Research Article

A Primer on Large Language Models and their Limitations

Sandra Johnson[1], David Hyland-Wood[2]

1. School of Mathematical Sciences, Queensland University of Technology, Australia; 2. School of Electrical Engineering and Computer Science, The University of Queensland, Brisbane, Australia

This paper provides a primer on Large Language Models (LLMs) and identifies their strengths, limitations, applications and research directions. It is intended to be useful to those in academia and industry who are interested in gaining an understanding of the key LLM concepts and technologies, and in utilising this knowledge in both day to day tasks and in more complex scenarios where this technology can enhance current practices and processes.

Corresponding author: Sandra Johnson, sandra.johnson@qut.edu.au

1. Introduction

The world of artificial intelligence AI is increasingly penetrating all aspects of our personal and professional lives. This proliferation of AI tools and applications are being met with a mixture of excitement, scepticism and even dread[1]. Excitement at the seemingly endless potential of AI applications such as LLMs, especially when they are integrated "within broader systems"[2], scepticism as the realisation dawns that LLMs are in fact fallible as evidenced by hallucinations and hence not the golden bullet that can solve all problems[3][4], and a feeling of dread for those who believe that LLMs and AI have the potential to detrimentally impact our lives and make people redundant[1].

The ability of some LLMs to pass Theory of Mind (ToM) [5][6] and Turing Tests [7][8] suggests support for the Computational Theory of Mind (CTM), that cognition may be substrate independent. These findings challenge biological essentialism and open new avenues for creating sophisticated AI systems capable of human-like reasoning and interaction. Viewed another way, these studies could be taken to

provide evidence for those critical of both the Turing Test and Theory of Mind tests in assessing cognition in humans and animals.

However, it should be noted that LLMs by themselves have no self monitoring (also called phenomenal consciousness or subjective experience) or internal, updatable model of their external environment (that is, a model of itself as a being in a world). Both of these conditions are required in some reasonable theories consciousness[9]. Those omissions alone may be taken as evidence against LLMs having any form of consciousness as that term is currently understood.

We begin this paper by providing an overview covering the basics of LLM in Section 2: An Overview of LLMs, in order to provide a common understanding and vocabulary for these natural language modelling approaches.

In Section 3: LLMs Orchestrated with Other Technologies, we note that LLMs may be combined with other, more traditional, information retrieval technologies to create full-featured systems that can adapt LLMs to their own use cases.

In Section 4: Risks and Mitigations we identify a series of common risks when working with LLMs.

LLMs, like any technology, are tools to be used with awareness and even caution. They may produce content that we wish they would not. Language models pre-trained on large text corpora that are highly likely to contain toxic and inappropriate content, are known to pass these biases on, or worse amplify them, when generating query responses and text[10]. The bias can present itself in various forms such as discrimination based on race, gender, disability, nationality or religion[11][12]. To this end researchers developed a challenge dataset, CrowS-Pairs, crowd sourced using Amazon Mechanical Turk (MTurk), to measure the extent of biasin masked language modelling (MLM)[11]. More recent approaches to address such biases were collected by Gallegos et al in[10].

Similar to the problems of socially inappropriate content are problems related to the accessibility of legally inappropriate content in training data. Most foundational LLMs are pre-trained to avoid assisting criminal activities such as composing blackmail letters or providing instructions on how to commit crimes, but prompt engineering may be used to work around these built-in checks and balances. We address these issues and various approaches to their resolution in Section 4.3: Jailbreak Attacks.

Finally, we conclude by summarising the key points related to LLMs and their usage.

2. An Overview of LLMs

The emergence of LLMs is preceded by an extensive body of research in language modelling[13] where word sequences were scored with the aid of a probabilistic model possibly dating back to 1976[14][13]. Natural language processing (NLP) modelling further evolved over time with recurrent neural networks (RNNs) becoming the model of choice for autoregressive language modelling tasks such as translation. However, RNNs typically process one token at a time because they are designed for sequential processing, making parallel processing hard to achieve and limiting the capture of long range sequences. The latter is often referred to as the "vanishing gradient problem"[15][16].

RNNs are further constrained due to their heavy use of computing resources. Although more computing power is available these days, relieving some of the constraints, the key change maker was the publication of the seminal paper "Attention Is All You Need"[17]. The authors introduced a model called the *Transformer*, signalling the arrival of what is commonly known as 'transformer architecture', which revolutionised NLP. As the title implies the researchers discovered that it was possible to rely solely on self-attention and feedforward layers without recurrent connections as in RNNs[17]. Moreover, with the introduction of *Reformer*, a revised version of Transformer, the same performance was obtained in a far more memory-efficient way and on much longer sequences of words[18].

Several LLMs are based on this architecture, such as bidirectional encoder representations from transformers (BERT)[19] and variations of BERT: a lite BERT (ALBERT), Robustly optimised BERT approach (RoBERTa) developed by researchers at Google; Large Language Model Meta AI (LLaMA) developed by Meta[12] and the Generative Pre-trained Transformer (GPT) models from OpenAI such as GPT-3, GPT-4, GPT-4o, GPT-1o preview and GPT-1o mini. The proliferation of large models over time are captured in Figure 1 (page 3), for the period 2019 to early 2024[20]. Moreover, the diagram distinguishes between open- and closed-source models with the former above the timeline and the latter below the timeline, showing a clear trend towards open-source models[20].

Despite the increasing number of open-source models compared to closed-source models, real concerns exist for the integration of LLMs into predominantly open source systems. Bloomberg notes[21], "To deliver viable alternatives that compete with centralized, closed-source solutions, decentralized AI teams will need to innovate on model architectures and leverage model coordination

platforms. In practice, this will enable ML researchers and engineers to broadly experiment with a wide variety of models aimed at different application verticals. This largely reflects how crypto networks can accelerate AI development."

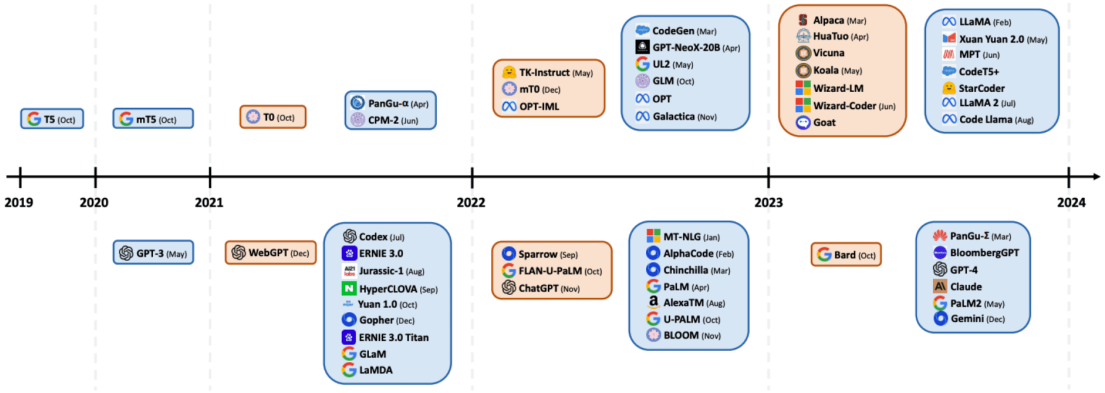

Figure 1. Timeline of LLM releases: blue rounded rectangles are 'pre-trained' models, while orange rectangles are 'instruction-tuned' models. Models above the line indicate open-source availability, and those below the line are closed-source (image from Naveed, H., et al.[20])

2.1. AI Project Development utilising LLMs

The hype around AI and particularly LLMs sparked the realisation that there are real benefits to be had if this technology can be integrated in every day personal activities and in businesses and organisations to improve productivity through automation of repetitive trivial tasks, and consequently enable employees, students and researchers to dedicate more time on interesting and complex tasks.

When embarking on an AI/LLM project to harness the power of this technology, we observe that the life cycle of such a project consists mainly of four distinct phases[22]:

1. Project scoping
2. Model selection
3. Model adaption and alignment
4. Application integration

doi.org/10.32388/NHJYVS

Defining a detailed and accurate description of the use case is the first and crucial step in the project life cycle, ensuring a clear and concise scope of the project at hand.

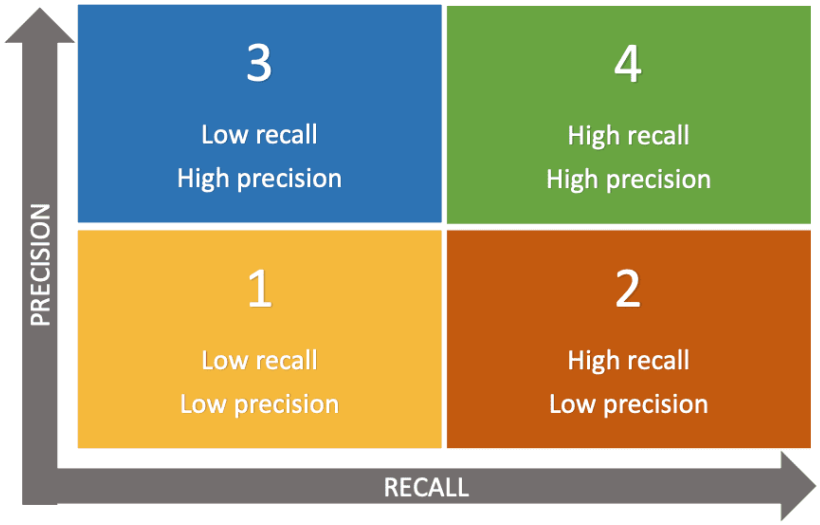

Figure 2. Generative AI use case categorisation

When defining the use cases, we find that they require different degrees of precision and recall to satisfy the main task or objective. Project objectives and use cases may be viewed as roughly falling in one of the quadrants in Figure 2, page 4, indicating the combination of precision and recall required for the project tasks. High precision is crucial when the output needs to be highly accurate and errors may have major consequences. Whereas high recall is essential when capturing as much relevant information as possible is the primary focus. Most use cases are likely to require at least moderate recall and precision.

- *Quadrant 1: Low recall, low precision*

  Use cases in this category are typically of a creative nature, where the goal is not to get high quality or exhaustive content but simply to spark ideas and inspire. In a creative environment, generated completions may even be incomplete, vague, or somewhat nonsensical, but they can still serve as useful prompts for creative thinking. Researchers recently explored the concept of enhancing secondary school students' creative writing skills by leveraging AI in the language classroom[23]. In another study researchers experimented with AI to aid song composition, including the song's structure, harmony, lyrics, and hook melody[24].

- *Quadrant 2: High recall, low precision*

  Use cases in this quadrant are similar to the way we use search engines, e.g. searching for product support or services. Instead products such as MetaMask or Linea could integrate AI into customer support. Here high recall is necessary to ensure that as many customer issues as possible are addressed, while precision needs to be only moderate to low, since some responses could be reasonably generic, such as suggesting appropriate next steps for investigation or resolution. The extent to which generic advice can be given would depend on the product and the support requests[25].

- *Quadrant 3: Low recall, high precision*

  Language translation falls into this quadrant, in the moderate to high precision and moderate recall area. In some situations it is important that the translation is accurate and precise rather than exhaustive in capturing all nuances of the source text. On the other hand, although precision is important for fluency and accuracy in certain instances, some variations in wording may be acceptable[26].

- *Quadrant 4: High recall, high precision*

  For use cases in healthcare and medical diagnosis it is critical to have both high precision and high recall[27]. Similarly, legal tasks require high precision and moderate to high recall, depending on the task, e.g. in legal reasoning, "generating arguments for and against particular outcomes" would need accurate references to relevant cases and judgements, but the formulation of arguments can accomodate some creativity[28]. Nonetheless in both instances the results need be reviewed by relevant experts, before implementing any generated advice or diagnosis. These types of use cases would ultimately benefit from more complex solutions that combine LLMs with retrieval augmented generation (RAG) via orchestration. We can think of use cases in this quadrant as more analogous to the way we previously heavily relied on relational databases to achieve the same ends.

2.2. Choosing a Foundational LLM

The choice of which LLM to use in an AI project for a given task or functionality may appear daunting due to the sheer number of LLMs currently available, and the emergence of a seemingly endless stream of new LLMs, including updated versions of existing LLMs, each purporting to improve on the

previous version (Figure 3, page 5)[29]. Moreover, pre-training and fine tuning techniques for models differ depending on the desired capabilities of the language model[30].

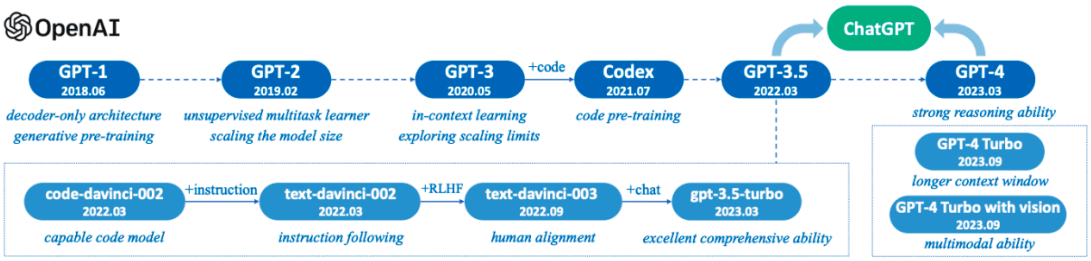

Figure 3. Technical evolution of the OpenAI GPT-series models[29]

LLMs have evolved from relatively simple language tasks such as text generation to models capable of performing more complex tasks, such as those illustrated in Figure 4, page 6[29].

When weighing up the options, we need to take several aspects into consideration. The number of parameters in models vary widely and can limit the range of devices it can run on, as well as the task objectives and applications that are best suited to a particular LLM. Smaller models do not necessarily perform worse than large models, especially if the model is being optimised to perform a specific task well, rather than aiming to cater for multiple use cases. Determining model size is just one consideration when choosing an existing LLM, adapting an existing model, or building a new fit-for-purpose model from scratch.

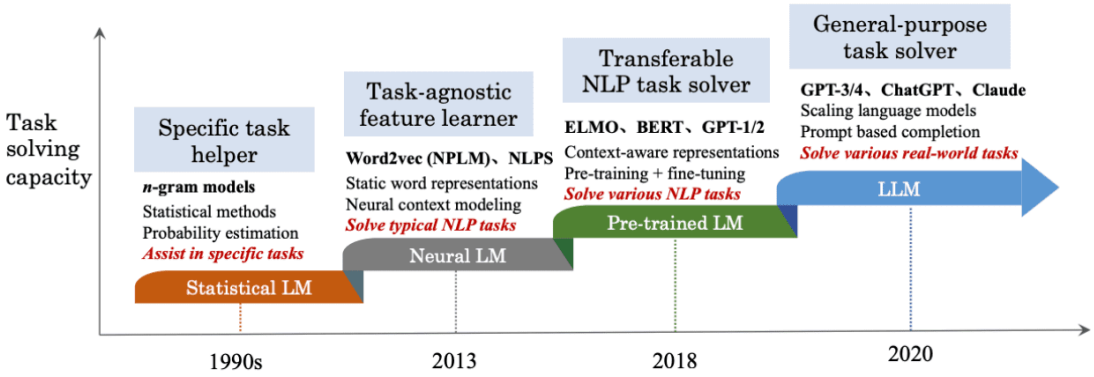

Figure 4. Task solving[29]

doi.org/10.32388/NHJYVS

It is helpful to keep the Transformer LLM framework Figure 5(a) and high level architecture Figure 5(b), page 6, in mind when choosing a suitable foundational LLM model.

The encoder component encodes the model input, entered as a "prompt", with contextual understanding and produces one vector per input token. The decoder processes the input tokens and uses the contextual understanding from the encoder to produce new tokens.[1] Prompt engineering is described in Section 2.6.1 on page 18.

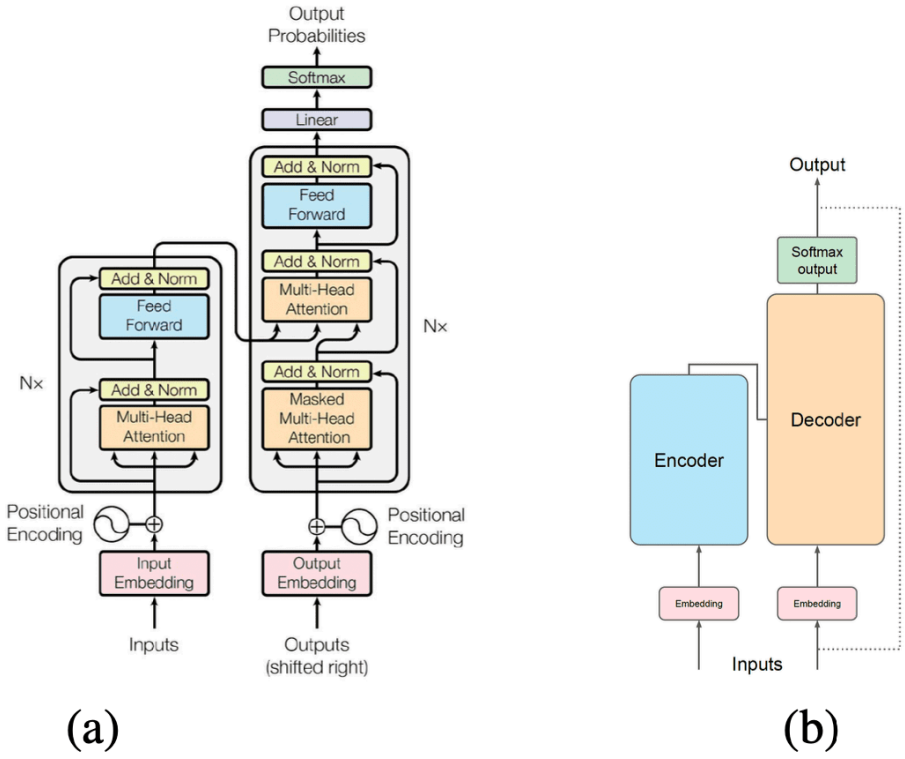

Figure 5. (a) Attention framework (image by Vaswani, A., et al.[17]) (b) Encoder Decoder components of transformer architecture (image by DeepLearning.AI[22])

We can categorise foundational Transformer LLM models as essentially one of three types: decoder-only, encoder-only, or encoder decoder models.

2.2.1. Decoder-only models

Decoder-only models are autoregressive models pre-trained to predict the next token based on previous tokens, making them well suited to text generation (e.g. for creative writing or content

generation), autocompletion (e.g. autocompletion of sentences or lines of code), language translation, and text summarisation.

There are several well known decoder-only LLMs, such as the GPT series of models from OpenAI, LLaMA and open pretrained transformer (OPT) from Meta, Claude from Anthropic, peer-to-peer (p2p) from Google and Gopher from DeepMind.

Causal language modelling (CLM), a self-supervised learning approach, is the preferred method for training decoder-only LLMs. CLMs apply autoregressive modelling to input data to predict future tokens based on past tokens, an approach that is common in time series prediction and recurrent neural networks. Decoder LLMs leverage their uni-directional, autoregressive nature to learn language patterns (Figure 6, page 7). CLM token prediction is uni-directional because only the past tokens are used to predict the next tokens.

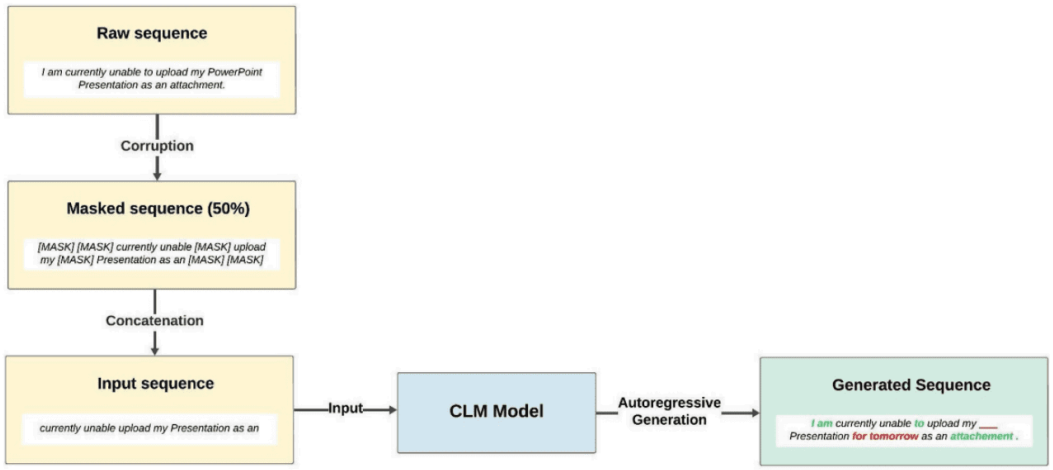

Figure 6. Causal language modelling (CLM) for decoder-only models (image by Clark, K.,et al.[31])

2.2.2. Encoder-only models

Encoder-only models are auto-encoding models, well suited to tasks that involve understanding and extracting meaning from text, such as word classification, named entity recognition, question answering and sentiment analysis.

Key foundational encoder-only models include BERT[19], variant RoBERTa, and ELECTRA[32].

doi.org/10.32388/NHJYVS

MLM, most frequently used to pre-train encoder models[19][26], is a self-supervised learning technique that randomly masks tokens in an input sequence with the aim of learning the masked tokens based on the surrounding context provided by the unmasked tokens[11][31]. Hence MLM differs from CLM by using unmasked tokens both before and after masked tokens, providing a bi-directional understanding of context instead of being limited to the words that precede it (Figure 7, page 8).

Alternatively, techniques such as replaced token detection (RTD), which was used to train ELECTRA, may be used[32]. With RTD instead of masking tokens as in MLM, a small generator model replaces some tokens with plausible alternatives and the encoder (discriminator) is then trained to detect the tokens that have been replaced[31].

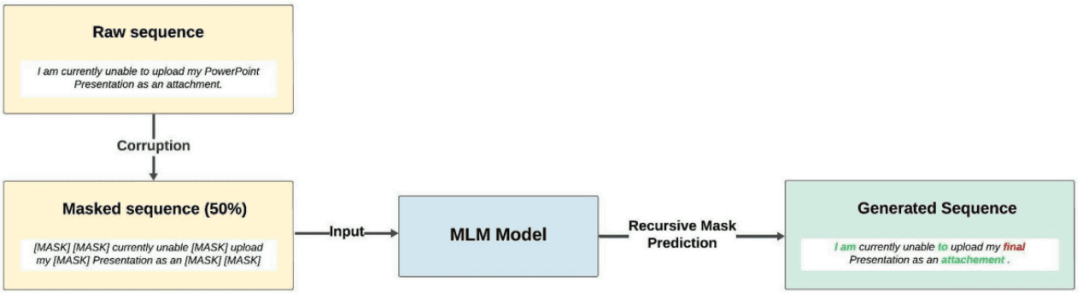

Figure 7. Masked language modelling (MLM) for encoder-only models (image by Clark, K.,et al.[31])

2.2.3. Encoder Decoder models

Encoder-decoder models are sequence-to-sequence models, and suited to tasks that require both understanding and generation of text, such as translation, summarisation, question answering and dialogue systems.

Two notable sequence-to-sequence LLMs are T5[33] and BART[34] with both aiming to denoise corrupted inputs, via slightly different pre-training approaches.

For T5 the encoder is pre-trained using span corruption, where random sequences of tokens are masked and replaced with unique Sentinel tokens ($< x >$) that are added to the vocabulary. The decoder then reconstructs the masked token sequences in an autoregressive manner (Figure 8, page 8). On the other hand BART uses more varied forms of corruption, including sentence permutation. Another interesting encoder-decoder model is the multilingual variant of T5, mT5[35].

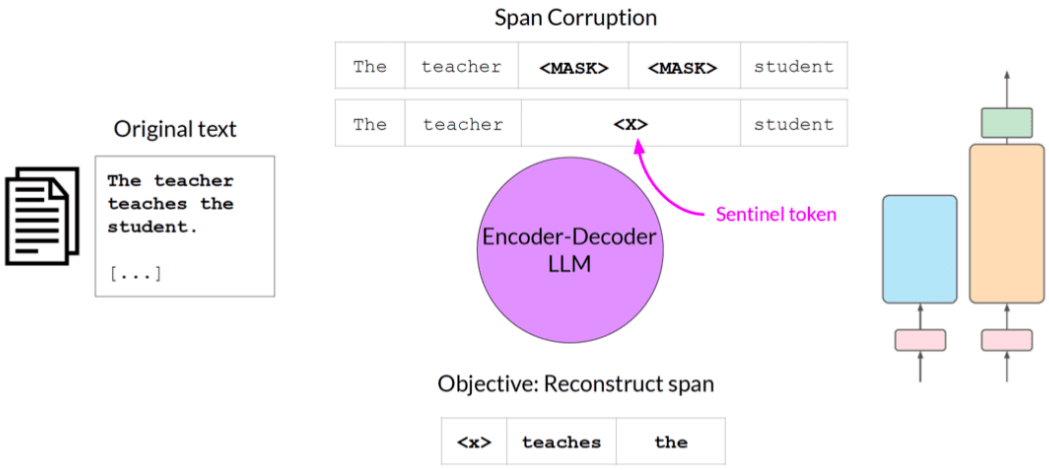

Figure 8. Encoder-decoder model (image by DeepLearning.AI[22])

2.3. *Pre-training Foundational LLMs*

LLMs are pre-trained on vasts amounts of textual data using a variety of strategies and techniques, which is often followed by more specific fine-tuning of the model to suit its intended use[36]. Interestingly, LLaMA was trained exclusively on publicly available data sources[12] (Figure 9).

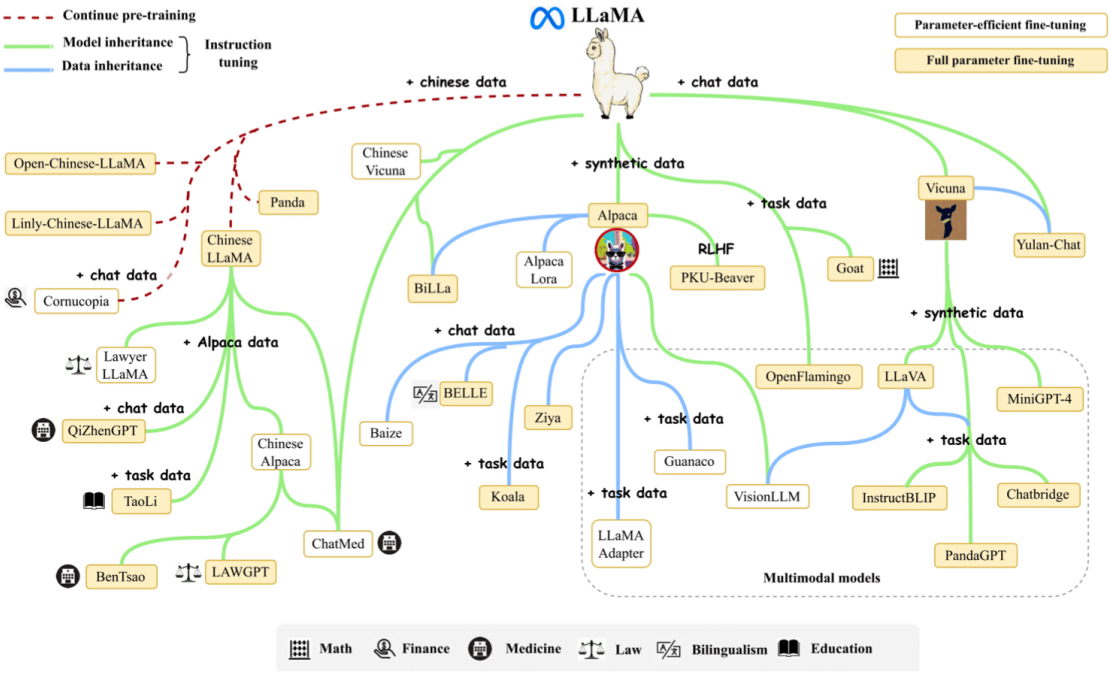

Figure 9. Overview of research work on LLaMA showing data sources, pre-training, fine-tuning and instruction tuning (image from[29])

The general trend has been to train ever larger models because large pre-trained Transformer models were found to be capable of performing tasks for which they had not been specifically trained on[30]. Conversely, Hoffman et al.[37] found that training a smaller model on more data for a given compute budget is more performant than only increasing model size while keeping the size of the training data unchanged. However, focussing on the optimal combination of model size and training dataset size does not take into account the importance of the speed of inference[12]. Instead Touvron et al. [12] concluded that training smaller models for longer results in faster inference.

doi.org/10.32388/NHJYVS

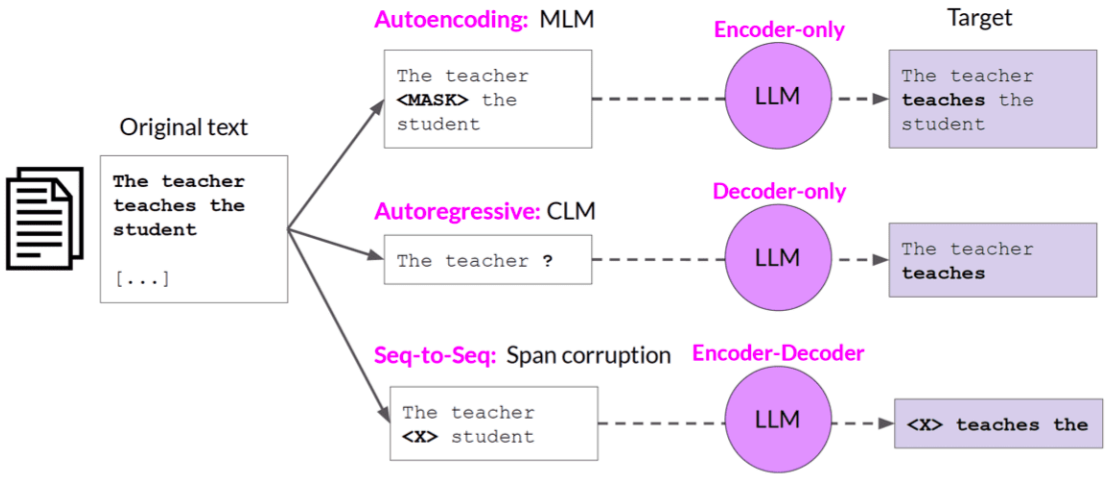

Figure 10. Model architecture and pre-training objectives (image by DeepLearning.AI[22])

Pre-training of LLMs is performed using data labels. Adding labels to data prior to training (supervised learning) typically requires human annotation which is infeasible when training on very large corpora. Supervised learning for LLMs is more typically used when training a model for a specific task, such as fine-tuning a model. Unsupervised learning on the other hand is a well known machine learning technique to learn patterns and structure from unlabelled data through methods like clustering and dimensionality reduction. self-supervised learning (SSL) may be viewed as a 'blend' of supervised and unsupervised learning. During SSL the unlabelled data itself provides the supervision by generating labels from the input data and this can be done in several ways, as discussed below[38].

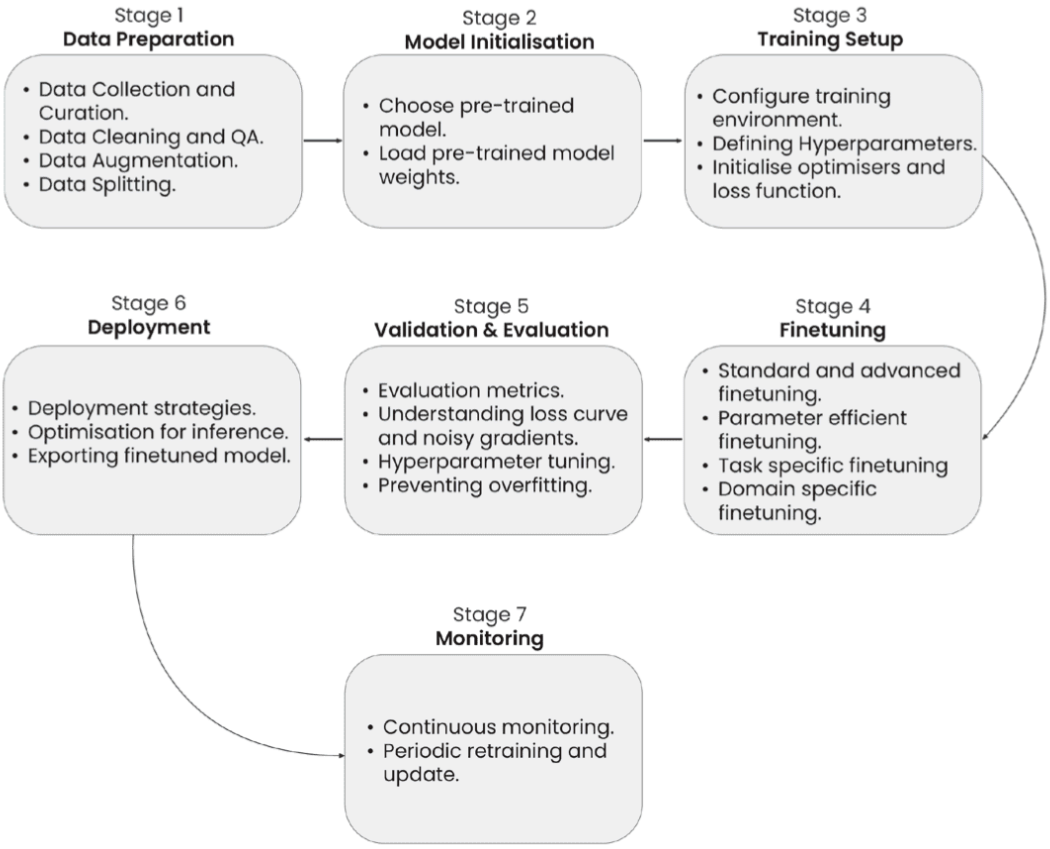

Figure 11. Pre-training and fine-tuning Large Language Models (LLMs), illustrating the seven essential stages (image from[39])

2.3.1. Self-supervised learning

SSL is the most popular machine learning technique for LLMs. This approach is often referred to as the "dark matter of intelligence"[38] and includes learning methods such as CLM, MLM, Span-Level Masking, and Contrastive Learning, where models learn without the need to explicitly apply external labels to the data.

ELECTRA, also an encoder-only mode like BERT uses an alternative pre-training method to MLM called "replaced token detection"[32].

Various SSL approaches have been used in training foundational LLMs (Table 1 on page 11). Moreover, as can be seen from summary table 1, page 11, some notable LLMs combine multiple self-supervised approaches to leverage the strengths of each method, e.g. BART, BERT and T5[33][34].

Approach	Description	Notable Models
Masked Language Modeling (MLM) - Figs. 7 & 10	Predict masked tokens in input sequence (bi-directional) [31]	BERT, RoBERTa, ALBERT
Causal Language Modeling (CLM) - Figs. 6 & 10	Predict next token from previous tokens (uni-directional)	GPT series, Transformer-XL
Permuted Language Modeling	Predict tokens in random order [31]	XLNet
Next Sentence Prediction (NSP)	Predict if one sentence follows another	BERT
Sentence Order Prediction (SOP)	Predict correct order of sentence pairs	ALBERT
Span-Based Masking - Fig. 10	Predict missing spans of tokens	T5, BART
Denoising Autoencoders	Reconstruct original text from corrupted input	T5, BART
Contrastive Learning	Differentiate similar and dissimilar inputs	SimCSE

Table 1. Summary of self-supervised learning (SSL) approaches

2.4. Adapting LLMs for Specific Use Cases

Large pre-trained Transformer models were found to be capable of performing tasks for which they had not been specifically trained on[30]. This is known as "zero-shot" inference[30]. However, when the output from the LLM for a certain task is less than satisfactory, there are two main techniques to achieve better results: in-context learning and fine-tuning.

2.4.1. In-context learning

In-context learning (ICL) refers to the capability of pre-trained LLMs to perform new tasks by leveraging information provided within the context window, without any explicit parameter updates or fine-tuning[36].

Instead of adjusting weights through gradient descent, the model adapts its behaviour based on examples, instructions, or demonstrations included in the prompt[40]. Figure 12 (page 12) shows

prompting with none, one and two examples in the context window. This strategy allows LLMs to generalise to a wide range of tasks using natural language interactions[36].

Figure 12. Example of in-context learning (ICL) (image by DeepLearning.AI[41])

In few-shot prompting the model uses the examples provided in the context window to infer the task's structure and apply it to new inputs. The study by[36] (Figure 13, page 13) shows in-context learning curves with few-shot learning of a simple task. We can observe that model performance improves with increases in both model size and number of examples in the context window[36].

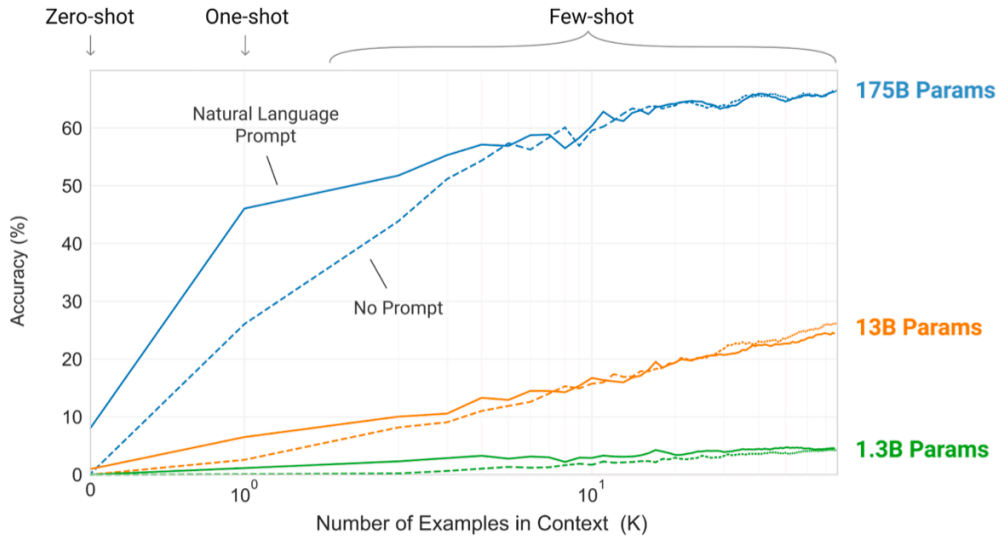

Figure 13. In-context learning performance with different model sizes and number of examples[36]

Chain-of thought-prompting[40] is another effective ICL technique to help LLMs perform complex reasoning required for tasks such as arithmetic computations that LLMs have been known to struggle with. In chain-of-thought reasoning, the user provides an example with the steps a human would take to achieve the desired outcome or calculation (Figure 14, page 14). This technique can also be used for commonsense and symbolic reasoning tasks[40].

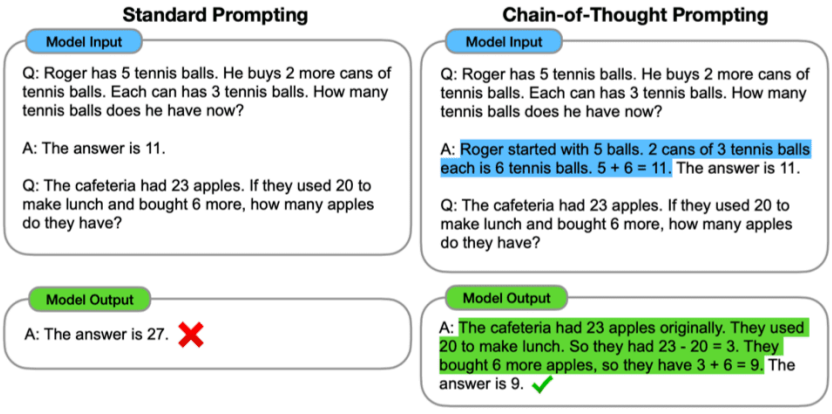

Figure 14. Chain-of-thought prompting example[40]

2.4.2. *Fine-tuning*

Fine-tuning is the process of updating pre-trained LLM weights by training on specific datasets for chosen tasks[36][33]. Supervised fine-tuning (SFT)[19], reinforcement learning with human feedback (RLHF)[42], and parameter efficient fine-tuning (PEFT)[43] are three of the most popular fine-tuning approaches for LLMs.

However, there are several other fine-tuning approaches that can be employed. They can broadly be categorised as full model fine-tuning (e.g. SFT[19] and RLHF[42]), PEFT (e.g. low-rank adoption (LoRA)[44]), and model compression and deployment optimisation (e.g. quantisation-aware fine-tuning as used for Q8BERT LLM[45]). A visualisation of this grouping and associated fine-tuning techniques in each category are shown as a mind map in Figure 15 on page 14.

Sometimes a single technique may be insufficient in delivering the desired outcomes, and instead we can combine multiple fine-tuning strategies, leveraging the strengths of each technique in order to address shortcomings, such as solving multiple constraints simultaneously or the need to optimise for performance, efficiency, and alignment. For example, combining RLHF with LoRA would yield models that are both aligned with human preferences and parameter-efficient.

Figure 15. LLM fine-tuning categories

To complement the visual representation of fine-tuning strategies in Figure 15 on page 14, Table LABEL:tbl:finetune on page LABEL:tbl:finetune provides a bit more detail for each of the strategies in the diagram.

Fine-Tuning Approach	Description	When to Use
Supervised Fine-Tuning (SFT)	Updating all pre-trained model parameters on labeled data for a specific task[36].	• Substantial amount of labelled data available. • Well-defined tasks requiring high accuracy.
Instruction Fine-Tuning	Using instruction-response pairs to enhance the model's ability to follow human instructions[46].	• Improving the model's ability to understand and execute human instructions. • Developing assistant-like applications.
Reinforcement Learning from Human Feedback (RLHF)	Using human feedback to train a reward model to guide models via reinforcement learning to align with human preferences[42].	• Aligning model outputs with human values and preferences. • Improving response quality and safety.
Multi-Task Fine-Tuning	Simultaneously fine-tuning for multiple tasks to achieve better generalisation[47].	• Models that perform well on multiple tasks. • To improve generalisation.
Continual Learning	Sequentially fine-tuning the model on new tasks while preserving previous knowledge[48].	• Model needs to adapt over time, e.g. evolving data distributions. • To prevent catastrophic forgetting.
Fine-Tuning with Frozen Layers	Freezing certain layers and updating only the top layers to reduce computation and retain general knowledge[49].	• Limited computational resources. • Limited fine-tuning data available. • To prevent overfitting.

Fine-Tuning Approach	Description	When to Use
Sparse Fine-Tuning	Updating only a subset of model parameters relevant to the new tasks[50].	• For computational efficiency. • Limited fine-tuning data available. • To prevent overfitting.
Prefix Tuning and Prompt Tuning	Adding trainable continuous prompts or prefix tokens to inputs to adapt model with minimal changes to original parameters[51][52].	• For parameter-efficient fine-tuning. • Adapting to multiple tasks with minimum alteration to core weights.
Low-Rank Adaptation (LoRA)	LoRA freezes model weights and inserts trainable low-rank matrices in the model layers which reduces the number of trainable parameters[44].	• Limited computational resources. • Rapid experimentation required.
Adapter Layers	Inserting lightweight adapter modules in model layers to adapt to new tasks, only updating adapter parameters[43][53] .	• Limited computational resources. • For multi-task learning with a shared base model . • Avoiding catastrophic forgetting because base model is unchanged.
Federated Learning for LLMs	Fine-tuning across decentralised data sources while preserving privacy[54].	• Data privacy is a concern. • Well suited for sensitive or proprietary data.
Progressive Training	Trains models in stages, starting with smaller models and gradually increasing complexity[55].	• Large datasets or models. • Improve generalisation over progressive complexity.

Fine-Tuning Approach	Description	When to Use
Quantisation-Aware Fine-Tuning	Simulating quantisation effects to ensure robustness and maintain performance on low-precision hardware (e.g., 8-bit systems)[45].	• Deploying on devices with limited computational power. • Reducing model size and increasing inference speed.
Knowledge Distillation	Training a smaller model to mimic a larger model for deployment in resource-constrained environments[56].	• A lightweight model is required. • Ideal for real-time inference. • To compress models without significant performance loss.

Table 2. Summary of fine-tuning strategies

2.5. Creating a bespoke LLM

In some instances it may be preferable to develop a bespoke LLM instead of fine-tuning one of the popular foundational models. To do this, the following steps provide a general approach:

1. *Data Selection and Preparation*

- *Data gathering:* The foundation of any LLM is the data it learns from. Therefore, identifying the appropriate and relevant data sources is an important first step in developing a bespoke LLM and requires a clear understanding of the key objectives of the LLM. The data gathering exercise typically involves obtaining extensive text data from various sources such as books, websites, and articles, but in other cases the inclusion of a highly diverse corpus of text data may be less relevant and attention is instead focussed on sourcing only a few, but high quality datasets to train the model on. Nonetheless, some additional refinements may be achieved at a later stage by employing a variety of learning approaches as discussed in Section 2.4 Adapting LLMs for Specific Use Cases.
- *Preprocessing:* The data typically needs some degree of preprocessing, such as data cleansing to remove noise and irrelevant content, normalisation to standardise text formats, and tokenisation to convert text into a format that the model can understand.

- *Annotation:* If supervised learning is involved, this stage may also include annotating the data with labels.
- *Training data:* Finally, the dataset is split into training, validation, and test datasets to enable effective learning and unbiased evaluation. The training data allocation is usually set around 15%.

2. *Model Design and Configuration*

   Choosing the right model architecture is critical to achieving the desired performance. For LLMs, Transformer-based architectures are commonly used due to their ability to capture long-range dependencies in text. This step involves configuring the model's parameters, such as the number of layers, hidden units, and attention heads, to balance performance with computational feasibility. Hyperparameter tuning is conducted to find optimal settings for learning rate, batch size, and regularisation techniques, which can significantly impact the efficiency and effectiveness of the training process.

3. *Training the model*

   Once the data has been sourced and cleaned, and the model architecture chosen, the training environment needs to be set up. This includes selecting appropriate loss functions (like cross-entropy loss) and optimisers (such as Adamor Adafactor [57]). The model learns by minimising the loss function over the training data, adjusting its internal parameters to improve predictions. Throughout training, it is important to monitor metrics like loss and accuracy, and to validate the model on the validation set to prevent overfitting, a problem inherent in machine learning techniques.

4. *Fine-tuning and Deployment*

   Once a model is trained we need to evaluate it against expected behaviour and through approaches such as fine-tuning and prompt engineering to ensure that the model performs as desired. Using the prior technique will adjust model weights, whereas the latter leaves the original weights in tact. Further actions such as developing APIs to access the trained LLM may then be undertaken and deployed in production. Post deployment it is crucial to be cognisant of ethical implications and legal considerations, including assessment of unintended biases.

Ultimately LLM development is an iterative process and by leveraging information such as user feedback, metrics of loss and accuracy, changes to task requirements and/or current data, and

compliance with the validation set to prevent overfitting, we can ensure that the LLM remains relevant throughout its life.

2.6. Interacting with LLMs

There is a myriad of ways in which we can interact with LLMs, depending on the desired end goal(s). Some interactions such as fine-tuning (See Section 2.4.2, page 13) adjust the base model while others focus on the most effective way to perform tasks and extract information without adjusting the underlying model. Figure 15, page 14 visually summarises the various approaches.

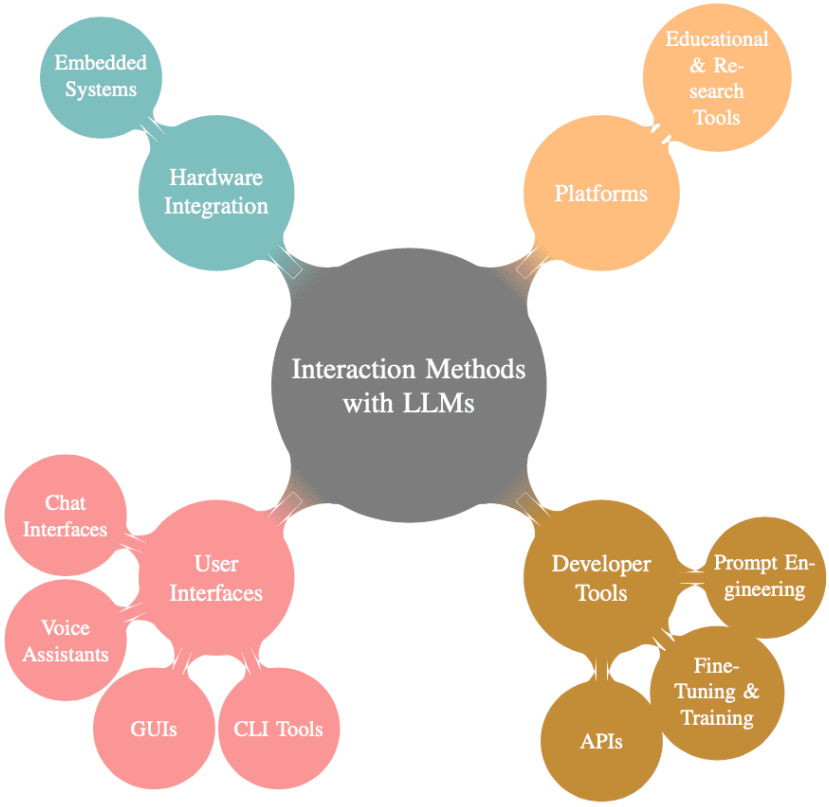

Figure 16. Ways of interacting with an LLM

Table 3, page 19, gives a brief overview of these methods, but arguably the most common and well known way of interacting with LLMs is through a chat bot such as ChatGPT (by OpenAI), BARD (by Google), Claude (by Anthropic) and Bing Chat (by Microsoft, powered by OpenAI) [7][58][59].

doi.org/10.32388/NHJYVS

Method	Explanation
Chat Interfaces	User-friendly platforms for real-time conversational interaction with LLMs.
Voice Assistants	Use of speech to interact with LLMs in voice-enabled applications.
Graphical User Interfaces (GUIs)	GUI-based applications enabling interaction with LLMs without coding.
Command-Line Interfaces (CLIs)	Interaction with LLMs via command-line tools for scripting and automation tasks.
APIs & Wrappers	Programmatic access to LLMs and associated libraries for ease of integration into applications and services.
Fine-Tuning and Training	Adjusting model parameters to perform specialised tasks using machine learning tools.
Prompt Engineering	The art of crafting specific prompts to elicit desired outputs by LLMs.
Educational and Research Tools	Using platforms such as Jupyter and Colab for experimenting and learning with LLMs.
Embedded Systems	Integration of LLMs into hardware devices for natural language understanding.

Table 3. Methods to Interact with LLMs

In Section 2.6.1, page 18, we discuss prompt engineering in more detail, a simple and effective way for most users to harness the knowledge, and explore the functionality, of an LLM. However, a flexible, powerful and effective way of interacting with LLMs is through application program interfaces (APIs), but that requires a higher level of technical expertise. Several of the well known pre-trained LLMs provide APIs, some are open source and others not. The more popular APIs are: Hugging Face's Transformers Library and Inference API, Google Cloud's Natural Language API, IBM Watson Language Translator API, APIs to access BERT can be obtained via Google Research BERT repository or through Hugging Face's BERT model webpage, and APIs for OpenAI GPT-4o and GPT-4o mini.

doi.org/10.32388/NHJYVS

2.6.1. Prompt Engineering

Prompt engineering is a technique used to maximise the effectiveness of an existing LLM without altering its internal structure. The process comprises three parts: the *prompt* itself is the model input, model *inference* is the generation of text in response to the prompt, and lastly *completion* is the resulting output text. The *context window* is the all the text and memory that is available.

By carefully crafted prompts, users can harness these models more effectively, leading to better outcomes in tasks ranging from simple queries to complex problem solving, but it has limitations. One effective strategy to improve model outcomes is by including examples inside the context window (Figure 12, page 12). This process is called *in-context learning* and the variations of in-context learning are: [36]:

- *zero-shot* inference - no examples provided
- *one-shot* inference - one example provided
- *few-shot* inference - more than one example provided

We can also view prompt engineering as a complementary technique to fine-tuning by using it to generate training data or as an interim solution to improve the model's performance. A general guide for progressing on to fine-tuning is when the number of examples (few shot learning) is growing to more than 5 or 6, with diminishing improvements in LLM output. Nonetheless, the research study by Brown, T.B., et al. [36] used a few dozen examples in their few-shot settings (Figure 13, page 12).

2.6.2. Summary of LLM Overview

This overview of LLMs is visually captured in Figure 17 on page 20 depicting the different phases and characteristics of LLMs.

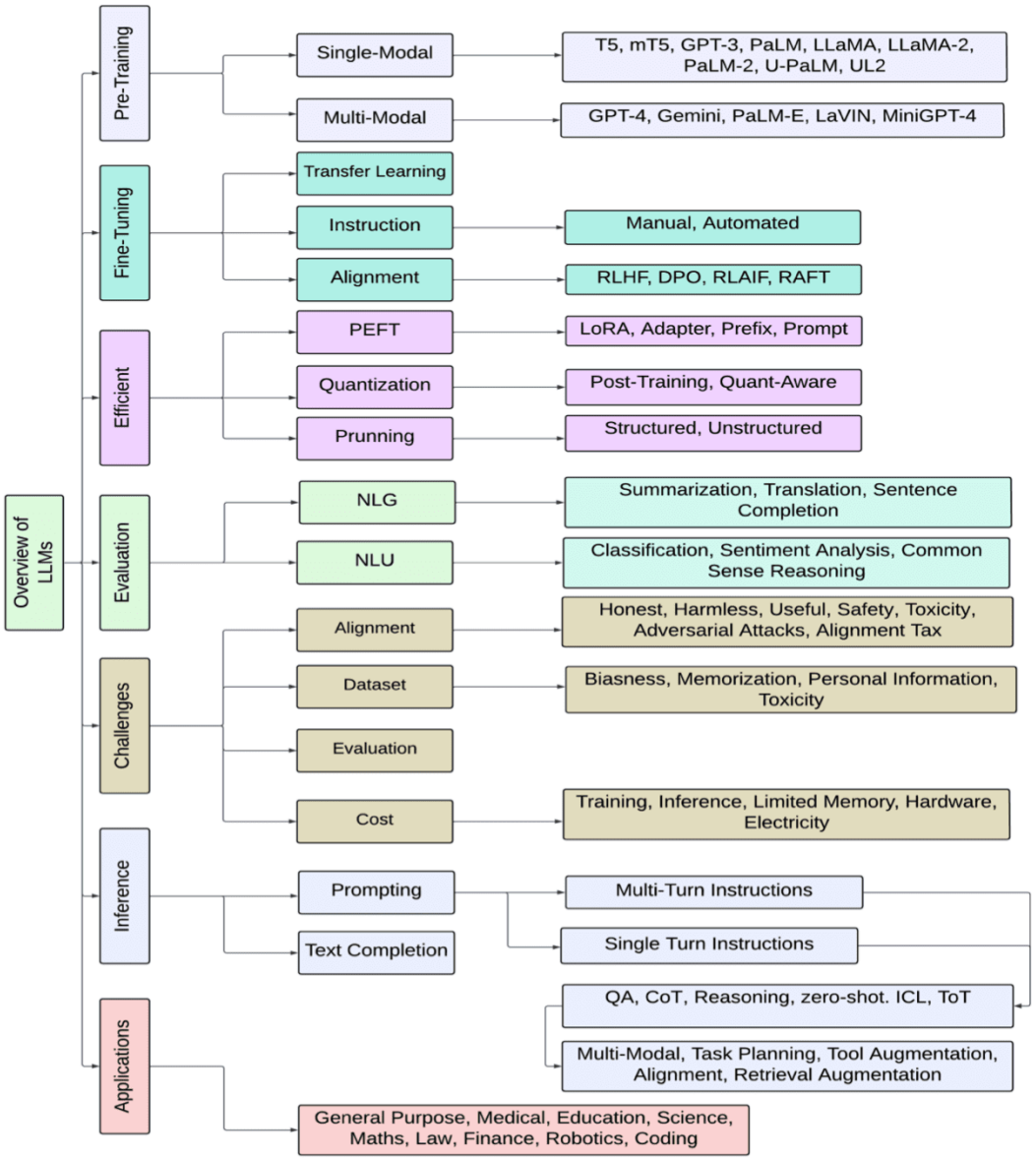

Figure 17. Overview of the various characteristics, activities and strategies of LLMs : 1. Pre-Training 2. Fine-Tuning 3. Efficient 4. Inference 5. Evaluation 6. Applications 7. Challenges (image by Naveed, H., et al.[20])

3. LLMs Orchestrated with Other Technologies

Orchestration of LLMs with traditional information retrieval systems has been explored since the early stages of this technology. Google researchers developed a platform in 2017 to speed up the creation and maintenance of production platforms when combining components of their TensorFlow

machine learning system[60]. Some of those same techniques are present in more modern systems today. In 2018 medical informatics researchers combined image caption-generation engines with structured data stores to yield better captions[61]. By 2020, Google and collaborators were retrieving textual data from a textual knowledge corpus based on Wikipedia documents to augment pre-training of LLMs.

Starting in 2022, LangChain[2] was released as an open source software project to assist software developers with the integration of LLMs into software applications. A venture-funded company was later built around the project. Many other orchestration platforms have since appeared, including close competitor n8n[3], Haystack[4] and LlamaIndex[5]. Many of these systems are released under open source licenses.

The subfield of Knowledge Representation (KR) provides the intellectual foundation and practical tooling to represent data in ways that serve as input to other AI or data management system (DMS) systems. KR systems include ontologies, metadata, and other forms of structured information that enable meaningful representation of domain-specific knowledge. The orchestration of structured KR and unstructured LLM systems can result in an LLM fine-tuned for a specific domain by runtime reference to a specific ontology[62][63]. For example, application of such an orchestrated system in an engineering domain can output engineering intention artefacts[63]. Another example is orchestration in a medical domain where it is highly desirable to have explainable AI (XAI) so that the LLM can explain the reasoning leading to the conclusions and output making it verifiable by humans. This is especially important given the sensitive and critical nature of medical advice and the potential harmful implications of mis-diagnoses[62].

Alternatively, RAG may be orchestrated with LLMs to create custom chatbots or agents, document summarisation systems using specialist vocabularies and provide data integration with existing systems[64].

Figure 18 illustrates the emerging architecture of systems orchestrated with LLMs in 2024. That figure is courtesy of Andreessen Horowitz Enterprise[6].

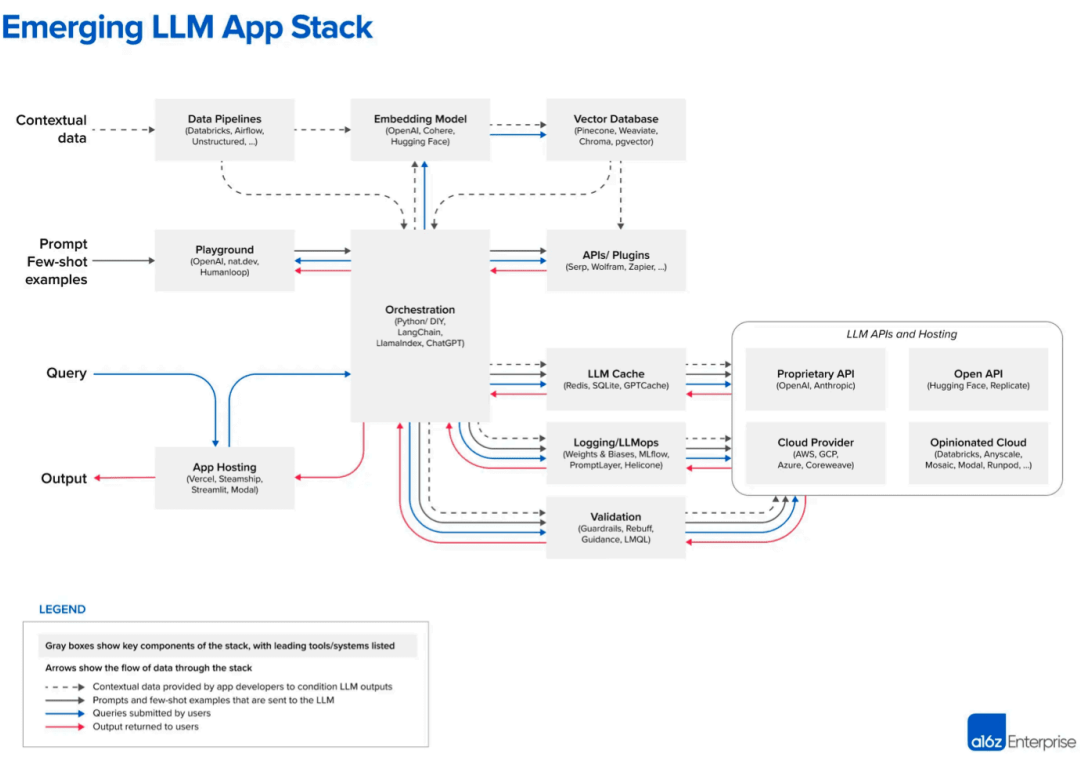

Figure 18. Emerging LLM application stack (provided by Andreessen Horowitz Enterprise)

4. Risks and Mitigations

4.1. Catastrophic Forgetting

Catastrophic forgetting, or catastrophic interference, is when neural networks, including LLMs, become less performant on tasks that they previously excelled at[65][66][67]. In other words, they essentially "forget" previously learned information[65]. This behaviour is typically observed when LLMs are fine-tuned sequentially on different tasks or datasets, a process known as continual learning[48]. The underlying cause is that the fine-tuning exercise updates the model's weights to optimise performance on the new task. Several strategies have been proposed to prevent catastrophic forgetting, such as:

- Regularisation-based method: Adding regularisation terms to penalise significant changes to important weights. For example elastic weight consolidation (EWC) adds a regularisation term to the loss function for changes to important weights[48].

- Replay-based method: Retraining the model on a mix of old and new data, or using synthetic data generated from the model's memory of previous tasks[68][69].
- Architectural methods: Using separate subnetworks for different tasks, or dynamically expanding the network. For example, progressive neural networks that create new subnetworks for each task while keeping the original fixed[70], and adapter modules that can be inserted into the network and fine-tuned separately for each task[71].

4.2. Model Collapse

LLMs are trained on many public data sources as described in Section 2 An Overview of LLMs. Since many users are using LLMs to generate content that is being put onto those same public fora, future versions of those LLMs are very likely to ingest content generated by earlier versions of themselves. It is not difficult to envision a future in which LLMs become trained on an ever-increasing amount of machine-generated content and a decreasing amount of human-generated content. The ramifications are intriguing; without a change in the way the models are trained their weights will be increasingly influenced by machine-generated content. Human-generated content could even become a minority input for some models.

The unintended or unrecognised prevalence of machine-generated content in training data coupled with the failure of LLMs to differentiate human- and machine-generated content is known as model collapse[72]. An LLM in model collapse would not treat human-generated content in a preferred manner. Instead, a positive feedback loop would be set up whereby new LLMs will learn to write like old LLMs.

Possible mitigations for model collapse include the use of data provenance techniques to label human- and/or machine-generated content[73]. Such approaches are limited to mitigating, not solving, the problem of model collapse because many systems and users may simply fail to provide or choose to ignore data provenance hints.

Other mitigations may be possible via governmental AI strategies and subsequent regulation[74]. A common analysis technique for such frameworks is the PESTEL analysis technique. PESTEL is an acronym standing for political, economic, social, technological, environmental, and legal factors in an environment external to an organisation[75]. Tjondronegoro notes that a PESTEL analysis of AI adoption barriers and themes suggests that "Data availability, quality, and structure" fall under the

technology rubric[74]. Governments may choose to selectively regulate some data availability to reduce negative consequences of model collapse.

None of the currently-identified mitigations to model collapse appear to be sufficient to prevent the phenomenon from occurring. More research into this area is urgently needed.

4.3. Jailbreak Attacks

A jailbreak is an adversarial attack in which users craft specific prompts designed to bypass the model's ethical safeguards. These jailbreak prompts trick the model into generating harmful or unethical responses, circumventing its alignment with moral guidelines[76][39]. Users may craft jailbreak prompts for various reasons, including:

- *Bypassing restrictions*: Some users may want to elicit responses that are blocked by default, such as unethical or illegal content that the LLM would typically refuse to generate.
- *Malicious intent*: Jailbreaks can be used to manipulate the LLM into generating content for harmful purposes, such as misinformation, hate speech, or instructions for illegal activities like fraud or cybercrime.
  (An example of tricking a chatbot to generate a blackmail letter is shown in Figure 19, page 24).
- *Security and Research*: Researchers or security personnel might craft jailbreaks to gain a thorough understanding of the vulnerabilities in the AI system, which they can then guard against.
- *Entertainment*: Others might use jailbreaks for humour or entertainment, pushing the LLM to say things it wouldn't normally say.

Xie et al.[76] propose several strategies to defend against jailbreaks. AI models can employ "self-regulation techniques" like system-mode self-reminders, which wrap user queries in prompts that remind the model to behave ethically. This method significantly reduces the success rate of jailbreak attacks by reinforcing the model's ethical guidelines. Other safeguards include RLHF to continually align the model with moral values, and content filtering systems that automatically detect and block adversarial prompts[76].

Additionally, watermarking and classifiers can help identify when a model's behaviour deviates from its ethical programming, enabling automatic interventions. Prompt optimisation and testing to study and enhance the model's resistance to jailbreaks, can also better inform strategies to combat jailbreaking[76].

Figure 19. Example of jailbreaking and the use of a system-mode self-reminder to defend this attack (image by[76])

4.4. Hallucinations and their Impacts

As anyone who has played, however briefly, with LLMs and multimodal AI models knows, they can sometimes produce output that goes very rapidly from amazing to badly wrong. These forays into fantasy are generally known as hallucinations. We argue that the term "hallucinations" is misleading and has, in fact, been the cause of much misunderstanding about the limitations of LLMs.

LLMs do not hallucinate, they produce *bullshit*, in that word's technical sense.

Naturally, we recognise that language changes over time. The Cambridge Dictionary has already added a second definition to "hallucinate" specifically related to AI systems[7]. Nevertheless, the semantics seems worthy to us of pursuit because it increases the explanatory power of our ability to conceptualise LLMs.

In the seminal paper, *On bullshit*, by philosopher Harry Frankfurt[7,7] "bullshit" is described as a form of communication where the speaker is indifferent to the truth. This indifference to the truth means that bullshit doesn't necessarily involve untrue statements; rather, it involves a total disregard for the truth. He argues that this makes bullshit a greater threat to truth than outright lying because it undermines the very value of truth in discourse. Inspired by this concept, Hannigan et al.[3] and Hicks et al.[4] note that because LLMs do not comprehend the meaning of the responses they generate, "the activity they are engaged in is bullshitting, in the Frankfurtian sense" (quoted from Hicks). Hannigan et al. have termed the specific creation of Frankfurtian bullshit by chatbots "botshit".

To guard against botshit from LLMs there are a few simple strategies that a user can follow: for critical tasks all information provided by the chatbot should be verified through trusted sources, cross-checked and not relied on blindly. Much can be done by training users to craft clear prompts and have a sound understanding of the chatbot's limitations, such as outdated data or areas prone to hallucinations. Within an organisation it is good practice to establish guidelines on when and how to use chatbot content. In general users are advised to simply maintain a critical mindset, using chatbot outputs as a starting point, rather than the final answer, especially for non-critical tasks where the LLM output may be treated as creative input instead of the truth.

As an example, we asked ChatGPT 4, 4o and o1-preview to identify the canonical academic references for LLM orchestration. Most of the resulting academic paper suggested by the LLMs did not exist, the links to the papers did not resolve, and the conferences and journals cited do not list the papers. They were bullshit, appearing plausible but untrue.

But are they really hallucinating? We would argue that they are not, at least not in the first sense given by the Cambridge Dictionary ("to see or hear something that does not exist"). The decoder portions of language models have been designed to produce content based on their input. In important ways, that is "all" they are in spite of their complexity. We should property view them as language generators.

LLMs thus share one critical thing in common with people (although on a different scale and at radically different levels of complexity): They learn what is normal based primarily on their inputs mediated only by their architecture.

The problem with calling any LLM output a "hallucination" is that it is a post-facto subjective judgement by a human who is judging the truth or falsity of the output. That is, is a value judgement.

One cannot separate one language generation output from another in a meaningful way because there is no algorithm for truth.

The best that LLM designers or trainers can do is to adjust the systems to produce "better" content as judged by humans. We doubt that this approach will ever lead to a hallucination-free design.

We along with some of our colleagues had an intuition that multiple LLMs would be very unlikely to hallucinate in similar ways. That turns out not to be so. We were able to generate quite similar hallucinations from ChatGPT 4, ChatGPT 4o, Llama 3, Claude and Gemini using the same prompts. This was probably due to the similarities in both their architectures and their training data.

Galileo produces a "Hallucination Index" to evaluate the extent to which well known LLMs hallucinate. They published an evaluation of the hallucination without additional RAG in 2023[8], and recently published a RAG version[9] testing LLMs with varying lengths of text. Claude 3.5 Sonnet was the overall winner, and most models performed best when retrieving information from medium-length documents.

4.5. Areas of Less-Than-Human Performance

One of the now-classic ways to trick an LLM into giving a bad answer, or to show how the technology fails, is the prompt, "How many times does the letter r occur in the word 'strawberry'?"

Most LLMs will answer "2", which is incorrect. The correct answer should be "3". The LLMs get this wrong because they never see the word "strawberry" in their input. Instead, they only see a number representing a token for that word. That is, they cannot reason over the word because they only receive a number.

There are ways to work around this problem. For example,

1. Can you list all letters in the word "strawberry" in the order that they appear?
2. How many times does the letter r appear in the list that you generated?

ChatGPT 4 or 4o will correctly answer "3".

New approaches to reasoning in LLMs are addressing these limitations while at the same time introducing new performance penalties. ChatGPT o1-preview (intentionally code-named "Strawberry") will correctly answer "3" to the initial question because it does parse the word and then

double check itself. However, as of this writing the process takes around 22 seconds. No doubt additional research will improve that performance.

5. Conclusions

Based on the LLM literature we reviewed, we endeavoured to describe the technology, and the potential of these language models when used "out of the box" (e.g. foundational models) or adapted for specific use cases or tasks, or as part of an orchestrated system. We highlighted potential positive and negative impacts of LLMs and strategies used to mitigate the latter. The paper is aimed at providing students, practitioners, researchers, and decision makers an overview and insight into the various aspects this technology and its potential with some caveats.

A prudent strategy to minimise unexpected consequences of misbehaving AI tools including LLMs is continual evaluation of the accuracy and correctness of the output[39]. There are several tools that assess the relative performance of LLMs which can aid in choosing an LLM that is well suited for specific tasks and scenarios. The rate of development of AI, and LLMs specifically, is rapid and hence it is important to check regularly whether the current tool is still fit for purpose[39], and this rate of progress and innovation of LLMs continues unabated. Since we started our background research into LLMs and their applications, we have seen the emergence of an exciting new suite of models and architectures in software and hardware.

Most notably the recent announcement of the arrival of Liquid Foundation Models (LFMs)[10] in September 2024 by Liquid AI Inc, a spin-off startup of MIT. In contrast to traditional transformer foundational models, LFMs utilise a different architecture, known as liquid neural networks[78]. These neural networks are typically smaller, highly efficient, and adept at adjusting dynamically to changes in input data. These models are also generally much smaller than the traditional transformer models with a simpler structure which should make them easier to understand compared to conventional neural networks.

In the sphere of hardware advances we have seen Groq[11] design the *language processing unit (LPU)*, which is optimised for high-speed and low-latency machine learning tasks, especially inference. This hardware design emphasises efficient parallel processing and is tailored for workloads in data centers requiring rapid computation.

We envision that the evolving landscape will greatly benefit end-users by making this powerful technology more accessible and available on every day devices with improved accuracy and performance.

Footnotes

[1] From DeepLearning.AI 'Generative AI and LLMs' course.

[2] https://www.langchain.com/

[3] https://n8n.io/

[4] https://haystack.deepset.ai/

[5] https://www.llamaindex.ai/

[6] https://a16z.com/enterprise/

[7] https://dictionary.cambridge.org/us/dictionary/english/hallucinate

[8] https://www.rungalileo.io/hallucinationindex-2023

[9] https://www.rungalileo.io/hallucinationindex

[10] https://www.liquid.ai/liquid-foundation-models

[11] https://groq.com/resources/

Acknowledgements

We thank Consensys Software Inc for funding this research.

References

1. ^a, ^b*Araz Z, Syed Imran A, Nazrul I (2023). Worker and workplace Artificial Intelligence (AI) coexistence: Emerging themes and research agenda. Technovation.* ***124****: 102747. doi:10.1016/j.technovation.2023.102747. Link.*
2. ^*Eloundou T, Manning S, Mishkin P, Rock D (2024). "GPTs are GPTs: Labor market impact potential of LLMs". Science.* ***384*** *(6702): 1306–1308. doi:10.1126/science.adj0998.*
3. ^a, ^b*Hannigan TR, McCarthy IP, Spicer A (2024). "Beware of botshit: How to manage the epistemic risks of generative chatbots". Business Horizons.* ***67****(5): 471–486. doi:10.1016/j.bushor.2024.03.001. Link to a*

rticle. SPECIAL ISSUE: WRITTEN BY CHATGPT.

4. ^a, b^*Hicks MT, Humphries J, Slater J (2024). "ChatGPT is bullshit". Ethics and Information Technology. **26**(2): 38. doi:10.1007/s10676-024-09775-5. https://link.springer.com/10.1007/s10676-024-09775-5.*
5. ^^*Strachan JWA, Albergo D, Borghini G, Pansardi O, Scaliti E, Gupta S, Saxena K, Rufo A, Panzeri S, Manzi G, Graziano MSA, Becchio C (2024). "Testing theory of mind in large language models and humans". Nat Hum Behav. **1**: 1–11. doi:10.1038/s41562-024-01882-z.*
6. ^^*Kosinski M (2024). "Evaluating Large Language Models in Theory of Mind Tasks". arXiv. doi:10.48550/arXiv.2302.02083. Available from: http://arxiv.org/abs/2302.02083.*
7. ^a, b^*Biever C (2023). "ChatGPT broke the Turing test — the race is on for new ways to assess AI". Nature. **619** (7971): 686–689. doi:10.1038/d41586-023-02361-7. https://www.nature.com/articles/d41586-023-02361-7.*
8. ^^*Mei Q, Xie Y, Yuan W, Jackson MO (2024). "A Turing test of whether AI chatbots are behaviorally similar to humans". Proceedings of the National Academy of Sciences. **121**(9). doi:10.1073/pnas.2313925121.*
9. ^^*Kuhn RL (2024). "A landscape of consciousness: Toward a taxonomy of explanations and implications". Progress in Biophysics and Molecular Biology. **190**: 28–169. doi:10.1016/j.pbiomolbio.2023.12.003. Link.*
10. ^a, b^*Gallegos IO, Rossi RA, Barrow J, Tanjim MM, Kim S, Dernoncourt F, Yu T, Zhang R, Ahmed NK (2024). "Bias and fairness in large language models: A survey". Computational Linguistics. pp. 1–79.*
11. ^a, b, c^*Nangia N, Vania C, Bhalerao R, Bowman SR (2020). "CrowS-Pairs: A Challenge Dataset for Measuring Social Biases in Masked Language Models". Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics; Online. p. 1953–1967. doi:10.18653/v1/2020.emnlp-main.154. https://aclanthology.org/2020.emnlp-main.154.*
12. ^a, b, c, d, e^*Touvron H, Lavril T, Izacard G, Martinet X, Lachaux M-A, Lacroix T, Rozière B, Goyal N, Hambro E, Azhar F, Rodriguez A, Joulin A, Grave E, Lample G (2023). "LLaMA: Open and Efficient Foundation Language Models". arXiv. Available from: https://arxiv.org/abs/2302.13971.*
13. ^a, b^*Wu S, Irsoy O, Lu S, Dabravolski V, Dredze M, Gehrmann S, Kambadur P, Rosenberg D, Mann G (2023). "BloombergGPT: A Large Language Model for Finance". arXiv.org. Available from: https://arxiv.org/abs/2303.17564v3.*
14. ^^*Jelinek F (1976). "Continuous speech recognition by statistical methods". Proceedings of the IEEE. **64**(4): 532–556. doi:10.1109/PROC.1976.10159. https://ieeexplore.ieee.org/document/1454428.*

15. ^Hochreiter S (1998). "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions". International journal of uncertainty, fuzziness, and knowledge-based systems. 6(2): 107–116. doi:10.1142/S0218488598000094.

16. ^Roodschild M, Gotay Sardiñas J, Will A (2020). "A new approach for the vanishing gradient problem on sigmoid activation". Progress in Artificial Intelligence. 9(4): 351–360. doi:10.1007/s13748-020-00218-y.

17. a, b, cVaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2023). "Attention Is All You Need". arXiv. doi:10.48550/arXiv.1706.03762. Available from: http://arxiv.org/abs/1706.03762.

18. ^Kitaev N, Kaiser Ł, Levskaya A (2020). "Reformer: The Efficient Transformer". arXiv. doi:10.48550/arXiv.2001.04451. Available from: http://arxiv.org/abs/2001.04451.

19. a, b, c, d, eDevlin J, Chang MW, Lee K, Toutanova K (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". arXiv:1810.04805 [cs.CL].

20. a, b, c, dNaveed H, Khan AU, Qiu S, Saqib M, Anwar S, Usman M, Akhtar N, Barnes N, Mian A (2024). "A Comprehensive Overview of Large Language Models". arXiv. Available from: https://arxiv.org/abs/2307.06435.

21. ^Bloomberg S (2024). Dissecting the Intersection of AI and Crypto. Technical Report. Messari. Available from: https://messari.io/report/dissecting-the-intersection-of-ai-and-crypto.

22. a, b, c, dBarth A, Fregly C, Eigenbrode S, Chambers M. Generative AI with LLMs - DeepLearning.AI. Available from: https://www.deeplearning.ai/courses/generative-ai-with-llms/. Accessed 2024 Sep 5.

23. ^Woo DJ, Guo K, Salas-Pilco SZ (2024). "Writing creative stories with AI: learning designs for secondary school students". Innovation in language learning and teaching. pp. 1–13.

24. ^Micchi G, Bigo L, Giraud M, Groult R, Lev\'e F (2021). "I Keep Counting: An Experiment in Human/AI Co-creative Songwriting". Transactions of the International Society for Music Information Retrieval. 4: 263+. doi:10.5334/tismir.93.

25. ^Smith R, Gonzalez MC, McKeon E. The AI Revolution in Customer Service and Support: A Practical Guide to Impactful Deployment of AI to Best Serve Your Customers. [First edi edition]. Hoboken, New Jersey: Pearson; 2024. ISBN 0138286523.

26. a, bMohan G, Satish G, Patil H, Vekariya V, Natrayan L, Barve A (2023). "AI-Powered Chatbot for Bridging Language Barriers with Translation". In: 2023 3rd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA). IEEE, pp. 1559–1565.

27. ^Chu A, Mathews LR, Yu K-H (2023). "Chapter 1 - Artificial intelligence in health care: past and presen t". In: Su T-H, Kao J-H, editors. Artificial Intelligence, Machine Learning, and Deep Learning in Precisio n Medicine in Liver Diseases. Academic Press. pp. 3–17. doi:10.1016/B978-0-323-99136-0.00001-5. Li nk.

28. ^Ashley KD. Artificial intelligence and legal analytics: New tools for law practice in the digital age. 6th p rint. ed. CAMBRIDGE: Cambridge Univ Press; 2017.

29. a, b, c, d, eZhao WX, Zhou K, Li J, Tang T, Wang X, Hou Y, Min Y, Zhang B, Zhang J, Dong Z, Du Y, Yang C, Chen Y, Chen Z, Jiang J, Ren R, Li Y, Tang X, Liu Z, Liu P, Nie JY, Wen JR (2023). "A Survey of Large Lang uage Models". arXiv. doi:10.48550/arXiv.2303.18223. Available from: http://arxiv.org/abs/2303.18223.

30. a, b, c, dWang T, Roberts A, Hesslow D, Scao TL, Chung HW, Beltagy I, Launay J, Raffel C (2022). "What L anguage Model Architecture and Pretraining Objective Work Best for Zero-Shot Generalization?" arXiv. doi:10.48550/arXiv.2204.05832. Available from: http://arxiv.org/abs/2204.05832. arXiv:2204.05832 [c s, stat].

31. a, b, c, d, e, fMicheletti N, Belkadi S, Han L, Nenadic G (2024). "Exploration of Masked and Causal Langu age Modelling for Text Generation". arXiv. Available from: https://arxiv.org/abs/2405.12630.

32. a, b, cClark K, Luong MT, Le QV, Manning CD (2020). "ELECTRA: Pre-training Text Encoders as Discrimi nators Rather Than Generators". arXiv. Available from: https://arxiv.org/abs/2003.10555.

33. a, b, cRaffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, Zhou Y, Li W, Liu PJ (2023). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". arXiv. doi:10.48550/arXiv.19 10.10683. Available from: http://arxiv.org/abs/1910.10683.

34. a, bLewis M, Liu Y, Goyal N, Ghazvininejad M, Mohamed A, Levy O, Stoyanov V, Zettlemoyer L (2019). "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension". arXiv. arXiv:1910.13461.

35. ^Xue L, Constant N, Roberts A, Kale M, Al-Rfou R, Siddhant A, Barua A, Raffel C (2021). "mT5: A massiv ely multilingual pre-trained text-to-text transformer". arXiv. Available from: https://arxiv.org/abs/201 0.11934.

36. a, b, c, d, e, f, g, h, i, jBrown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A, Shya m P, Sastry G, Askell A, Agarwal S, Herbert-Voss A, Krueger G, Henighan T, Child R, Ramesh A, Ziegler D M, Wu J, Winter C, Hesse C, Chen M, Sigler E, Litwin M, Gray S, Chess B, Clark J, Berner C, McCandlish S, Radford A, Sutskever I, Amodei D (2020). "Language Models are Few-Shot Learners". arXiv. Available f rom: https://arxiv.org/abs/2005.14165.

37. ^Hoffmann J, Borgeaud S, Mensch A, Buchatskaya E, Cai T, Rutherford E, de Las Casas D, Hendricks LA, Welbl J, Clark A, Hennigan T, Noland E, Millican K, van den Driessche G, Damoc B, Guy A, Osindero S, Si monyan K, Elsen E, Rae JW, Vinyals O, Sifre L (2022). "Training Compute-Optimal Large Language Mo dels". arXiv (Cornell University). doi:10.48550/arxiv.2203.15556.

38. a, bBalestriero R, Ibrahim M, Sobal V, Morcos A, Shekhar S, Goldstein T, Bordes F, Bardes A, Mialon G, Ti an Y, Schwarzschild A, Wilson AG, Geiping J, Garrido Q, Fernandez P, Bar A, Pirsiavash H, LeCun Y, Gold blum M (2023). A Cookbook of Self-Supervised Learning. Available from: https://arxiv.org/abs/2304.12 210.

39. a, b, c, dParthasarathy VB, Zafar A, Khan A, Shahid A (2024). "The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs: An Exhaustive Review of Technologies, Research, Best Practices, Applied Research Challenges and Opportunities". arXiv. Available from: https://arxiv.org/abs/2408.13296.

40. a, b, c, dWei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, Chi E, Le Q, Zhou D (2023). "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models". arXiv. Available from: https://arxiv.o rg/abs/2201.11903.

41. ^Ng A. DeepLearning.AI [Internet]. Available from: https://www.deeplearning.ai/.

42. a, b, cOuyang L, Wu J, Jiang X, Almeida D, Wainwright CL, Mishkin P, Zhang C, Agarwal S, Slama K, Ray A, Schulman J, Hilton J, Kelton F, Miller L, Simens M, Askell A, Welinder P, Christiano P, Leike J, Lowe R (2022). "Training language models to follow instructions with human feedback". arXiv. Available from: https://arxiv.org/abs/2203.02155.

43. a, bHoulsby N, Giurgiu A, Jastrzebski S, Morrone B, de Laroussilhe Q, Gesmundo A, Attariyan M, Gelly S (2019). "Parameter-Efficient Transfer Learning for NLP". arXiv.org. Available from: http://arxiv.org/ab s/1902.00751.

44. a, bHu EJ, Shen Y, Wallis P, Allen-Zhu Z, Li Y, Wang S, Wang L, Chen W (2021). "LoRA: Low-Rank Adapt ation of Large Language Models". arXiv. doi:10.48550/arXiv.2106.09685. Available from: http://arxiv.or g/abs/2106.09685.

45. a, bZafrir O, Boudoukh G, Izsak P, Wasserblat M (2019). "Q8BERT: Quantized 8Bit BERT". In: 2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-N IPS). IEEE, pp. 36-39. doi:10.1109/emc2-nips53020.2019.00016.

46. ^Wei J, Bosma M, Zhao VY, Guu K, Yu AW, Lester B, Du N, Dai AM, Le QV (2022). "Finetuned Language Models Are Zero-Shot Learners". arXiv. Available from: https://arxiv.org/abs/2109.01652.

47. ^Lin N, Lara H, Guo M, Rastogi A (2024). MoDE: Effective Multi-task Parameter Efficient Fine-Tuning with a Mixture of Dyadic Experts. Available from: https://arxiv.org/abs/2408.01505.

48. a, b, cKirkpatrick J, Pascanu R, Rabinowitz N, Veness J, Desjardins G, Rusu AA, Milan K, Quan J, Ramalho T, Grabska-Barwinska A, Hassabis D, Clopath C, Kumaran D, Hadsell R (2017). "Overcoming catastrophic forgetting in neural networks". Proceedings of the National Academy of Sciences. **114** (13): 3521–3526. doi:10.1073/pnas.1611835114. Link.

49. ^Panoutsopoulos H, Espejo-Garcia B, Raaijmakers S, Wang X, Fountas S, Brewster C (2024). "Investigating the effect of different fine-tuning configuration scenarios on agricultural term extraction using BERT". Computers and Electronics in Agriculture. **225**: 109268. doi:10.1016/j.compag.2024.109268. Link.

50. ^Mu B, Giannoula C, Wang S, Pekhimenko G (2024). "Sylva: Sparse Embedded Adapters via Hierarchical Approximate Second-Order Information". In: Proceedings of the 38th ACM International Conference on Supercomputing, ICS '24. New York, NY, USA: Association for Computing Machinery. pp. 485–497. doi:10.1145/3650200.3656619.

51. ^Li XL, Liang P (2021). "Prefix-Tuning: Optimizing Continuous Prompts for Generation". Available from: https://arxiv.org/abs/2101.00190.

52. ^Shrestha P, Kandel J, Tayara H, Chong KT (2024). "Post-translational modification prediction via prompt-based fine-tuning of a GPT-2 model". Nature Communications. **15** (1): 6699. doi:10.1038/s41467-024-51071-9. Link.

53. ^Lialin V, Deshpande V, Rumshisky A (2023). "Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning". arXiv. doi:10.48550/arXiv.2303.15647. Available from: http://arxiv.org/abs/2303.15647. ArXiv:2303.15647 [cs].

54. ^McMahan HB, Moore E, Ramage D, Hampson S, Agüera y Arcas B (2023). "Communication-Efficient Learning of Deep Networks from Decentralized Data". arXiv. Available from: https://arxiv.org/abs/1602.05629.

55. ^Lu J, Zhong W, Wang Y, Guo Z, Zhu Q, Huang W, Wang Y, Mi F, Wang B, Wang Y, Shang L, Jiang X, Liu Q (2024). "YODA: Teacher-Student Progressive Learning for Language Models". arXiv. Available from: https://arxiv.org/abs/2401.15670.

56. ^Hinton G, Vinyals O, Dean J (2015). Distilling the Knowledge in a Neural Network. Available from: https://arxiv.org/abs/1503.02531.

57. ^Keras Team. Keras documentation: About Keras 3. Available from: https://keras.io/about/.

58. ^Z\uoofa\uoof1iga Salazar G, Z\uoofa\uoof1iga D, Vindel CL, Yoong AM, Hincapie S, Z\uoofa\uoof1iga AB, Z\uoofa\uoof1iga P, Salazar E, Z\uoofa\uoof1iga B (2023). "Efficacy of AI Chats to Determine an Emergency: A Comparison Between OpenAI's ChatGPT, Google Bard, and Microsoft Bing AI Chat". Cureus. **15** (9). doi:10.7759/cureus.45473.

59. ^Cheng HW (2023). "Challenges and Limitations of ChatGPT and Artificial Intelligence for Scientific Research: A Perspective from Organic Materials". AI. **4** (2): 401. doi:10.3390/ai4020021.

60. ^Baylor D, Breck E, Cheng HT, Fiedel N, Foo CY, Haque Z, Haykal S, Ispir M, Jain V, Koc L, et al. (2017). "Tfx: A tensorflow-based production-scale machine learning platform". In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, pp. 1387–1395.

61. ^Li Y, Liang X, Hu Z, Xing EP (2018). "Hybrid retrieval-generation reinforced agent for medical image report generation". Advances in neural information processing systems. **31**.

62. a, bHolzinger A, Biemann C, Pattichis CS, Kell DB (2017). "What do we need to build explainable AI systems for the medical domain?" arXiv. Available from: https://arxiv.org/abs/1712.09923.

63. a, bSchoch N, Hoernicke M (2024). "NL2IBE – Ontology-controlled Transformation of Natural Language into Formalized Engineering Artefacts". In: 2024 IEEE Conference on Artificial Intelligence (CAI), IEEE, Singapore, Singapore, pp. 997–1004. doi:10.1109/CAI59869.2024.00182. https://ieeexplore.ieee.org/document/10605389/.

64. ^Guu K, Lee K, Tung Z, Pasupat P, Chang M (2020). "Retrieval augmented language model pre-training". In: International conference on machine learning. PMLR, pp. 3929–3938.

65. a, bLuo Y, Yang Z, Meng F, Li Y, Zhou J, Zhang Y (2024). An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning. Available from: https://arxiv.org/abs/2308.08747.

66. ^Zhai Y, Tong S, Li X, Cai M, Qu Q, Lee YJ, Ma Y (2024). "Investigating the catastrophic forgetting in multimodal large language model fine-tuning". In: Conference on Parsimony and Learning. PMLR, pp. 202–227. Available from: https://proceedings.mlr.press/v234/zhai24a.html.

67. ^Li H, Ding L, Fang M, Tao D (2024). "Revisiting Catastrophic Forgetting in Large Language Model Tuning". arXiv. doi:10.48550/arXiv.2406.04836. Available from: http://arxiv.org/abs/2406.04836.

68. ^Rolnick D, Ahuja A, Schwarz J, Lillicrap TP, Wayne G (2019). "Experience Replay for Continual Learning". arXiv. doi:10.48550/arXiv.1811.11682. Available from: http://arxiv.org/abs/1811.11682.

69. ^Merlin G, Lomonaco V, Cossu A, Carta A, Bacciu D (2022). "Practical Recommendations for Replay-Based Continual Learning Methods". In: Image Analysis and Processing, ICIAP 2022 Workshops, PT II. Lect

*ure Notes in Computer Science. **13374**. CHAM: Springer Nature, pp. 548–559.*

70. ^*Rusu AA, Rabinowitz NC, Desjardins G, Soyer H, Kirkpatrick J, Kavukcuoglu K, Pascanu R, Hadsell R (2022). "Progressive Neural Networks". arXiv. doi:10.48550/arXiv.1606.04671. Available from: http://arxiv.org/abs/1606.04671.*

71. ^*Pfeiffer J, Kamath A, Rücklé A, Cho K, Gurevych I (2021). "AdapterFusion: Non-destructive task composition for transfer learning". In: EACL 2021 - 16th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference, pp. 487–503.*

72. ^*Shumailov I, Shumaylov Z, Zhao Y, Gal Y, Papernot N, Anderson R (2024). "The Curse of Recursion: Training on Generated Data Makes Models Forget". arXiv. Available from: http://arxiv.org/abs/2305.17493. arXiv:2305.17493 [cs].*

73. ^*Baioumy M, Cheema A (2024). AI x Crypto Primer. Technical Report. University of Oxford. Available from: https://alexcheema.github.io/AIxCryptoPrimer.pdf.*

74. a, b*Tjondronegoro DW (2024). Strategic AI Governance: Insights from Leading Nations. doi:10.48550/arXiv.2410.01819.*

75. ^*Aguilar FJ (1967). Scanning the business environment. Macmillan.*

76. a, b, c, d, e*Xie Y, Yi J, Shao J, Curl J, Lyu L, Chen Q, Xie X, Wu F (2023). "Defending ChatGPT against jailbreak attack via self-reminders". Nature Machine Intelligence. **5** (12): 1486–1496. doi:10.1038/s42256-023-00765-8. https://www.nature.com/articles/s42256-023-00765-8.*

77. ^*Frankfurt HG (2009). On Bullshit. Princeton: Princeton University Press. ISBN 9781400826537.*

78. ^*Etherington D (2021). "MIT researchers develop a new 'liquid' neural network that's better at adapting to new info". TechCrunch. Available from: https://techcrunch.com/2021/01/28/mit-researchers-develop-a-new-liquid-neural-network-thats-better-at-adapting-to-new-info.*

Declarations

Funding: We thank Consensys Software Inc for funding this research.

Potential competing interests: No potential competing interests to declare.