In this context, this work shows that LLMs achieve an accuracy of 68%, which means that out of a given set of single-choice questions, created by university professors, the LLM answers approximately 68% of them correctly. This level of accuracy suggests a notable capability to comprehend and reason through the question, as well as the ability to generate a relevant answer that aligns with the provided options. It is important to note that while a 68% accuracy indicates a significant level of accuracy, there is still room for improvement. As LLMs continue to evolve and be fine-tuned, their accuracy rates are likely to increase, enhancing their performance in single-choice question scenarios. Nonetheless, achieving a 68% accuracy already underscores the impressive potential of LLMs to process and respond to such questions, showcasing their contribution to natural language understanding and communication.
The main purpose of this chapter is to focus on experiments of how LLMs can generate answers to single-choice questions, and to illustrate how these models can be prompted and fine-tuned, showing how results can change. With this in mind, this chapter has been divided into 3 sections: problem statement, inference and fine-tuning.
Before going through results of inference this chapter introduce a better overview of the problem statement.

3.1 Problem statement

LLMs have revolutionized the field of natural language processing by demonstrating remarkable proficiency in generating text sequences. However, their performance varies significantly depending on the nature of the questions they are posed.

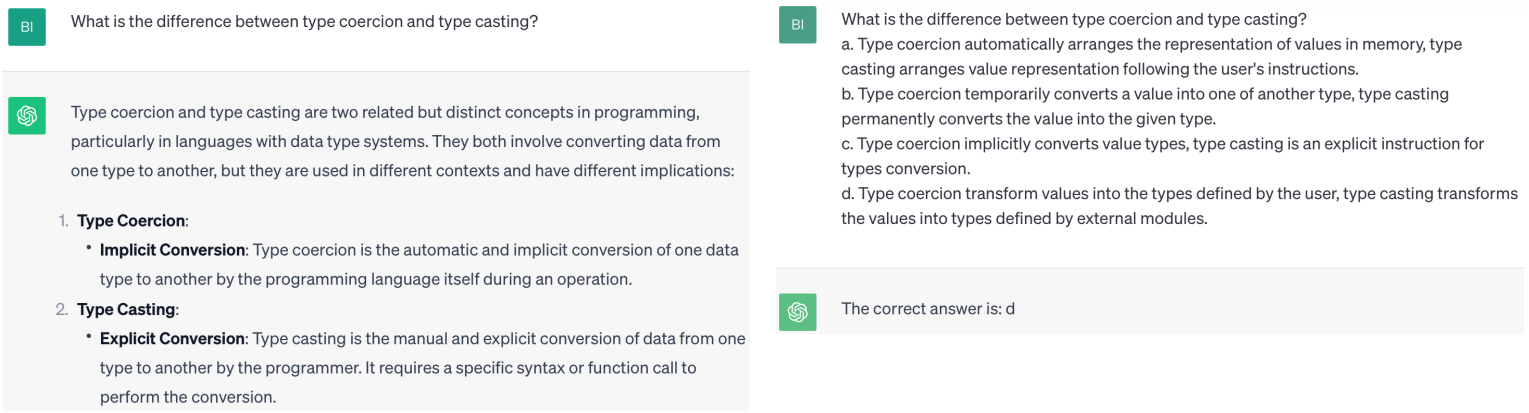

Figure 3.1: RQ1: open-ended vs. single-choice question

In particular, LLMs excel at answering open-ended questions, often producing responses that are coherent and informative. Yet, when confronted with the rigidity of single or multiple-choice questions commonly found in university courses and quizzes, their efficiency diminishes (see Fig. 3.1). This discrepancy raises a pressing problem: how can we harness the full potential of LLMs, especially in the context of quiz questions? This research delves into this challenge by examining various versions of LLMs, ranging from 7 billion to 70 billion parameters, under the framework of LLaMA2. Moreover, it endeavors to fine-tune the smaller model to attain the same level of accuracy as their larger counterparts, shedding light on the crucial aspect of cost efficiency in terms of resource consumption in the realm of AI-driven question answering.
In this work the attention is posed on three different research questions:

- **RQ1**: how to harness the full potential of LLMs, not only in the context of open-ended questions but also in the context of quiz questions?
- **RQ2**: Do smaller fine-tuned LLMs provide better results with regards to bigger only pre-trained LLMs?
- **RQ3**: If smaller LLMs produce better results, are they less resource consuming?

3.2 Inference

LLMs, including ChatGPT [4] and LlaMA2-chatbot [44], have been harnessed to perform inference on a diverse range of topics, such as Machine Learning, Programming Languages and Security. In a comprehensive evaluation involving 60 single-choice questions from each domain, these advanced language models have demonstrated an impressive level of accuracy by achieving a remarkable 68% correctness in their answers. This achievement underscores the models' remarkable capacity to comprehend and reason through a wide array of complex questions, accurately identifying the correct choices among a set of options.

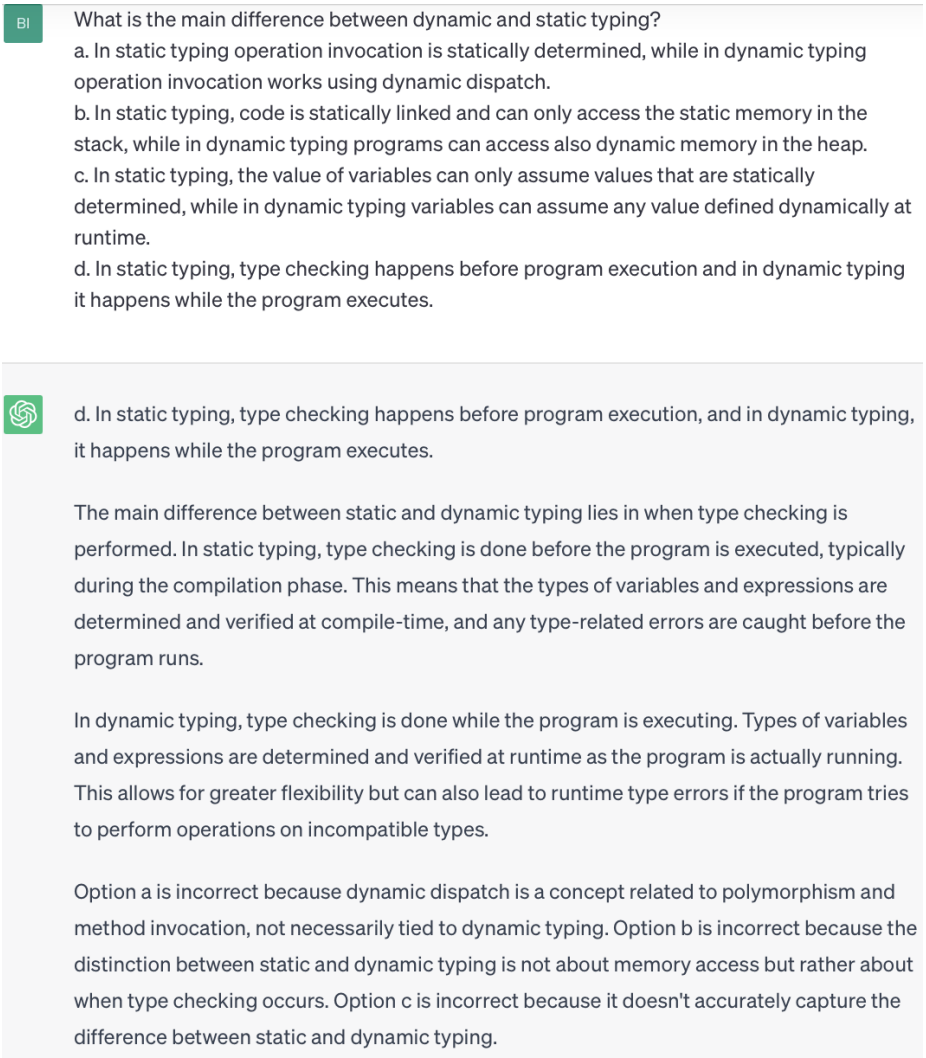

Figure 3.2: Example of single-choice question proposed to ChatGPT

In figure 3.2 a representation of a single-choice question proposed to ChatGPT.
By effectively leveraging their extensive training data and understanding of context, these LLMs have proven their ability to provide informed responses across multiple domains. This outcome reflects the potential of LLMs to serve as versatile tools for knowledge inference, offering valuable insights and information to users across a spectrum of subjects, including machine learning, programming languages, and security. As ongoing advancements continue to refine these models, it is foreseeable that their accuracy rates will further improve, reinforcing their utility as reliable sources of information and expertise.

	Domains			
	ML	**S**	**PL**	**Mean**
ChatGPT	**17/25 = 68%**	**8/10 = 80%**	**16/25 = 64%**	**41/60 = 68%**
LLaMA2 70B	14/25 = 56%	**8/10 = 80%**	15/25 = 60%	37/60 = 62%
LLaMA2 13B	14/25 = 56%	7/10 = 70%	13/25 = 52%	34/60 = 56%
LLaMA2 7B	10/25 = 40%	5/10 = 50%	13/25 = 52%	28/60 = 47%

Table 3.1: Percentage of correct answers - ML: Machine Learning, S: Security, PL: Programming Languages

In table 3.1 there are the results of the inference of these two different LLMs (ChatGPT and LLaMA2 chatbot) used to answer to 60 questions for 3 domains: Machine Learning (25 questions), Security (10 questions) and Programming Languages (25 questions).
Focusing on three different domains provides a well-rounded and comprehensive assessment of the capabilities and performance of LLMs like ChatGPT and llama2-chatbot. The choice to span these diverse areas reflects a strategic approach to evaluating the models' versatility, adaptability, and knowledge representation across different subject matters. The principal motivations are resumed into the following listing.

- Representation of Expertise: These domains encompass a wide spectrum of topics that are highly relevant in today's technology-driven world. By assessing the models' accuracy in each domain, we gain insights into how well they can capture and reproduce expert-level knowledge across various disciplines.

- Real-world Applicability: Machine learning, Programming Languages and Security are crucial fields with practical applications across industries. Analyzing the models' performance in these domains helps us

understand their potential utility in addressing real-world challenges and assisting users with complex problem-solving.

- Cognitive Flexibility: Testing LLMs across diverse domains evaluates their cognitive flexibility and adaptability. It showcases their ability to navigate and comprehend different terminologies, concepts, and contexts, making them more versatile tools for a range of users with distinct needs.

- Knowledge Generalization: Focusing on three distinct domains enables us to assess how well the models can generalize knowledge from their training data to new and unseen questions. This provides insights into their capacity to infer information and make educated guesses even when confronted with unfamiliar scenarios.

- Limitations and Insights: Evaluating LLMs in multiple domains helps us identify potential limitations and areas for improvement. It sheds light on the types of questions and contexts where the models may struggle, guiding further research and development efforts.

In essence, the decision to focus on these three different domains allows for a more comprehensive evaluation of LLMs' capabilities, showcasing their potential as versatile and knowledgeable AI assistants across a wide array of subjects.

3.2.1 Inference resource consumption

Testing the inference capabilities of various Large Language Models (LLMs) provides valuable insights into their performance and suitability for different applications. Due to the size of LLMs used, tests were made using different interfaces and frameworks:

- ChatGPT was evaluated through the official web interface [4].

- LLaMA 7B was tested using the lit-gpt framework [45].

- LLaMA 13B was assessed through the llama2.ai web interface [44].

- LLaMA 70B was also evaluated via the llama2.ai web interface.

Each of these tests aimed to gauge the LLMs' ability to comprehend and generate human-like text. The choice of interface and framework depended on the specific model's availability and hardware constraints. Overall, these assessments help users determine the most suitable LLM and interface for

their specific needs, whether it's for conversational AI, content generation, or advanced natural language understanding tasks.
Therefore, a particular interest was posed over LLaMA 7B because it is the only LLM (of those chosen) that allows to be used by a relatively small size GPU.

Settings	Inference Memory
Default (bfloat16-mixed)	N/A
--precision "bf16-true"	13.82 GB
--quantize "bnb.nf4"	4.66 GB
--quantize "bnb.nf4-dq"	4.34 GB
--precision "bf16-true" --quantize "bnb.nf4"	4.66 GB
--precision "bf16-true" --quantize "bnb.nf4-dq"	4.34 GB

Figure 3.3: lit-gpt inference resource consumption

In terms of resource consumption, lit-gpt provides a table (Fig. 3.3) showing the hardware requirements for LLaMA 7B inference. The values refer to LLaMA 7B model inference using a GPU with ∼24GB of memory (NVIDIA RTX 3090), the same used for the experiments in this work. The first column represents the settings to be used when executing the inference command (or the arguments passed to the inference Python program). In this particular case, it is possible to pass two arguments:

- *precision*: an argument that allows the model weights to be automatically converted to a lower precision (e.g. 16 instead of 32) to use less memory.

- *quantize*: an argument that allows the weights to be represented as fixed point numbers with a limited number of decimals (e.g. 4-bit or 8-bit).

First row represents the inference over the LLaMA 7B model as it is, without setting precision or quantization arguments. In this work the value was tested and is similar to the one of the second row: ∼14GB.

3.3 Fine-tuning

This section deals with the fine-tuning results and into the three key research questions aimed at optimizing the utilization of LLMs, particularly within the realm of educational content derived from textbook:

- **RQ1** - Harnessing the full potential of LLMs: A fine-tuning approach is used to address the first research question. Fine-tuning LLMs on educational content sourced from textbook allowed to maximize their potential not only for open-ended questions but also for quiz-style questions. By tailoring the model's knowledge and understanding to the specific domain of the textbook, the aim is to enhance its performance in generating contextually relevant and accurate responses to a wide range of educational queries.

- **RQ2** - Smaller vs. Bigger LLMs: The findings of this work affirmed the second research question. Smaller fine-tuned LLM consistently outperformed larger, pre-trained LLM when it came to generating responses aligned with the textbook content. This highlights the value of domain-specific fine-tuning, where a smaller model, specifically adapted to the educational context, exhibited superior performance compared to its larger, more generalized counterpart.

- **RQ3** - Resource Efficiency of Smaller LLMs: The results of this study also confirmed the third research question. Smaller fine-tuned LLM not only delivered better results but was also notably less resource-consuming. This efficiency makes it an attractive choice for educational applications, especially in scenarios where computational resources are limited or where faster response times are imperative. Consequently, the research underscores the potential of smaller, domain-specific LLM as a cost-effective and efficient solution for educational content generation and understanding tasks.

Following paragraphs illustrate all steps involved in the fine-tuning process, from the data preprocessing to the results, including some explanation of fine-tuning techniques like Lora [46] and Adapters [47].

3.3.1 Preprocessing

Data preprocessing is a critical and foundational step in fine-tuning large language models (LLMs) for natural language understanding and generation tasks. The process involves a series of operations aimed at preparing the raw input data to be compatible with the LLM architecture, enhancing its quality, and optimizing its utility for the specific task at hand. Common preprocessing tasks include cleaning and filtering out irrelevant or noisy data, handling special tokens, and applying appropriate text normalization techniques like lowercasing can significantly improve model performance.
In this work, data include textbook paragraph from a university course of *Programming Languages*. This choice was taken considering the inference questions about this field.
The data is provided in LaTeX form, so there is a need to prepare and structure textual information for fine-tuning LLMs. Data preprocessing, in this context, involves parsing the LaTeX-formatted documents to extract meaningful text, handling LaTeX-specific syntax, and converting them into tokenized sequences for the LLM. This process preserves the integrity of scientific content while enabling the LLM to process and understand the textual information effectively. Furthermore, it underscores the significance of domain-specific data preprocessing when dealing with specialized content like textbooks.

```
[
    {
        "instruction": "Write a limerick about a
                        pelican.”,
        "input": "",
        "output": "There once was a pelican so fine,
                   \nHis beak was as colorful as
                   sunshine,\nHe would fish all day,\nIn
                   a very unique way,\nThis pelican was
                   truly divine!\n\n\n"
    },
    {
        "instruction": "Identify the odd one out from
                        the group.",
        "input": "Carrot, Apple, Banana, Grape",
        "output": "Carrot\n\n"
    },
]
```

Figure 3.4: lit-gpt - data format

As said in previous sections, the framework used to fine-tune LLM is lit-

gpt [45]. To be used for fine-tuning, this framework requires data in a specific format (Fig. 3.4). They require to fill in the *instruction* field with a specific requirement to be used by the model to generate the *output* field. The *input* field can be left unfilled. The idea of this work is to leverage the fine-tuning phase to make the model learn new notions from the textbook.

```
\section{Data Types}
\label{sec8.1}
Data types are present in programming languages for at least three different
reasons:
\begin{enumerate}
\item At the design level, as support for the conceptual organisation;
\item At the program level, as support for correctness;
\item At the translation level, as support for the implementation.
\end{enumerate}

Before entering into a  detailed discussion of these aspects, which we will do in
the coming sections, we give a definition which, as is often the case with
programming languages, is not formally precise but suffices to explain the
phenomena which we intend studying.
```

Figure 3.5: lit-gpt - Example of latex paragraph

```
###Data Types
Data types are present in programming languages for at least three different
reasons:
1. At the design level, as support for the conceptual organisation;
2. At the program level, as support for correctness;
3. At the translation level, as support for the implementation.
Before entering into a  detailed discussion of these aspects, which we will do in
the coming sections, we give a definition which, as is often the case with
programming languages, is not formally precise but suffices to explain the
phenomena which we intend studying.
```

Figure 3.6: lit-gpt - Example of preprocessed paragraph

Starting from textbook data, paragraphs are provided in latex form (Fig. 3.5). From this type of informations data are preprocessed removing latex command, i.e. $\backslash some\_text$. Paragraph titles are marked using # symbol, like in markdown format to be identifiable. Fig. 3.6 represents an example of preprocessed data.

```
{
    "instruction": "Explain Data Types",
    "input": "",
    "output": "Data types are present in programming languages for at least three different reasons ... "
}
```


Figure 3.7: lit-gpt - Example of preprocessed paragraph in lit-gpt format

Next phase includes filling in the *instruction*, *input* and *output* fields. Those boxes are filled in with the following values (Fig. 3.7):

- *instruction*: this field is a composition of the string "Explain " plus the title of the paragraph.
- *input*: this field has been left unfilled.
- *output*: this field has been filled with the paragraph text.


```
{
    "instruction": "Explain Data Types - Part 1",
    "input": "",
    "output": "Data types are present in programming languages for at least three different reasons ... "
},
{
    "instruction": "Explain Data Types - Part 2",
    "input": "",
    "output": "Before entering into a detailed discussion of these aspects, which we will do in the ... "
},
```


Figure 3.8: lit-gpt - Example of preprocessed paragraph in lit-gpt format after splitting

An important step in the preprocessing phase is to set a limit on the token sequence length of the *output* field. Limiting this length can help to reduce the amount of memory used during fine tuning, thus avoiding errors such as OOM (Out Of Memory). In this work, this limit has been set to a maximum of 1000 tokens for each pattern. Paragraphs exceeding this limit are split into different samples by adding the string " - Part k" after the paragraph title (Fig. 3.8).
The last step of the preprocessing is the tokenisation. This part has been left to the lit-gpt framework, which includes a script that does this.

3.3.2 LoRA - Fine-tuning technique

In the realm of natural language processing, a significant paradigm involves the initial pre-training of models on vast, general-domain datasets, followed by their adaptation to specific tasks or domains. However, as we continue to scale up model sizes, the conventional approach of full fine-tuning, which entails retraining all model parameters, becomes increasingly impractical. To illustrate this, consider GPT-3 with a staggering 175 billion parameters - deploying individual instances of fine-tuned models, each with an equivalent parameter count, incurs prohibitively high costs. In this direction, in literature, were proposed different efficient techniques such as LoRA [46] and LLaMA-Adapter [47]. In this work it's used the LoRA technique, therefore the following paragraph provides a brief overview of such technique.

LoRA - Low-Rank Adaptation

LoRA operates by preserving the pre-trained model's weights while introducing trainable rank decomposition matrices into each layer of the Transformer architecture. This ingenious approach substantially reduces the number of trainable parameters for downstream tasks. When compared to fine-tuning GPT-3 175B with the Adam optimizer, LoRA manages to achieve a remarkable 10,000-fold reduction in trainable parameters and a 3-fold decrease in GPU memory requirements. This superior performance is achieved despite having fewer trainable parameters, a higher training throughput, and, notably, no additional inference latency, unlike adapter-based approaches.

$$\max_{\Phi} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(P_{\Phi}(y_t|x, y_{<t})\right)$$

Figure 3.9: LLMs general objective function

Let's imagine we have a pre-trained autoregressive language model, denoted as $P_{\Phi}(y|x)$ and characterized by the parameter set Φ. To illustrate, this model could be a versatile multi-task learner like GPT, which is based on the Transformer architecture. Now, let's consider the scenario where we want to adapt this pre-trained model for specific text generation tasks (e.g. summarization, machine reading comprehension or converting natural language to SQL). Each of these tasks is defined by a training dataset containing pairs of context and target sequences: $\mathcal{Z} = \{(x_i, y_i)\}_{i=1,..,N}$, where both x_i and y_i are sequences composed of individual tokens. To give examples, in NL2SQL, x_i

represents a natural language query, and y_i corresponds to the SQL command it should generate; in the case of summarization, x_i contains the content of an article, and y_i represents its summary.
In the process of full fine-tuning, the model starts with its pre-trained weights denoted as Φ_0. Subsequently, it undergoes iterative updates, progressing from Φ_0 to $\Phi_0 + \Delta\Phi$, as it continuously follows the gradient to maximize the objective function (Fig. 3.9) of conditional language modeling.

$$\max_{\Theta} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(p_{\Phi_0+\Delta\Phi(\Theta)}(y_t|x, y_{<t})\right)$$

Figure 3.10: LoRA general objective function

A significant limitation of the conventional full fine-tuning process is that it necessitates the acquisition of a distinct parameter set $\Delta\Phi$ for each individual downstream task, and the dimension of $\Delta\Phi$ ($|\Delta\Phi|$) is equivalent to that of the pre-trained model $|\Phi_0|$. Consequently, when dealing with sizable pre-trained models like GPT-3, which boasts approximately 175 billion parameters ($|\Phi_0| \approx 175$Billion), the prospect of storing and deploying numerous independent fine-tuned models becomes not only challenging but potentially infeasible.
The authors propose a new method, wherein the task-specific parameter update $\Delta\Phi$, denoted as $\Delta\Phi(\Theta)$, is represented using a significantly smaller set of parameters Θ, with $|\Theta|$ substantially smaller than $|\Phi_0|$. This transformation effectively frames the process of finding $\Delta\Phi$ as an optimization task over the compact parameter set Θ, offering a practical and efficient solution to the challenge posed by large-scale pre-trained models during fine-tuning.
LoRA introduces a method utilizing a low-rank representation to encode $\Delta\Phi$, offering the dual advantages of computational efficiency and minimal memory usage. When applied to a pre-trained model like GPT-3 175B, this approach can result in the number of trainable parameters, denoted as $|\Theta|$, being reduced to a mere 0.01% of $|\Phi_0|$.

3.3.3 Fine-tuning resource consumption

As said in the above section, fine-tuning technique can be expensive in terms of resource consumption. This paragraph illustrate the costs implicated in this phase that comprehend two different aspects:

- Memory space: the GB used in VRAM
- Time: the time of execution

As yet mentioned, this work made use of a ∼24GB GPU to fine-tune the smaller model of those involved in the inference phase: LLaMA 7B. This due to memory constraints. This learning step make use of a relative small quantity of data: 101 samples in training set and 4 samples in validation set. The test set was not created because the model was tested, as for inference, over the single-choice questions. Data, after preprocessing, occupy only 188KB.

Settings	Training Memory	Training Time	Loss
Default (bfloat16-mixed)	OutOfMemoryError	N/A	N/A
--precision "bf16-true"	20.60 GB	876.30s	0.8696
--quantize "bnb.nf4"	19.62 GB	1320.63s	1.0178
--quantize "bnb.nf4-dq"	19.32 GB	1359.10s	1.0132
--precision "bf16-true" --quantize "bnb.nf4"	13.44 GB	1089.79s	1.0130
--precision "bf16-true" --quantize "bnb.nf4-dq"	13.15 GB	1135.86s	1.0124

Figure 3.11: lit-gpt inference resource consumption

The framework used for fine-tuning is the same of that used for inference: lit-gpt [45]. As for inference the authors provide a little overview of resources consumption (Fig. 3.11).

Learning Rate	Batch Size	Micro Batch Size	Max Iters	Memory (GB)	Time (s)
0.001	16	2	2000	10.95	1188.53
0.001	32	2	1000	17.75	572.75
0.001	32	4	500	17.75	500.00
0.001	32	4	1000	17.75	997.30
0.001	64	4	2000	17.77	1907.33
0.0001	16	4	2000	17.77	2095.63
0.0001	32	2	500	17.75	304.67
0.0001	**32**	**4**	**2000**	**17.76**	**1941.85**
0.0001	64	4	1000	17.76	949.20
0.0001	64	4	2000	17.77	1906.97
0.0001	128	2	2000	17.77	1107.62
0.0001	128	4	1000	17.77	951.99

Table 3.2: Fine-tuning resource consumption

In this work the only setting used is the last one: precision of 16 and double quantization of 4. Table 3.2 reports the memory and time consumption of the best models. These are the results of a fine-tuning involving a number of trainable parameters equals to 4,194,304 over all 7B parameters.

Pre-training and Fine-tuning Times of LLaMA2: A Comparison

		Time (GPU hours)
Llama 2	7B	184320
	13B	368640
	34B	1038336
	70B	1720320
Total		3311616

Figure 3.12: LLaMA-2 pre-training times

The pre-training times for different configurations of the Llama2 model [48] vary significantly (Fig. 3.12). All pre-training was done with an A100-80GB GPU. For the Llama2 7B model, the pre-training time is approximately 184,320 hours, reflecting the extensive computational resources required to train a model of this scale. In comparison, the Llama2 13B model demands even more computational power, with a pre-training time of 368,640 hours, indicating its larger architecture and increased complexity. Finally,

the Llama2 70B model is the most resource-intensive, with an astonishing pre-training time of 1,720,320 hours, highlighting the enormous investment in time and hardware required to develop such a massive language model. These variations in pre-training times underscore the trade-off between model size and the computational resources necessary for training, with larger models offering potential improvements in language understanding and generation at the cost of increased training times.

The Llama2 7B model offers an efficient fine-tuning process with textbook data. When fine-tuned, it exhibits a mean time consumption of just 5-30 minutes. This striking contrast highlights the advantage of using the Llama2 7B model when considering both pre-training and fine-tuning phases, as it is significantly less time-consuming compared to its larger counterparts, the Llama2 13B and 70B models. Despite its smaller architecture, Llama2 7B still manages to deliver strong language capabilities while making more efficient use of resources in real-world applications.

Importantly, the comparative results of inference after fine-tuning suggest that the Llama2 7B model, when fine-tuned, outperforms both the Llama2 13B and 70B models when used in their pre-trained states (see Sec. 3.3.4 vs. Sec. 3.2). This outcome underscores the significance of the fine-tuning process, demonstrating that the Llama2 7B model's efficiency in balancing pre-training and fine-tuning leads to superior performance when compared to its larger counterparts, the Llama2 13B and 70B models, which rely solely on their pre-trained capabilities. Thus, in terms of both computational efficiency and practical effectiveness, the Llama2 7B model emerges as a compelling choice.

3.3.4 Results

The results of fine-tuning LLaMA2 7B have yielded impressive and promising outcomes. This advanced language model, after undergoing fine-tuning, exhibits a remarkable ability to understand and generate human-like text in a in the domain and application of Programming Languages. The fine-tuning process has further refined its ability to comprehend nuanced context, adapt to specific tasks, and generate contextually relevant and contextually accurate responses, marking a significant step forward in the capabilities of AI language models.

The fine-tuning of the model over three chapters of a university-level programming languages textbook has produced noteworthy results, as depicted in the table below. During this process, various hyperparameters were carefully tuned to optimize performance. The model's effectiveness was evaluated through a battery of 25 single-choice questions, covering a wide array of topics from the textbook. The table summarizes key hyperparameters used in the fine-tuning process, including learning rate, batch size, and training steps, alongside the results achieved in the test consisting of these 25 questions. The results exhibit the model's proficiency in comprehending and responding to questions related to programming languages, showcasing its ability to provide accurate and contextually relevant answers within this specialized domain.

As the questions used to test the models are the same as those used to test the results of the inference section, the results of the models shown in the table below are the only ones that give an accuracy equal to or better than the LLaMA2 70B model (60% of accuracy). Other configurations have been tested but result in a poor performing model.

Learning Rate	Batch Size	Microbatch Size	Training Steps	Test Results (%)
0.001	16	2	2000	17/25=68%
0.001	32	2	1000	16/25=64%
0.001	32	4	500	16/25=64%
0.001	32	4	1000	16/25=64%
0.001	64	4	2000	16/25=64%
0.001	128	4	500	15/25=60%
0.0001	16	2	1000	15/25=60%
0.0001	16	4	100	15/25=60%
0.0001	16	4	500	15/25=60%
0.0001	16	4	2000	16/25=64%
0.0001	32	2	500	16/25=64%
0.0001	32	4	100	15/25=60%
0.0001	32	4	500	15/25=60%
0.0001	**32**	**4**	**2000**	**18/25=72%**
0.0001	32	4	5000	15/25=60%
0.0001	64	2	1000	15/25=60%
0.0001	64	4	100	15/25=60%
0.0001	64	4	500	15/25=60%
0.0001	64	4	1000	16/25=64%
0.0001	64	4	2000	16/25=64%
0.0001	128	2	500	15/25=60%
0.0001	128	2	1000	15/25=60%
0.0001	128	2	2000	16/25=64%
0.0001	128	4	100	15/25=60%
0.0001	128	4	1000	16/25=64%
0.0001	128	4	2000	15/25=60%

Table 3.3: Fine-tuning - results

These results (Tab. 3.3) demonstrate the model's remarkable adaptability to domain-specific knowledge and its potential to assist students and professionals in mastering programming languages concepts with a high degree of accuracy.

Conclusions and Future Works

In conclusion, this study has focused on the utilization of LaTeX-formatted textbook data for fine-tuning Large Language Models (LLMs) and underscored the critical importance of domain-specific data preprocessing when dealing with specialized content like textbooks. The research has demonstrated the adaptability of fine-tuned LLMs in addressing single-choice questions, a task for which pretrained LLMs exhibit suboptimal performance. Furthermore, the study has highlighted the contrasting resource consumption between a less computationally intensive fine-tuned small LLM (7 billion parameters) and a resource-intensive pretrained large LLM (70 billion parameters). These findings provide valuable insights for practitioners seeking to optimize model selection and resource allocation in real-world applications, contributing to the ongoing advancement of natural language processing solutions across diverse domains.
In addition to the findings presented in this study, there are several promising directions for future research:

- Multimodal Integration: Investigate the incorporation of visual elements, such as images and diagrams from textbooks, into the fine-tuning process. This could enable LLMs to better comprehend and generate text in conjunction with visual information, enhancing their performance on tasks involving multimodal content understanding. Another interesting point can be to extract informations from video and audio recordings of university courses lessons.

- Different fine-tuning techniques: trying different fine-tuning techniques can help to explore how these LLMs can learn and what is the best technique to use for each individual model and domain.

- Ethical and Bias Considerations: Investigate and address potential biases in the fine-tuned LLMs, especially when dealing with educational content. Develop methods for mitigating biases and ensuring fair and inclusive responses to user queries.

- User-Centric Evaluation: Conduct user studies and evaluations to assess the real-world utility of fine-tuned LLMs in educational settings. Gather feedback from students, teachers, and educational professionals to refine the model's design and capabilities.

- Prompt-Engineering: Prompt engineering is the art and science of crafting precise and effective instructions or queries to elicit desired responses from artificial intelligence models, such as language models and chatbots. It involves formulating prompts that are clear, contextually relevant, and tailored to the task at hand. Effective prompt engineering requires a deep understanding of the model's capabilities, limitations, and biases, as well as the ability to fine-tune prompts to achieve specific goals. Whether it's generating creative content, conducting research, or solving complex problems, skilled prompt engineers play a crucial role in harnessing the potential of AI to deliver meaningful and accurate results. In the rapidly evolving field of AI, mastering prompt engineering is essential for optimizing the interaction between humans and machines, enabling AI systems to assist, augment, and enhance various aspects of our lives.

By pursuing these avenues of research, we can continue to advance the capabilities of fine-tuned LLMs in educational contexts and address the evolving needs of learners and educators in an increasingly digital and data-driven educational landscape.

Bibliography

[1] Henner Gimpel et al. "Unlocking the power of generative AI models and systems such as GPT-4 and ChatGPT for higher education". In: (2023).

[2] Jackson Stokes. *A guide to language model sampling in AllenNLP*. https://blog.allenai.org/a-guide-to-language-model-sampling-in-allennlp-3b1239274bc3.

[3] Hoo-Chang Shin, Le Lu, and Ronald M Summers. "Natural language processing for large-scale medical image analysis using deep learning". In: *Deep learning for medical image analysis* (2017), pp. 405–421.

[4] *ChatGPT*. https://chat.openai.com.

[5] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[6] Wayne Xin Zhao et al. "A survey of large language models". In: *arXiv preprint arXiv:2303.18223* (2023).

[7] *GPT-4 API*. https://openai.com/blog/gpt-4-api-general-availability.

[8] *OpenAI*. https://openai.com.

[9] Mingyu Zong and Bhaskar Krishnamachari. "A survey on GPT-3". In: *arXiv preprint arXiv:2212.00857* (2022).

[10] Jared Kaplan et al. "Scaling laws for neural language models". In: *arXiv preprint arXiv:2001.08361* (2020).

[11] Long Ouyang et al. "Training language models to follow instructions with human feedback". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 27730–27744.

[12] Henry Lai. *What Are the Data-Centric AI Concepts behind GPT Models?* https://towardsdatascience.com/what-are-the-data-centric-ai-concepts-behind-gpt-models-a590071bb727.

[13] Alec Radford et al. “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8 (2019), p. 9.

[14] Colin Raffel et al. “Exploring the limits of transfer learning with a unified text-to-text transformer”. In: *The Journal of Machine Learning Research* 21.1 (2020), pp. 5485–5551.

[15] Mike Lewis et al. “Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension”. In: *arXiv preprint arXiv:1910.13461* (2019).

[16] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. “Building a large annotated corpus of English: The Penn Treebank”. In: (1993).

[17] Stephen Merity et al. “Pointer sentinel mixture models”. In: *arXiv preprint arXiv:1609.07843* (2016).

[18] Leo Gao et al. “The pile: An 800gb dataset of diverse text for language modeling”. In: *arXiv preprint arXiv:2101.00027* (2020).

[19] Kishore Papineni et al. “Bleu: a method for automatic evaluation of machine translation”. In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 2002, pp. 311–318.

[20] Chin-Yew Lin. “Rouge: A package for automatic evaluation of summaries”. In: *Text summarization branches out*. 2004, pp. 74–81.

[21] Dan Hendrycks et al. “Measuring coding challenge competence with apps”. In: *arXiv preprint arXiv:2105.09938* (2021).

[22] Mark Chen et al. “Evaluating large language models trained on code”. In: *arXiv preprint arXiv:2107.03374* (2021).

[23] Jacob Austin et al. “Program synthesis with large language models”. In: *arXiv preprint arXiv:2108.07732* (2021).

[24] Dan Hendrycks et al. “Measuring massive multitask language understanding”. In: *arXiv preprint arXiv:2009.03300* (2020).

[25] Mirac Suzgun et al. “Challenging big-bench tasks and whether chain-of-thought can solve them”. In: *arXiv preprint arXiv:2210.09261* (2022).

[26] Percy Liang et al. “Holistic evaluation of language models”. In: *arXiv preprint arXiv:2211.09110* (2022).

[27] Ramon Dijkstra et al. *Reading Comprehension Quiz Generation using Generative Pre-trained Transformers*. 2022.

[28] Tareq Rasul et al. “The role of ChatGPT in higher education: Benefits, challenges, and future research directions”. In: *Journal of Applied Learning and Teaching* 6.1 (2023).

[29] Joseph Weizenbaum. “ELIZA – a computer program for the study of natural language communication between man and machine”. In: *Communications of the ACM* 9.1 (1966), pp. 36–45.

[30] Donald L Bitzer et al. “The Plato System and Science Education.” In: (1970).

[31] Jaeho Jeon. “Exploring AI chatbot affordances in the EFL classroom: Young learners’ experiences and perspectives”. In: *Computer Assisted Language Learning* (2021), pp. 1–26.

[32] Serge Bibauw et al. “Dialogue systems for language learning: A meta-analysis”. In: *Language Learning & Technology* 26.1 (2022).

[33] Jaeho Jeon and Seongyong Lee. “Large language models in education: A focus on the complementary relationship between human teachers and ChatGPT”. In: *Education and Information Technologies* (2023), pp. 1–20.

[34] Enkelejda Kasneci et al. “ChatGPT for good? On opportunities and challenges of large language models for education”. In: *Learning and Individual Differences* 103 (2023), p. 102274.

[35] Jirapond Muangprathub, Veera Boonjing, and Kosin Chamnongthai. “Learning recommendation with formal concept analysis for intelligent tutoring system”. In: *Heliyon* 6.10 (2020).

[36] Pablo Arnau-González et al. “Towards automatic tutoring of Math Word Problems in Intelligent Tutoring Systems”. In: *IEEE Access* (2023).

[37] Erik Nijkamp et al. “Codegen: An open large language model for code with multi-turn program synthesis”. In: *arXiv preprint arXiv:2203.13474* (2022).

[38] Md Doulotuzzaman Xames and Jannatul Shefa. “ChatGPT for research and publication: Opportunities and challenges”. In: *Available at SSRN 4381803* (2023).

[39] Fernando Antonio Flores Limo et al. “Personalized tutoring: ChatGPT as a virtual tutor for personalized learning experiences”. In: *Social Space* 23.1 (2023), pp. 293–312.

[40] Rehan Ahmed Khan et al. “ChatGPT-Reshaping medical education and clinical management”. In: *Pakistan Journal of Medical Sciences* 39.2 (2023), p. 605.

[41] Walid Hariri. "Unlocking the Potential of ChatGPT: A Comprehensive Exploration of its Applications, Advantages, Limitations, and Future Directions in Natural Language Processing". In: *arXiv preprint arXiv:2304.02017* (2023).

[42] Daniel M Ziegler et al. "Fine-tuning language models from human preferences". In: *arXiv preprint arXiv:1909.08593* (2019).

[43] Nisan Stiennon et al. "Learning to summarize with human feedback". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 3008–3021.

[44] *LLaMA2 chatbot*. `https://llama2.ai`.

[45] *lit-gpt*. `https://github.com/Lightning-AI/lit-gpt`.

[46] Edward J Hu et al. "Lora: Low-rank adaptation of large language models". In: *arXiv preprint arXiv:2106.09685* (2021).

[47] Renrui Zhang et al. "Llama-adapter: Efficient fine-tuning of language models with zero-init attention". In: *arXiv preprint arXiv:2303.16199* (2023).

[48] Hugo Touvron et al. "Llama 2: Open foundation and fine-tuned chat models". In: *arXiv preprint arXiv:2307.09288* (2023).

Ringraziamenti

Un sentito ringraziamento va al mio relatore Saverio Giallorenzo e al mio correlatore Maurizio Gabbrielli che mi hanno seguito, con disponibilitá e gentilezza, in ogni passo della realizzazione dell'elaborato, fin dalla scelta dell'argomento, facendomi vedere la ricerca come un punto di svolta ma anche di partenza per la mia futura carriera.
Vorrei anche ringraziare i due professori Cosimo Laneve e Giuseppe Lisanti e la dottoressa Adele Veschetti con cui ho lavorato durante questo ultimo anno. Grazie a loro ho avuto l'occasione di accrescere le mie competenze comunicative in un ambiente inclusivo e collaborativo che ha reso il mio lavoro piacevole e costruttivo.
Voglio fare un grosso ringraziamento anche ai miei fantastici compagni di universitá: Filippo, Nicoló, Isabella e Alfonso. Abbiamo condiviso risate, ansie da esami e una quantitá incredibile di caffé durante questi due anni. La vostra compagnia e il nostro sostegno reciproco hanno reso il percorso universitario molto piú sopportabile e divertente. Grazie per essere stati una parte importante della mia esperienza accademica e per aver reso tutto piú significativo.
Non per ultimo, un grazie speciale alle mie amiche di sempre, Francesca e Martina, che hanno alleggerito i momenti piú pesanti e mi hanno spronata a dare sempre il meglio.