International Journal of Science and Research Archive
eISSN: 2582-8185
Cross Ref DOI: 10.30574/ijsra
Journal homepage: https://ijsra.net/

(RESEARCH ARTICLE)

Check for updates

Building an LLM from Scratch

ML Sharma, Sunil Kumar, Rajveer Mittal, Shubhankar Rai *, Akshat Jain, Anurag Gandhi, Swayam Nagpal, Anurag Ranjan, Riya Yadav and Vatshank Mishra

Department of Electronics and Communication, Maharaja Agrasen Institute of Technology, Delhi, India.

International Journal of Science and Research Archive, 2025, 15(01), 1426-1434

Publication history: Received on 22 February 2025; revised on 22 April 2025; accepted on 24 April 2025

Article DOI: https://doi.org/10.30574/ijsra.2025.15.1.1140

Abstract

In this work, the development of a basic large language model (LLM) has been presented, with a primary focus on the pre-training process and model architecture. A simplified transformer-based design has been implemented to demonstrate core LLM principles with the incorporation of reinforcement learning techniques. Key components such as tokenization, and training objectives have been discussed to provide a foundational understanding of LLM construction. Additionally, an overview of several established models—including GPT-2, LLaMA 3.1, and DeepSeek—has been provided to contextualize current advancements in the field. Through this comparative and explanatory approach, the essential building blocks of large-scale language models have been explored in a clear and accessible manner.

Keywords: Pretraining; Introduction to the Neural Networks in LLM; Transformer Architecture; Post Training; Post Training with Reinforcement Learning

1. Introduction

Large Language Models play an important role in this new era of Artificial Intelligence and Machine Learning. The advancement and development of Large Language Models(LLM) has marked a breakthrough in the field of natural language processing, which enables and helps machines in understanding and interpreting human-like text in minutes and with a lot of accuracy.

Building an LLM from scratch involves an in-depth knowledge of topics like pre-training, tokenization, model training, transformer architecture, and post-training. reinforcement learning and many other things.

This research paper examines the complications of building an LLM from scratch, focusing on pre-training, post-training, transformer architecture, and the role of these factors in achieving state-of-the-art performance in several tasks.

* Corresponding author: Shubhankar Rai Email:- shubhankarrai2018@gmail.com

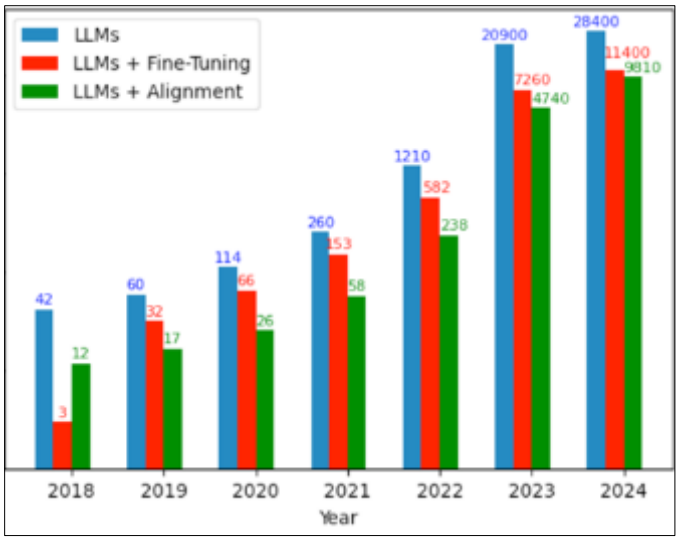

Figure 1 The trend of papers released over the years containing keywords "Large Language Model", "Large Language Model + Fine-Tuning", and "Large Language Model + Alignment"

2. Pre-training

2.1. Download and preprocess the data from the internet

To initiate the pretraining process, data were downloaded and preprocessed from publicly available internet sources. The dataset utilized for this purpose is the FineWeb[6] dataset, which is hosted on Hugging Face and contains approximately 15 trillion tokens, amounting to around 44TB of disk space. Before training, a multi-stage filtering and cleaning pipeline was applied.

2.1.1. URL Filtering

In the first stage, URL filtering was performed to eliminate undesirable sources. A blocklist was employed to exclude URLS associated with malicious, adult, or racially inappropriate content. This step was necessary to ensure that the model was not exposed to harmful or biased material.

2.1.2. Text Extraction

Since the collected web pages were in raw HTML format, a text extraction process was implemented. HTML tags and markup content were removed, and only the clean textual data was retained. This was done to prevent the model from learning non-linguistic patterns and to ensure training occurred on meaningful natural language text.

2.1.3. Language Filtering

Language detection was carried out using a classifier embedded within the FineWeb[6] pipeline. Each web page was analyzed, and only those with more than 65% English content were preserved. It should be noted that for other language-specific models, such as those intended for Hindi, the use of FineWeb[6] might not be appropriate. In such cases, custom datasets should be created to align with the intended language and use case, as inadequate language representation could impair model performance.

2.1.4. PII Removal (Personally Identifiable Information)

Finally, a PII removal step was included to sanitize the dataset. Sensitive information such as physical addresses, phone numbers, and social security numbers was identified and removed. This measure was taken to comply with ethical AI standards and to prevent the unintentional generation of personal data during inference.

2.2. Tokenization

Before textual data is fed into the neural network, a representation strategy must be determined. Since neural networks operate on fixed-length sequences of numbers, the input text must first be converted into numerical form. Each unique sequence of numbers corresponds to a distinct token, and these numerical tokens serve as the foundational units

through which language is processed. As these token combinations are finite and represent a valuable computational resource, efficient tokenization is critical.

To optimize token usage and reduce vocabulary size, a technique known as Byte Pair Encoding (BPE) is commonly employed. Through BPE, frequently occurring character or sub word pairs are iteratively merged, resulting in a compact and efficient vocabulary that balances granularity with representational power. For instance, GPT-4 utilizes approximately 1,000,277 unique token combinations, showcasing the extensive token space that modern language models operate within.

2.3. Model Training

In this step, the statistical relationships between tokens are modelled. Sliding windows of tokens, ranging from the beginning of a sequence up to a predefined maximum length, are selected and fed into the neural network. Given a vocabulary size of 1,000,277 tokens, the network is trained to predict the most probable next token in the sequence.

For each position in the input, the model assigns a probability distribution over the entire vocabulary. The objective is to assign a high probability to the correct next token while assigning lower probabilities to all others. The network's parameters are iteratively updated using gradient-based optimization techniques to minimize the difference between the predicted distribution and the actual next token. This process is repeated across all tokens in the training dataset, enabling the model to learn the statistical patterns of natural language over time.

These updates allow the model to improve its understanding of token relationships over time. A more detailed explanation of the underlying architecture, including Transformers and attention mechanisms, will be provided in a later section.

2.4. Inference

During inference, new data is generated by the model to observe the patterns that have been internalized within its parameters. When a sequence of tokens is provided as input, the next token is predicted based on learned probability distributions. In many cases, the generated tokens may not appear exactly as they did in the training data; instead, they are produced by the model based on its understanding and generalization of the learned patterns. These outputs can be seen as novel combinations that are inspired by, but not limited to, the training corpus.

2.5. GPT-2(Generative Pretrained Transformer) model

GPT-2 was trained using a single-window context length of 1024 tokens, meaning that only 1024 tokens were considered at a time during both training and inference [5]. The model was configured with approximately 1.5 billion parameters, distributed across 48 layers with 1600-dimensional hidden states [5]. The training was conducted on a dataset known as WebText, which consisted of roughly 40GB of cleaned and filtered Internet text [5]. In total, around 10 billion tokens were used during the pretraining process [5].

The training of GPT-2, which contains 1.5 billion parameters, was estimated to have cost between $50,000 and $100,000 in 2019. This cost was incurred due to the use of 256 V100 GPUS over several weeks [5]. If the same model were to be trained using current hardware and optimized training techniques, the cost would be significantly reduced, potentially falling within the range of $10,000 to $20,000. These improvements have been enabled by advancements in GPU efficiency, mixed-precision training, and better-distributed training frameworks.

It is a base model, mostly a token simulator.

2.6. LLaMA 3.1(Large Language Model Meta AI)

LLaMA[3], short for Large Language Model Meta AI, was developed by Meta and trained on over 15 trillion tokens. The context window length was extended up to 8,192 tokens, allowing the model to handle longer sequences of text. The largest variant, LLaMA 3.1 400B, was built with approximately 400 billion parameters. Despite its advanced capabilities, it remains a token generator, producing text based solely on learned statistical patterns rather than actual understanding or reasoning. The model was not exposed to any data after December 2023.

3. Introduction to Neural Networks in LLM

Large language models rely on specialized neural network architecture called the transformer. Neural networks consist of interconnected computational units called artificial neurons that learn to recognize patterns in numerical representations of real world data such as text, images, audio and time series. These neurons are organized into layers. The input layer receives raw data, one or more hidden layers extract and transform features, and the output layer produces the result. Each connection between neurons has a weight that is adjusted during training. Connections that lead to correct predictions are strengthened and those that cause errors are weakened. Through many iterations of this process the network learns to map inputs to the desired outputs. In transformers this learning is enhanced by self-attention which allows the model to focus on the most relevant parts of the input when generating its output.

3.1. Working

Weights and biases are the core parameters that allow a neural network to learn from data. When an image (for example, text rendered as pixels) is fed into the network, each pixel value—typically normalized between 0 and 1—is multiplied by a corresponding weight. Those weighted values are then summed and an additional bias term shifts the result before it passes through an activation function. During training, the network adjusts weights and biases: it increases those that help reduce prediction errors and decreases those that don't. By repeatedly tuning these parameters over many examples, the model learns to distinguish even very similar pixel patterns and produce accurate outputs.

3.1.1. Role of weights

Weights play a crucial role by scaling the input signals, either amplifying or diminishing their influence on the output. A high weight indicates that the corresponding input has a strong and positive impact on the neuron's output, meaning the network considers it important.

3.1.2 Role of biases:

Bias lets a neuron fire even if all inputs are zero by shifting its activation threshold, helping the model fit data more accurately. In practice you compute a weighted sum of inputs plus the bias, then apply an activation function. For example:

$$z = W1\ X1 + W2\ X2 + \ldots + Wn\ Xn + b$$

output = f(z)

Here, each Wi is a weight, Xi an input, b the bias, and f the activation function.

4. Transformer architecture

The Transformer architecture has revolutionized the field of Natural Language Processing (NLP) and now serves as the foundation for modern Large Language Models (LLMS). First introduced in the landmark paper "Attention is All You Need"[12] (Vaswani et al., 2017), transformers marked a major shift from traditional sequential models by introducing parallelized self-attention mechanisms. This breakthrough enabled the effective modelling of long-range dependencies and significantly improved scalability with increasing model size. For anyone aiming to build LLMS from scratch, a strong grasp of the Transformer architecture is essential. These models are not only scalable with parameter growth but also versatile across a wide range of NLP tasks. This research explores the fundamental components of the Transformer model, recent advancements, scalability features, and key implementation details, offering valuable insights for designing custom LLMS.

Modern large language models predominantly rely on the Transformer architecture, originally designed for machine translation tasks, particularly to translate English text into German and French. Introduced in 2017 through the "Attention is All You Need"[12] paper, the Transformer replaced recurrent architectures with attention-based mechanisms, allowing for better performance and training efficiency.

Here is a simplified diagram of the Transformer Architecture, illustrating the process of translating English into German.

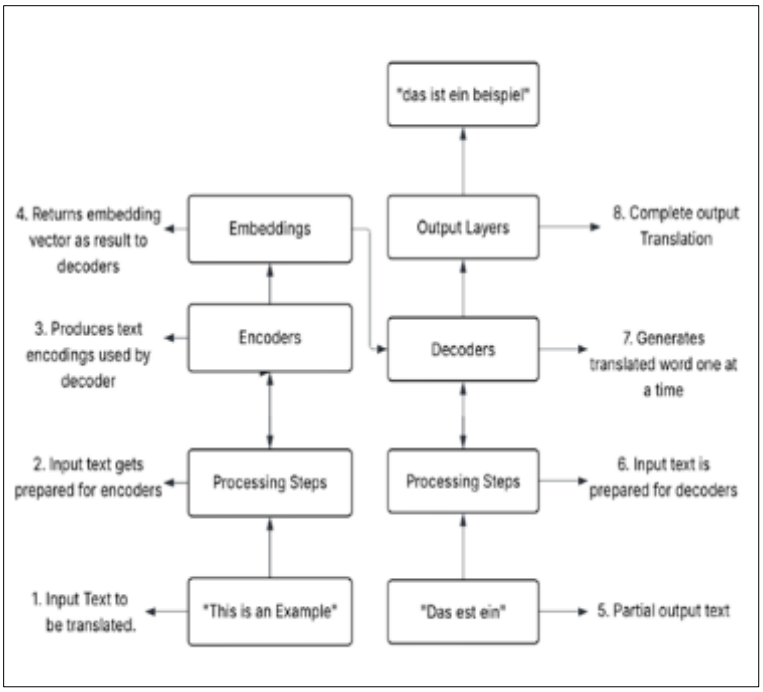

Figure 2 Sequence-to-sequence (encoder-decoder) architecture

4.1. Transformer models can be broadly categorised into three types

Encoder-only models focus solely on understanding the input text by converting it into dense vector representations. These vectors can represent the entire sequence or individual tokens, making these models suitable for tasks such as text classification, sentiment analysis, and named entity recognition. Prominent examples include BERT[8] (Devlin et al., 2019), RoBERTa[10] (Liu et al., 2019), and DeBERTa[11] (He et al., 2021). While highly effective at understanding language, encoder-only models are not designed for text generation, which limits their use in generative applications.

Encoder-decoder models, also known as sequence-to-sequence models, use the encoder to capture a contextual representation of the input and the decoder to generate the output sequence token-by-token. These models are well-suited for tasks such as machine translation, text summarization, and image captioning. Notable examples include the original Transformer model[12] (Vaswani et al., 2017), T5[13] (Raffel et al., 2020), and BART[14] (Lewis et al., 2020) . These models also support the use of control tokens to guide the output, for instance, specifying a target language or adjusting grammar complexity.

Decoder-only models are autoregressive language models that generate text by predicting one token at a time based on the previous ones. These models are especially useful for open-ended generation tasks such as chatbots, creative writing, and question-answering. Popular examples include GPT-3[15] (Brown et al., 2020), PaLM[16] (Chowdhery et al., 2022), and LLaMA[9] (Touvron et al., 2023). Decoder-only models are typically controlled via prompting—either zero-shot or few-shot—and do not require task-specific fine-tuning. However, they can be less interpretable and harder to control due to the lack of separation between input and instructions.

5. Post Training

Instead of explicitly programming conversations, a shift in focus toward example-driven learning is adopted. The assistant is trained using conversational data labelled by human annotators, who provide ideal responses to prompts. These examples guide the model in learning how an assistant is expected to behave. Following this, the original pretraining dataset is discarded, and the model is further trained on the curated conversational dataset. Through this process, the model is statistically adjusted to generate responses that align with human preferences. During inference, responses are produced based on the patterns learned from these annotated conversations. Although this stage resembles the pretraining process, it is typically shorter in duration and more focused in scope.

For post-training, the conversations are required to be tokenized using a suitable encoding scheme. Since the model cannot process raw text directly, each dialogue is converted into a sequence of tokens through this encoding. This

ensures that conversational structures—such as turns, roles (user/assistant), and context—are preserved and interpreted correctly by the model during training and inference.

During inference, the model is prompted to generate the most probable sequence of conversational tokens based on the given input. A response is produced not by retrieving an exact match from training data, but by generating a statistically likely continuation that resembles patterns observed during post-training. As a result, the output may not be identical to any specific example seen during training but is influenced by the conversational structure and tone present in the fine-tuning dataset.

In 2022, a research paper titled "Training language models to follow instructions with human feedback"[2] was published by OpenAI. In Section 3.3 of the paper, it was described that approximately 40 contractors were hired through platforms such as Upwork and ScaleAI. These contractors were tasked with generating a high-quality dataset for fine-tuning language models to better follow instructions. A detailed set of labelling guidelines was provided to them, and the dataset was constructed by adhering to these specifications. Despite its significance, this dataset was not made publicly available by OpenAI.

However, efforts have been made by the open-source community to create similar datasets. One notable example is OpenAssistant, which has released its conversation datasets to support the development of instruction-following models. The first dataset, OASST1, contains around 41.6 MB of multilingual conversation data, comprising over 160,000 messages and more than 10,000 conversation trees, along with extensive human-provided ratings. A second release, OASST2, further expands on this dataset, offering additional annotated dialogues. These datasets provide valuable resources for researchers and developers aiming to fine-tune models for more helpful and conversational behaviour.

Since the release of the InstructGPT paper, state-of-the-art techniques and technologies have significantly evolved over the past 2–3 years. As a result, the process of dataset creation has increasingly been assisted by language models themselves. Instead of relying entirely on human annotators, many modern instruction-following datasets are now generated synthetically, with minimal human involvement. One such example is UltraChat, where large language models are used to generate conversations and responses, guided by prompts and templates. While human oversight is still incorporated, primarily for quality control and occasional intervention, the majority of the data is produced by models. This shift has enabled the creation of large-scale instruction datasets at a much faster pace and lower cost than traditional methods.

5.1. Hallucinations

It is when information is fabricated by the LLMS despite not being known to them. Although this issue has improved compared to previous years, it has not yet been fully addressed. In the training set, the conversations are written by human labellers who have conducted proper research and are confident in their responses, as they know they are correct. However, the LLM being trained imitates the behaviour of the human labourers and generates answers based on the statistical patterns in the data, essentially making its best guess.

In the paper "Llama 3: Herd of Models"[4], Meta addressed this issue in Section 4.3.6 by applying the principle of "knowing what it knows" rather than attempting to add new knowledge. Instead of answering questions it was unsure about, the model was trained to refuse when necessary. To implement this, small chunks of data were taken from the model's original pretraining dataset. The model was prompted to generate questions based on these snippets, followed by generating answers to those questions. It was then used as a judge to evaluate the correctness and informativeness of its responses by comparing them to the original context. If the model consistently produced informative but incorrect answers, it was trained to refuse to answer instead, helping reduce hallucinations and ensuring it only responds when confident in its knowledge.

The second mitigation involves enabling the model to search for answers using tools that access the internet. Certain trigger tokens are used to initiate these tools, and the retrieved content is inserted into the context window, which is then fed into the neural network. The model only needs to be shown how to utilize these tools, after which adaptation occurs automatically.

Since the model is not truly human, it is unable to reflect on itself. Therefore, when a question like "What are you?" or "Who are you?" is asked, the model might not be able to respond accurately. To address this, the model must be explicitly fine-tuned using over 100 example conversations, so that it learns to respond correctly.

5.2. Thinking Patterns

The model is guided to produce intermediate reasoning steps before arriving at a final answer in Chain-of-Thought Reasoning. This approach is generally favoured for solving complex problems, as it has been shown to improve accuracy by making the model's internal reasoning process more transparent. Although more tokens are consumed, leading to higher computational costs and slower generation, this method ensures that the answer is derived through logical progression. As a result, confidence in the model's output is increased.

In contrast, direct answering involves predicting the final answer token immediately, without showing intermediate steps. This technique is computationally cheaper and faster, as fewer tokens are generated. However, it is often considered less reliable for complex tasks, since the answer is not supported by visible reasoning. The absence of a logical path can reduce trust in the model's output, especially in high-stakes or technical domains.

Both methods have their strengths and weaknesses depending on the use case. Direct answering may be preferred for simple factual queries where speed and efficiency are valued. However, for math, logic, or multi-step problems, Chain-of-Thought prompting is usually favoured due to its transparency and higher success rate. In recent models, CoT has increasingly been adopted to enhance interpretability and correctness, especially in domains where reasoning plays a central role.

Large Language Models are not inherently strong at spelling accuracy, especially for rare or unusual words. This limitation arises because text is processed as a sequence of tokens, rather than individual characters. Since spelling variations often produce different combinations of tokens, it becomes difficult for the model to reason about correct spellings unless those specific forms were seen during training. Tokenization can break words into subword units in unpredictable ways, and as a result, spelling errors are not easily isolated or corrected by the model. Handling spelling effectively is further complicated when subword boundaries do not align well with phonetic or orthographic patterns.

6. Post Training with Reinforcement Learning

Since a supervised fine-tuned model has already been obtained, the next step involves reinforcement learning. In this phase, token sequences are not meant to be explicitly created for the language model. Instead, the best possible token sequences — those that reliably lead to correct answers given a prompt — are meant to be discovered by the model itself through reinforcement learning and trial and error.

In this case, various types of solutions are explored to determine which ones perform best. As an exercise, around 100 responses can be generated from a single prompt, and the correct ones are selected. Among those, the most optimal answer is chosen and used repeatedly for training. It must be kept in mind that training is not conducted on just a single prompt, but on thousands — possibly tens of thousands — of different prompts. For each of these, the model is trained tens of thousands of times on the correct responses. Through this process, the model gradually discovers which types of token sequences consistently lead to correct answers.

The pretraining and supervised fine-tuning were considered the standard procedures to create a functioning LLM, while the reinforcement learning component is relatively new and is currently in its developmental stage.

6.1. DeepSeek R1

The RL part has been primarily utilised efficiently by the DeepSeek R1 model, as presented in the paper "DeepSeek-R1: Incentivising Reasoning Capability in LLMS via Reinforcement Learning."[3] The AIME accuracy of DeepSeek-R1-Zero was evaluated through training. AIME problems—advanced mathematics questions—were posed to the model, and 16 responses were sampled to calculate the overall average accuracy, ensuring a stable evaluation.

It can be observed that the model initially performed poorly; however, as the model was progressively updated, its accuracy was seen to increase. Even more impressive than the quantitative results of solving problems with greater accuracy were the qualitative aspects through which the results were achieved. It was noted, particularly in the graph on page 8 of the paper, that longer token sequences were being used to attain higher accuracy. These solutions were extensive and detailed.

Much like human problem-solving behaviour, the model was found to evaluate all aspects of a given problem, considering multiple approaches, engaging in out-of-the-box thinking, and demonstrating reasoning patterns that humans might use when solving math problems. Such behaviour could only be enabled through reinforcement learning.

It could not have been hardcoded. The model was allowed to discover new ways to think, manipulate problems, and explore varied approaches independently.

6.2. Reinforcement Learning from Human Feedback

Up to this point, the training of LLMS has been focused on using the best possible solutions for a given prompt. However, for prompts that are unverifiable or subjective, such as "Top 10 best holiday locations in the world" or joke generation, human intervention is required to judge the quality of the responses. Since this process is labour-intensive, a partial solution was proposed in the paper "Fine-Tuning Language Models from Human Preferences."[1]

In that paper, it was suggested that a separate neural network, referred to as the reward model, be trained to imitate human scoring. Human annotators were asked to score multiple responses (also known as rollouts) to subjective prompts, and those scores were then used to train the reward model. Once trained, this reward model acted as a simulator of human preferences, allowing reinforcement learning to be performed against it. This enabled the model to learn how to optimize for responses that align with human-like evaluations, even when ground truth answers do not exist.

6.2.1. Advantages and Disadvantages of Reinforcement Learning from Human Feedback (RLHF)

Advantages

Reinforcement Learning from Human Feedback (RLHF) offers significant benefits in aligning large language models (LLMS) with human values and expectations. One major advantage is its ability to guide models in generating responses that are more helpful, safe, and contextually appropriate, particularly in open-ended or ambiguous scenarios where correctness is subjective. Additionally, RLHF enhances user satisfaction by enabling models to better understand and adapt to nuanced human preferences. It also facilitates the training of models in tasks where explicit supervision is infeasible, thus broadening the range of problems that LLMS can address effectively.

Disadvantages

Despite its advantages, RLHF comes with notable challenges. Collecting high-quality human feedback is resource-intensive, requiring significant time, effort, and expert annotation. Moreover, the reward models used to approximate human judgment may introduce inaccuracies or biases, leading to unintended behaviours during reinforcement learning. Training stability is another concern, as models may learn to exploit imperfections in the reward model—a phenomenon known as reward hacking. Finally, if human feedback itself contains biases, these may be reinforced and magnified in the model's outputs, raising ethical and fairness concerns.

7. Future Scope

Future studies can improve LLM performance and reliability by creating more efficient and scalable alternatives to RLHF. This includes enhancing synthetic preference modeling and integrating offline reinforcement learning. There is also potential in expanding LLM reasoning capabilities beyond mere token-level predictions through structured multi-agent prompting and tool integration. As LLMs are increasingly tasked with making judgments on subjective or context-specific prompts, developing adaptive reward models that generalize human preferences across various tasks will be critical.

- Finally, exploring multilingual and multimodal extensions of alignment
- Techniques can make these models more globally usable and robust across real-world applications.

8. Conclusion

In this study, the core techniques used to build and align large language models (LLMS) were explored, focusing on pretraining, supervised fine-tuning, and the emerging role of reinforcement learning. While supervised fine-tuning forms the foundation of instruction-following models, it falls short in ensuring reliability, factuality, and alignment with human intent, particularly in subjective or unverifiable tasks. Techniques like chain-of-thought prompting have demonstrated clear advantages in improving reasoning and interpretability, and hallucination mitigation strategies have become vital for responsible model behaviour. Reinforcement Learning from Human Feedback (RLHF) presents a powerful method for refining model outputs by allowing them to learn from preference-based evaluations.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Fine-Tuning Language Models from Human Preferences Christiano, Paul F., et al. (2017). "Deep reinforcement learning from human preferences." Advances in Neural Information Processing Systems, 30.

[2] InstructGPT (Training language models to follow instructions with human feedback) Ouyang, Long, et al. (2022). "Training language models to follow instructions with human feedback." arXiv preprint arXiv:2203.02155.

[3] DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning DeepSeek-VL Team. (2024). "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning." arXiv preprint arXiv:2403.17181.

[4] Llama 3 Herd of Models (Meta AI) Meta AI. (2024). "Llama 3: Herd of Models." Meta AI Research, .

[5] Language Models are Unsupervised multitasking learners Radford, Alec, et al. (2019). OpenAI. [Technical Report]

[6] Hugging Face, FineWeb: Decanting the Web for the Finest Text Data at Scale, 2023. https://huggingface.co/spaces/HuggingFaceFW/blogpost-fineweb-v1

[7] Andrej Karpathy , "Deep Dive into LLMs like ChatGPT", YouTube, 2025. https://www.youtube.com/watch?v=7xTGNNLPyMI

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, GitHub, 2018. https://github.com/google-research/bert.

[9] Meta AI, LLaMA: Open and Efficient Foundation Language Models, GitHub, 2023. https://github.com/meta-llama/llama.

[10] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov, RoBERTa: A Robustly Optimized BERT Pretraining Approach, arXiv:1907.11692, 2019. https://arxiv.org/abs/1907.11692.

[11] Pengcheng He, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, DeBERTa: Decoding-enhanced BERT with Disentangled Attention, GitHub, 2020. https://github.com/microsoft/DeBERTa.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin, Attention Is All You Need, arXiv:1706.03762, 2017. https://arxiv.org/abs/1706.03762.

[13] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J. Liu, Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, GitHub, 2019. https://github.com/google-research/text-to-text-transfer-transformer.

[14] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, Luke Zettlemoyer, BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension, GitHub, 2019. https://github.com/facebookresearch/fairseq/tree/main/examples/bart.

[15] OpenAI, GPT-3: Language Models are Few-Shot Learners, GitHub, 2020. https://github.com/openai/gpt-3.

[16] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, N. Fiedel, "PaLM: Scaling Language Modeling with Pathways," arXiv preprint arXiv:2204.02311, 2022.