RESEARCH-ARTICLE

A Comprehensive Overview of Large Language Models

HUMZA NAVEED, The University of Sydney, Sydney, NSW, Australia

ASAD ULLAH KHAN, University of Engineering and Technology, Lahore, Lahore, Punjab, Pakistan

SHI QIU, Chinese University of Hong Kong, Hong Kong, Hong Kong

MUHAMMAD SAQIB, University of Technology Sydney, Sydney, NSW, Australia

SAEED ANWAR, King Fahd University of Petroleum and Minerals, Dhahran, Ash Sharqiyah, Saudi Arabia

MUHAMMAD USMAN, Ontario Tech University, Oshawa, ON, Canada

View all

Published: 19 August 2025
Online AM: 18 June 2025
Accepted: 26 May 2025
Revised: 03 April 2025
Received: 30 October 2024

A Comprehensive Overview of Large Language Models

HUMZA NAVEED, The University of Sydney, Sydney, Australia
ASAD ULLAH KHAN, University of Engineering and Technology, Lahore, Pakistan
SHI QIU, The Chinese University of Hong Kong, Hong Kong, Hong Kong
MUHAMMAD SAQIB, University of Technology Sydney, Sydney, Australiaand Commonwealth Scientific and Industrial Research Organisation, Canberra, Australia
SAEED ANWAR, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia and SDAIAKFUPM Joint Research Center for Artificial Intelligence, Dhahran, Saudi Arabia
MUHAMMAD USMAN, University of Ontario Institute of Technology, Oshawa, Ontario, Canada
NAVEED AKHTAR, University of Melbourne VCCC, Parkville, Australiaand The University of Western Australia, Perth, Australia
NICK BARNES, Australian National University, Canberra, Australia
AJMAL MIAN, The University of Western Australia - Perth Campus, Perth, Australia

Large Language Models (LLMs) have recently demonstrated remarkable capabilities in natural language processing tasks and beyond. This success of LLMs has led to a large influx of research contributions in this direction. These works encompass diverse topics such as architectural innovations, better training strategies, context length improvements, fine-tuning, multimodal LLMs, robotics, datasets, benchmarking, efficiency, and more. With the rapid development of techniques and regular breakthroughs in LLM research, it has become considerably challenging to perceive the bigger picture of the advances in this direction. Considering the rapidly emerging plethora of literature on LLMs, it is imperative that the research community is able to benefit from a concise yet comprehensive overview of the recent developments in this field. This article provides an overview of the literature on a broad range of LLM-related concepts. Our self-contained comprehensive overview of LLMs discusses relevant background concepts along with covering the advanced topics at the frontier of research in LLMs. This review article is intended to provide not only a systematic survey but

The author/s would like to acknowledge the support received from Saudi Data and AI Authority (SDAIA) and King Fahd University of Petroleum and Minerals (KFUPM) under SDAIA-KFUPM Joint Research Center for Artificial Intelligence Grant No. JRC-AI-RFP-11.

Authors' Contact Information: Humza Naveed (corresponding author), The University of Sydney, Sydney, Australia; e-mail: humza_naveed@yahoo.com; Asad Ullah Khan, University of Engineering and Technology, Lahore, Pakistan; e-mail: aukhanee@gmail.com; Shi Qiu, The Chinese University of Hong Kong, Hong Kong, Hong Kong; e-mail: shiqiu@cse.cuhk.edu.hk; Muhammad Saqib, University of Technology Sydney, Sydney, Australia and Commonwealth Scientific and Industrial Research Organisation, Canberra, Australia; e-mail: muhammad.saqib@data61.csiro.au; Saeed Anwar, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia and SDAIAKFUPM Joint Research Center for Artificial Intelligence, Dhahran, Saudi Arabia; e-mail: saeed.anwar@kfupm.edu.sa; Muhammad Usman, University of Ontario Institute of Technology, Oshawa, Ontario, Canada; e-mail: muhammad.usman8@ontariotechu.ca; Naveed Akhtar, University of Melbourne VCCC, Parkville, Australia and The University of Western Australia, Perth, Australia; e-mail: naveed.akhtar1@unimelb.edu.au; Nick Barnes, Australian National University, Canberra, Australia; e-mail: nick.barnes@anu.edu.au; Ajmal Mian, The University of Western Australia - Perth Campus, Perth, Australia; e-mail: ajmal.mian@uwa.edu.au.

ACM 2157-6912/2025/8-ART106
https://doi.org/10.1145/3744746

also a quick, comprehensive reference for the researchers and practitioners to draw insights from extensive, informative summaries of the existing works to advance the LLM research.

CCS Concepts: • **Computing methodologies** → **Natural language processing**;

Additional Key Words and Phrases: Large Language Models, LLMs, chatGPT, Augmented LLMs, Multimodal LLMs, LLM training, LLM Benchmarking

ACM Reference format:
Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2025. A Comprehensive Overview of Large Language Models. *ACM Trans. Intell. Syst. Technol.* 16, 5, Article 106 (August 2025), 72 pages.
https://doi.org/10.1145/3744746

1 Introduction

Language plays a fundamental role in facilitating communication and self-expression for humans and their interaction with machines. The need for generalized models stems from the growing demand for machines to handle complex language tasks, including translation, summarization, **information retrieval (IR)**, conversational interactions, and so on. Recently, significant breakthroughs have been witnessed in language models, primarily attributed to transformers [1], increased computational capabilities, and the availability of large-scale training data. These developments have brought about a revolutionary transformation by enabling the creation of **large language models (LLMs)** that can approximate human-level performance on various tasks [2, 3]. LLMs have emerged as cutting-edge AI systems that can process and generate text with coherent communication [4] and generalize to multiple tasks [5, 6].

The historical progress in **natural language processing (NLP)** evolved from statistical to neural **language modeling (LM)** and then from **pre-trained language models (PLMs)** to LLMs. While conventional LM trains task-specific models in supervised settings, PLMs are trained in a self-supervised setting on a large corpus of text [7–9] with the aim of learning a generic representation that is shareable among various NLP tasks. After fine-tuning for downstream tasks, PLMs surpass the performance gains of traditional LM. The larger PLMs bring more performance gains, which has led to the transitioning of PLMs to LLMs by significantly increasing model parameters (tens to hundreds of billions) [10] and training dataset (many GBs and TBs) [10, 11]. Following this development, numerous LLMs have been proposed in the literature [6, 10–15]. An increasing trend in the number of released LLMs and names of a few significant LLMs proposed over the years are shown in Figures 1 and 2, respectively.

The early work on LLMs, such as T5 [10] and mT5 [11] employed transfer learning until GPT-3 [6] showed LLMs are zero-shot transferable to downstream tasks without fine-tuning. LLMs accurately respond to task queries when prompted with task descriptions and examples. However, pre-trained LLMs fail to follow user intent and perform worse in zero-shot settings than in few-shot. Fine-tuning them with task instructions data [16–19] and aligning with human preferences [20, 21] enhances generalization to unseen tasks, improving zero-shot performance significantly and reducing misaligned behavior.

In addition to better generalization and domain adaptation, LLMs appear to have emergent abilities, such as reasoning, planning, decision-making, **in-context learning (ICL)**, answering in zero-shot settings, and so on. These abilities are known to be acquired by them due to their gigantic scale even when the pre-trained LLMs are not trained specifically to possess these attributes [22–24]. Such abilities have led LLMs to be widely adopted in diverse settings, including multimodal,

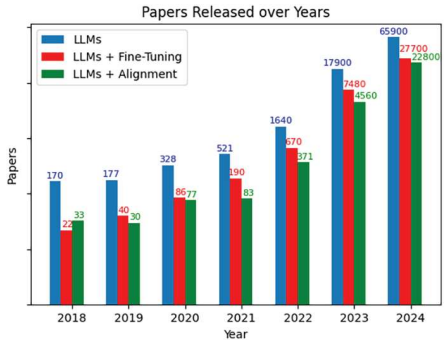

Fig. 1. The trend of papers released over the years containing keywords "Large Language Model," "Large Language Model + Fine-Tuning," and "Large Language Model + Alignment."

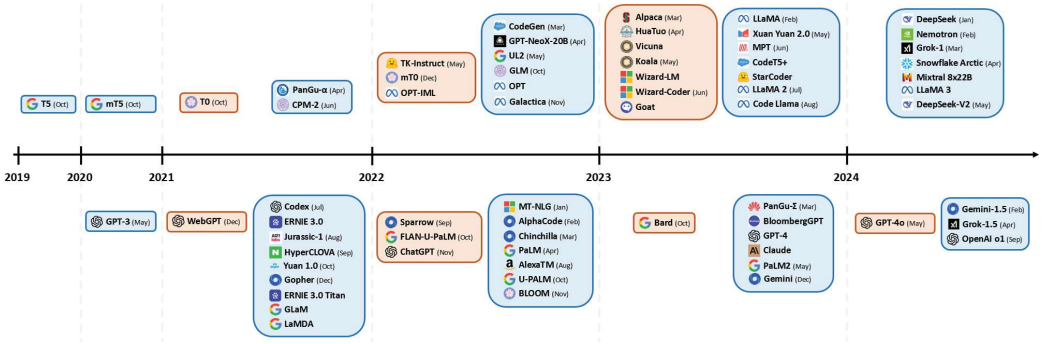

Fig. 2. Chronological display of LLM releases: blue cards represent "pre-trained" models, while orange cards correspond to "instruction-tuned" models. Models on the upper half signify open-source availability, whereas those on the bottom are closed-source. The chart illustrates the increasing trend toward instruction-tuned and open-source models, highlighting the evolving landscape and trends in NLP research.

robotics, tool manipulation, **question answering (QA)**, autonomous agents, and so on. Various improvements have also been suggested in these areas either by task-specific training [25–31] or better prompting [32].

The LLM abilities to solve diverse tasks with human-level performance come at the cost of slow training and inference, extensive hardware requirements, and higher running costs. Such requirements have limited their adoption and opened up opportunities to devise better architectures [15, 33–35] and training strategies [21, 36–41]. Parameter efficient tuning [38, 40, 41], pruning [42, 43], quantization [44, 45], knowledge distillation, and context length interpolation [46–49] among others are some of the methods widely studied for efficient LLM utilization.

Due to the success of LLMs on a wide variety of tasks, the research literature has recently experienced a large influx of LLM-related contributions. Researchers have organized the LLMs literature in surveys [50–53] and topic-specific surveys in [54–58]. In contrast to these surveys, our contribution focuses on providing a comprehensive yet concise overview of the general direction of LLM research. This article summarizes the architectural and training details of pre-trained LLMs. It delves deeper into the details of concepts like fine-tuning, **multimodal LLMs (MLLMs)**, augmented LLMs, datasets, evaluation, applications, challenges, and others to provide a self-contained comprehensive overview. A broader overview on different areas in LLMs is shown in Figure 3. Our key contributions are summarized as follows.

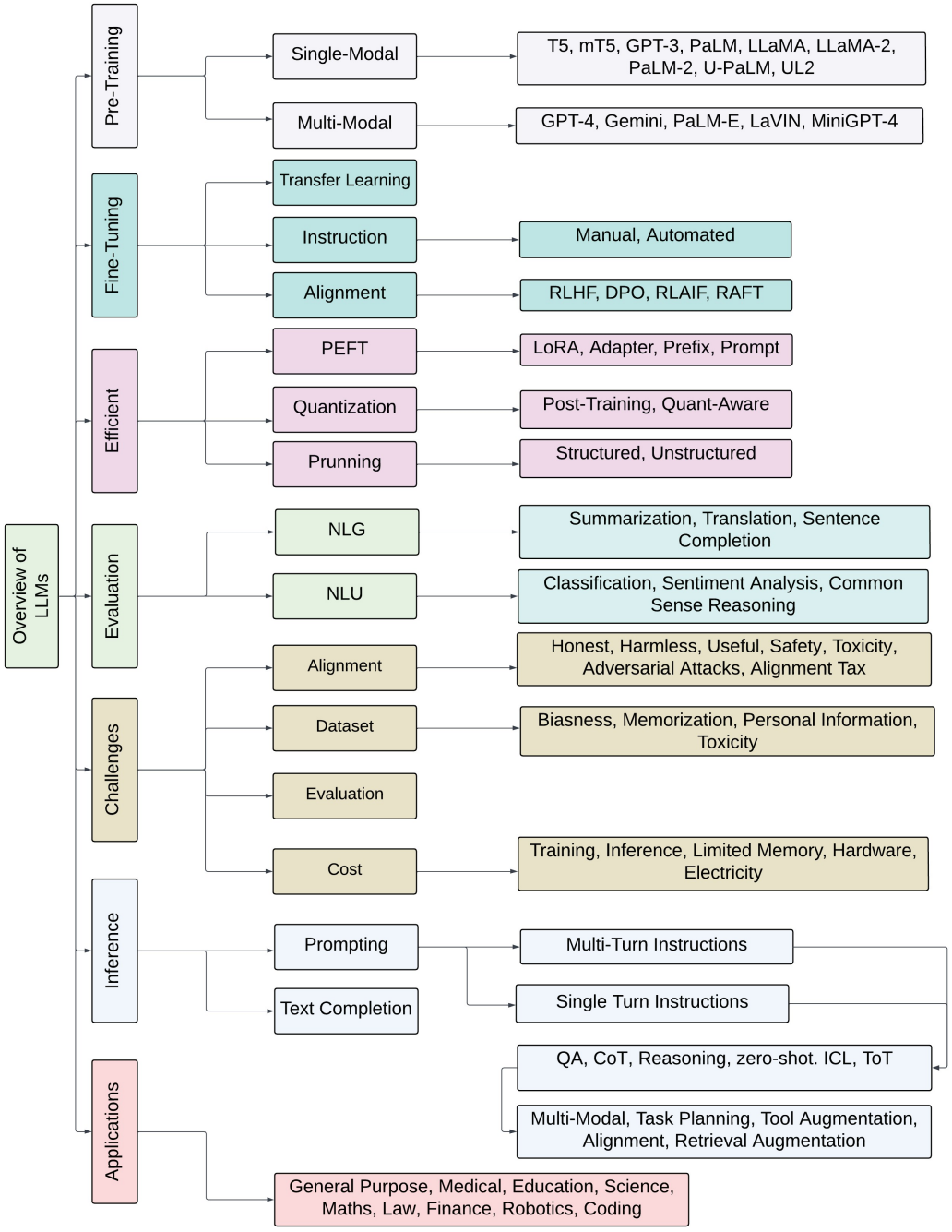

Fig. 3. A broader overview of LLMs, dividing LLMs into seven branches: (1) pre-training, (2) fine-tuning, (3) efficient, (4) inference, (5) evaluation, (6) applications, and (7) challenges.

—We present a survey on the developments in LLM research, providing a concise, comprehensive overview of the direction.
—We present extensive summaries of pre-trained models that include fine-grained details of architecture and training details.
—We summarize major findings of the popular contributions and provide a detailed discussion on the key design and development aspects of LLMs to help practitioners effectively leverage this technology.
—In this self-contained article, we cover a range of concepts to present the general direction of LLMs comprehensively, including background, pre-training, fine-tuning, MLLMs, augmented LLMs, LLMs-powered agents, datasets, evaluation, and so on.

We loosely follow the existing terminology to ensure a standardized outlook of this research direction. For instance, following [50], our survey discusses pre-trained LLMs with 10B parameters or more. We refer the readers interested in smaller pre-trained models to [51–53].

The organization of this article is as follows. Section 2 discusses the background of LLMs. Section 3 focuses on LLMs overview, architectures, training pipelines and strategies, fine-tuning, and utilization in different domains. Section 4 highlights the configuration and parameters that play a crucial role in the functioning of these models. Summary and discussions are presented in Section 3.8. The LLM training and evaluation, datasets, and benchmarks are discussed in Section 5, followed by challenges and future directions, and conclusion in Sections 8 and 9, respectively.

2 Background

We provide the relevant background to understand the fundamentals related to LLMs in this section. We briefly discuss necessary components in LLMs and refer the readers interested in details to the original works.

2.1 Tokenization

Tokenization [59] is an essential preprocessing step in LLM training that parses the text into non-decomposing units called tokens. Tokens can be characters, subwords [60], symbols [61], or words, depending on the tokenization process. Some of the commonly used tokenization schemes in LLMs include wordpiece [62], **byte pair encoding (BPE)** [61], and unigramLM [60]. Readers are encouraged to refer to [63] for a detailed survey.

2.2 Encoding Positions

The transformer processes input sequences in parallel and independently of each other. Moreover, the attention module in the transformer does not capture positional information. As a result, positional encodings were introduced in transformer [64], where a positional embedding vector is added to the token embedding. Variants of positional embedding include absolute, relative, or learned positional encodings. Within relative encoding, Alibi and RoPE are two widely used positional embeddings in LLMs.

Alibi [65]. It subtracts a scalar bias from the attention score that increases with the distance between token positions. This favors using recent tokens for attention.

RoPE [66]. It rotates query and key representations at an angle proportional to the token absolute position in the input sequence, resulting in a relative positional encoding scheme which decays with the distance between the tokens.

2.3 Attention in LLMs

Attention assigns weights to input tokens based on importance so that the model gives more emphasis to relevant tokens. Attention in transformers [64] calculates query, key, and value mappings

for input sequences, where the attention score is obtained by multiplying the query and key, and later used to weight values. We discuss different attention strategies used in LLMs below.

Self-Attention [64]. Calculates attention using queries, keys, and values from the same block (encoder or decoder).

Cross-Attention. It is used in encoder-decoder architectures, where encoder outputs are **key-value (KV)** pairs, and queries come from the decoder.

Sparse Attention [67]. Self-attention has $O(n^2)$ time complexity, which becomes infeasible for large sequences. To speed up the computation, sparse attention [67] iteratively calculates attention in sliding windows for speed gains.

Flash Attention [68]. Memory access is the major bottleneck in calculating attention using GPUs. To speed up, flash attention employs input tiling to minimize the memory reads and writes between the GPU high bandwidth memory and the on-chip SRAM.

2.4 Activation Functions

The activation functions serve a crucial role in the curve-fitting abilities of neural networks [69]. We discuss activation functions used in LLMs in this section.

Rectified Linear Unit (ReLU) [70]. The ReLU is defined as:

$$ReLU(x) = max(0, x). \tag{1}$$

Gaussian Error Linear Unit (GeLU) [71]. The GeLU is the combination of ReLU, dropout [72], and zoneout [73].

Gated Linear Unit (GLU) *Variants* [74]. The GLU [75] is a neural network layer that is an elementwise product ($\otimes$) of a linear transformation and a sigmoid transformed (σ) linear projection of the input given as:

$$GLU(x, W, V, b, c) = (xW + b) \otimes \sigma(xV + c), \tag{2}$$

where X is the input of layer and l, W, b, V, and c are learned parameters. Other GLU variants [74] used in LLMs are:

$$\begin{aligned} ReGLU(x, W, V, b, c) &= max(0, xW + b)\otimes, \\ GEGLU(x, W, V, b, c) &= GELU(xW + b) \otimes (xV + c), \\ SwiGLU(x, W, V, b, c, \beta) &= Swish\beta(xW + b) \otimes (xV + c). \end{aligned}$$

2.5 Layer Normalization

Layer normalization leads to faster convergence and is an integrated component of transformers [64]. In addition to LayerNorm [76] and RMSNorm [77], LLMs use pre-layer normalization [78], applying it before **multi-head attention (MHA)**. Pre-norm is shown to provide training stability in LLMs. Another normalization variant, DeepNorm [79] fixes the issue with larger gradients in pre-norm.

2.6 Distributed LLM Training

This section describes distributed LLM training approaches briefly. More details are available in [13, 37, 80, 81].

Data Parallelism. Data parallelism replicates the model on multiple devices where data in a batch gets divided across devices. At the end of each training iteration, weights are synchronized across all devices.

Tensor Parallelism. Tensor parallelism shards a tensor computation across devices. It is also known as horizontal parallelism or intra-layer model parallelism.

Pipeline Parallelism. Pipeline parallelism shards model layers across different devices. This is also known as vertical parallelism.

Model Parallelism. A combination of tensor and pipeline parallelism is known as model parallelism.

3D Parallelism. A combination of data, tensor, and model parallelism is known as 3D parallelism.

Optimizer Parallelism. Optimizer parallelism also known as zero redundancy optimizer [37] implements optimizer state partitioning, gradient partitioning, and parameter partitioning across devices to reduce memory consumption while keeping the communication costs as low as possible.

2.7 Libraries

Some commonly used libraries for LLMs training are as follows:

Transformers [82]. The library provides access to various pre-trained transformer models with APIs to train, fine-tune, infer, and develop custom models.

DeepSpeed [36]. A library for scalable distributed training and inference of deep learning models.

Megatron-LM [80]. It provides GPU-optimized techniques for large-scale training of LLMs.

JAX [83]. A Python library for high-performance numerical computing and scaleable machine learning. It can differentiate native Python and NumPy functions and execute them on GPUs.

Colossal-AI [84]. A collection of components to write distributed deep learning models.

BMTrain [81]. A library to write efficient stand-alone LLMs training code.

FastMoE [85]. Provides API to build **mixture-of-experts (MoE)** model in PyTorch.

MindSpore [86]. A deep learning training and inference framework extendable to mobile, edge, and cloud computing.

PyTorch [87]. A framework developed by Facebook AI Research lab to build deep learning models. The main features of PyTorch include a dynamic computation graph and a pythonic coding style.

Tensorflow [88]. A deep learning framework written by Google. The key features of TensorFlow are graph-based computation, eager execution, scalability, and so on.

MXNet [89]. Apache MXNet is a deep learning framework with support to write programs in multiple languages, including, Python, C++, Scala, R, and so on. It also provides support for dynamic and static computation graphs.

2.8 Data Preprocessing

This section briefly summarizes data preprocessing techniques used in LLMs training.

Quality Filtering. For better results, training data quality is essential. Some approaches to filtering data are: (1) classifier-based and (2) heuristics-based. Classifier-based approaches train a classifier on high-quality data and predict the quality of text for filtering, whereas heuristics-based approaches employ some rules for filtering like language, metrics, statistics, and keywords.

Data Deduplication. Duplicated data can affect model performance and increase data memorization; therefore, to train LLMs, data deduplication is one of the preprocessing steps. This can be performed at multiple levels, like sentences, documents, and datasets.

Privacy Reduction. Most of the training data for LLMs is collected through web sources. These data contain private information; therefore, many LLMs employ heuristics-based methods to filter information such as names, addresses, and phone numbers to avoid learning personal information.

2.9 Architectures

Here, we discuss the variants of the transformer architectures used in LLMs. The difference arises due to the application of the attention and the connection of transformer blocks. An illustration of attention patterns of these architectures is shown in Figure 4.

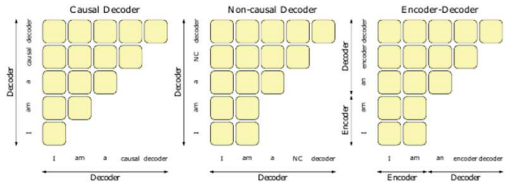

Fig. 4. An example of attention patterns in language models, image is taken from [93].

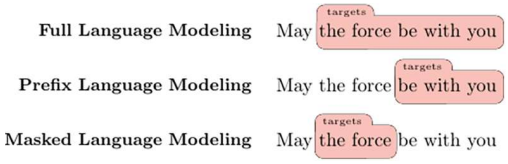

Fig. 5. An example of language model training objectives, image from [93].

Encoder-Decoder. This architecture processes inputs through the encoder and passes the intermediate representation to the decoder to generate the output. Here, the encoder sees the complete sequence utilizing self-attention, whereas the decoder processes the sequence one after the other with implementing cross-attention.

Causal Decoder. A type of architecture that does not have an encoder and processes and generates output using a decoder, where the predicted token depends only on the previous timesteps.

Prefix Decoder. It is also known as a non-causal decoder, where the attention calculation is not strictly dependent on the past information and the attention is bidirectional. An example of a non-causal attention mask is shown in Figure 4.

MoE. It is a variant of transformer architecture with parallel independent experts and a router to route tokens to experts. These experts are feed-forward layers after the attention block [90]. MoE is an efficient sparse architecture that offers comparable performance to dense models and allows increasing the model size without increasing the computational cost by activating only a few experts at a time [91, 92].

2.10 Pre-Training Objectives

This section describes LLMs pre-training objectives. For more details see the paper [93].

Full LM. An autoregressive LM objective where the model is asked to predict future tokens given the previous tokens, an example is shown in Figure 5.

Prefix LM. A non-causal training objective, where a prefix is chosen randomly and only remaining target tokens are used to calculate the loss. An example is shown in Figure 5.

Masked LM (MLM). In this training objective, tokens or spans (a sequence of tokens) are masked randomly and the model is asked to predict masked tokens given the past and future context. An example is shown in Figure 5.

Unified LM. Unified LM [94] is a combination of causal, non-causal, and masked language training objectives. Here in MLM, the attention is not bidirectional but unidirectional, attending either left-to-right or right-to-left context.

2.11 LLMs Scaling Laws

Scaling laws study the optimal combination of model parameters, dataset size, and computational resources that predict the improvement in the model performance. It has been shown that the loss scales according to the power law with model size, dataset size, and compute resources [95]. This

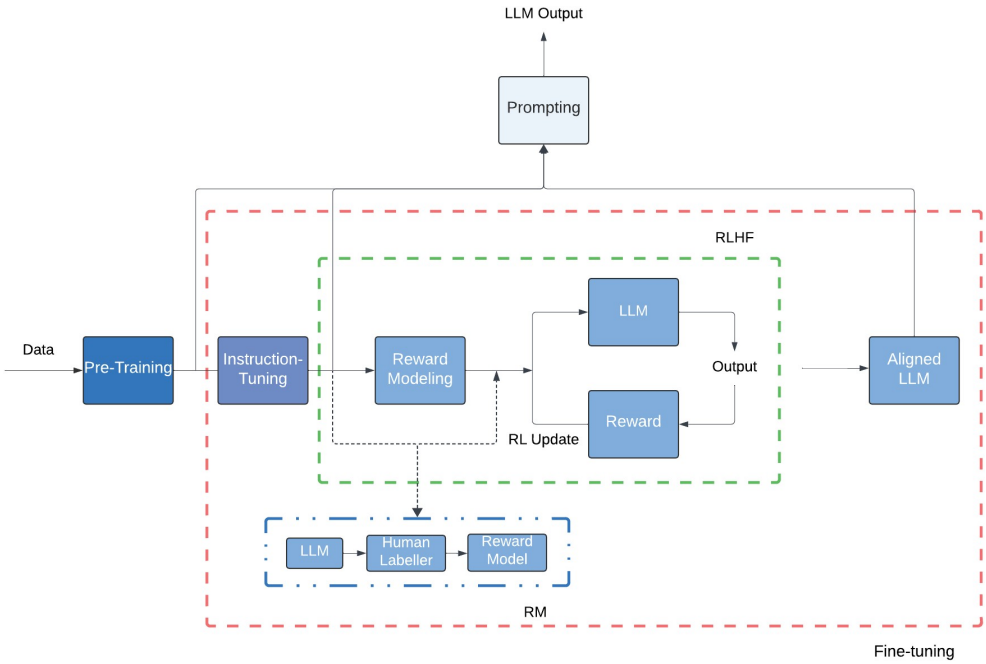

Fig. 6. A basic flow diagram depicting various stages of LLMs from pre-training to prompting/utilization. Prompting LLMs to generate responses is possible at different training stages like pre-training, instruction tuning, or alignment tuning. "RL" stands for reinforcement learning, "RM" represents reward modeling, and "RLHF" represents reinforcement learning with human feedback.

study suggests larger models are more important than big data for better performance. Another variant of scaling law [96] suggests the model size and the number of training tokens should be scaled equally.

2.12 LLMs Adaptation Stages

This section discusses the fundamentals of LLMs adaptation stages, from pre-training to fine-tuning for downstream tasks and utilization. An example of different training stages and inference in LLMs is shown in Figure 6. In this article, we refer to alignment tuning as aligning with human preferences, while occasionally the literature uses the term alignment for different purposes.

2.12.1 Pre-Training. In the very first stage, the model is trained in a self-supervised manner on a large corpus to predict the next tokens given the input. The design choices of LLMs vary from encoder-decoder to decoder-only architectures with different building blocks and loss functions in Sections 2.4, 2.5, and 2.10.

2.12.2 Fine-Tuning. There are different styles to fine-tune an LLM. This section briefly discusses fine-tuning approaches.

Transfer Learning. The pre-trained LLMs perform well for various tasks [6, 15]. However, to improve the performance for a downstream task, pre-trained models are fine-tuned with the task-specific data [10, 11], known as transfer learning.

Instruction Tuning. To enable a model to respond to user queries effectively, the pre-trained model is fine-tuned on instruction-formatted data, i.e., instruction and an input-output pair. Instructions

generally comprise multi-task data in plain natural language, guiding the model to respond according to the prompt and the input. This type of fine-tuning improves zero-shot generalization and downstream task performance. Details on formatting instruction data and its various styles are available in [16, 50, 97].

Alignment Tuning. LLMs are prone to generating false, biased, and harmful text. To make them helpful, honest, and harmless, models are aligned using human feedback. Alignment involves asking LLMs to generate unexpected responses and then updating their parameters to avoid such responses [20, 21, 98].

It ensures LLMs operate according to human intentions and values. A model is defined to be an "aligned" model if the model fulfills three criteria of helpful, honest, and harmless or "HHH" [99].

Researchers employ **reinforcement learning with human feedback (RLHF)** [100] for model alignment. In RLHF, a fine-tuned model on demonstrations is further trained with **reward modeling (RM)** and **reinforcement learning (RL)**, shown in Figure 6. Below, we briefly discuss RM and RL pipelines in RLHF.

RM. RM trains a model to rank generated responses according to human preferences using a classification objective. To train the classifier, humans annotate LLMs generated responses based on the HHH criteria.

RL. RL in combination with the reward model is used for alignment in the next stage. The previously trained reward model ranks LLM-generated responses into preferred vs. non-preferred, which is used to align the model with **proximal policy optimization (PPO)**. This process repeats iteratively until convergence.

2.12.3 Prompting/Utilization. Prompting is a method to query trained LLMs for generating responses, as illustrated in Figure 6. LLMs can be prompted in various prompt setups, where they can be adapted to the instructions without fine-tuning and in other cases with fine-tuning on data containing different prompt styles [16, 101, 102]. A good guide on prompt engineering is available at [32]. Below, we will discuss various widely used prompt setups.

Zero-Shot Prompting. LLMs are zero-shot learners and capable of answering queries never seen before. This style of prompting requires LLMs to answer user questions without seeing any examples in the prompt.

ICL. Also known as few-shot learning, here, multiple input-output demonstration pairs are shown to the model to generate the desired response. This adaptation style is also called few-shot learning. A discussion on formatting ICL templates is available in [16, 18, 50, 54].

Reasoning in LLMs. LLMs are zero-shot reasoners and can be provoked to generate answers to logical problems, task planning, critical thinking, and so on. with reasoning. Generating reasons is possible only by using different prompting styles, whereas to improve LLMs further on reasoning tasks many methods [16, 97] train them on reasoning datasets. We discuss various prompting techniques for reasoning below.

Chain-of-Thought (CoT). A special case of prompting where demonstrations contain reasoning information aggregated with inputs and outputs so that the model generates outcomes with step-by-step reasoning. More details on CoT prompts are available in [55, 101, 103].

Self-Consistency. Improves CoT performance by generating multiple responses and selecting the most frequent answer [104].

Tree-of-Thought (ToT). Explores multiple reasoning paths with possibilities to look ahead and backtrack for problem-solving [105].

Single-Turn Instructions. In this prompting setup, LLMs are queried only once with all the relevant information in the prompt. LLMs generate responses by understanding the context either in a zero-shot or few-shot setting.

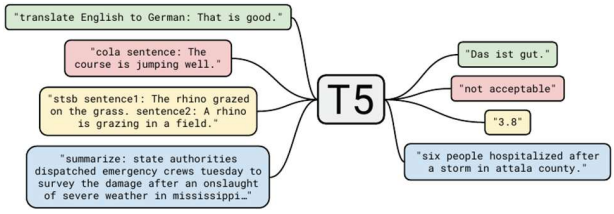

Fig. 7. Unified text-to-text training example, source image from [10].

Multi-Turn Instructions. Solving a complex task requires multiple interactions with LLMs, where feedback and responses from the other tools are given as input to the LLM for the next rounds. This style of using LLMs in the loop is common in autonomous agents.

3 LLMs

This section reviews LLMs, briefly describing their architectures, training objectives, pipelines, datasets, and fine-tuning details.

3.1 Pre-Trained LLMs

Here, we provide summaries of various well-known pre-trained LLMs with significant discoveries, changing the course of research and development in NLP. These LLMs have considerably improved the performance in NLU and NLG domains and are widely fine-tuned for downstream tasks. Moreover, we also identify key findings and insights of pre-trained LLMs in Tables 1 and 2 that improve their performance.

3.1.1 General Purpose.

T5 [10]. An encoder-decoder model employing a unified text-to-text training for all NLP problems is shown in Figure 7. T5 places layer normalization outside the residual path in a conventional transformer model [64]. It uses MLM as a pre-training objective where spans (consecutive tokens) are replaced with a single mask instead of separate masks for each token. This type of masking speeds up the training as it produces shorter sequences. After pre-training, the model is fine-tuned using adapter layers [106] for downstream tasks.

GPT-3 [6]. The GPT-3 architecture is the same as the GPT-2 [5] but with dense and sparse attention in transformer layers similar to the sparse transformer [67]. It shows that large models can train on larger batch sizes with a lower learning rate to decide the batch size during training, GPT-3 uses the gradient noise scale as in [107]. Overall, GPT-3 increases model parameters to 175B showing that the performance of LLMs improves with the scale and is competitive with the fine-tuned models.

GPT-4 [108]. An autoregressive MLLM trained on public and private data that takes image and text as input and output textual response. After pre-training, the model is aligned with RLHF.

GPT-4o [109]. An autoregressive MLLM capable of processing audio, text, image, and videos and generating image, audio, and text. The model shares the same set of parameters across all the modalities resulting in enhanced generalization capabilities.

OpenAI-o3 [110]. A model trained for reasoning with internal CoT prompting. Training a model with internal reasoning improves safety over GPT-4 and produces better, more helpful responses than those trained without reasoning.

mT5 [11]. A multilingual T5 model [10] trained on the mC4 dataset with 101 languages. The dataset is extracted from the public Common Crawl scrape. The model uses a larger vocabulary size of 250,000 to cover multiple languages. To avoid overfitting or under-fitting for a language, mT5 employs a data sampling procedure to select samples from all languages. The article suggests

Table 1. Noteworthy Findings and Insights of *Pre-Trained* LLMs

Models	Findings and Insights
T5	—Encoder and decoder with shared parameters perform equivalently when parameters are not shared —Fine-tuning model layers (adapter layers) work better than the conventional way of training on only classification layers
GPT-3	—Few-shot performance of LLMs is better than the zero-shot, suggesting that LLMs are meta-learners
mT5	—Large multilingual models perform equivalently to single language models on downstream tasks. However, smaller multilingual models perform worse
PanGu-α	—LLMs have a good few-shot capabilities
CPM-2	—Prompt fine-tuning requires updating very few parameters while achieving performance comparable to full model fine-tuning —Prompt fine-tuning takes more time to converge as compared to full model fine-tuning —Inserting prompt tokens in between sentences can allow the model to understand relations between sentences and long sequences —In an analysis, CPM-2 finds that prompts work as a provider (additional context) and aggregator (aggregate information with the input text) for the model
ERNIE 3.0	—A modular LLM architecture with a universal representation module and task-specific representation module helps in the fine-tuning phase —Optimizing the parameters of a task-specific representation network during the fine-tuning phase is an efficient way to take advantage of the powerful pre-trained model
Jurassic-1	—The performance of LLM is highly related to the network size —To improve run-time performance, more operations can be performed in parallel (width) rather than sequential (depth) —To efficiently represent and fit more text in the same context length, the model uses a larger vocabulary to train a SentencePiece tokenizer without restricting it to word boundaries. This further benefits few-shot learning tasks
HyperCLOVA	—By employing prompt-based tuning, the performances of models can be improved, often surpassing those of state-of-the-art models when the backward gradients of inputs are accessible
Yuan 1.0	—The model architecture that excels in pre-training and fine-tuning cases may exhibit contrasting behavior in zero-shot and few-shot learning

(Continued)

Table 1. Continued

Models	Findings and Insights
Gopher	—Relative encodings enable the model to evaluate for longer sequences than training
ERNIE 3.0 Titan	—Additional self-supervised adversarial loss to distinguish between real and generated text improves the model performance as compared to ERNIE 3.0
GPT-NeoX-20B	—Parallel attention + FF layers speed up training 15% with the same performance as with cascaded layers —Initializing feed-forward output layers before residuals with scheme in [157] avoids activations from growing with increasing depth and width —Training on Pile outperforms GPT-3 on five-shot
OPT	—Restart training from an earlier checkpoint with a lower learning rate if loss diverges —Model is prone to generate repetitive text and stuck in a loop
Galactica	—Galactica's performance has continued to improve across validation set, in-domain, and out-of-domain benchmarks, even with multiple repetitions of the corpus, which is superior to existing research on LLMs —A working memory token approach can achieve strong performance over existing methods on mathematical MMLU and MATH benchmarks. It sets a new state-of-the-art on several downstream tasks such as PubMedQA (77.6%) and MedMCQA dev (52.9%)
GLaM	—The model capacity can be maintained at reduced computation by replacing the feed-forward layer in each transformer layer with a MoE —The model trained on filtered data shows consistently better performances on both NLG and NLU tasks, where the effect of filtering is more significant on the former tasks —Filtered pre-training corpora plays a crucial role in the generation capability of LLMs, especially for the downstream tasks —The scaling of GLaM MoE models can be achieved by increasing the size or number of experts in the MoE layer. Given a fixed budget of computation, more experts contribute to a better performance
LaMDA	—The model can be fine-tuned to learn to call different external information resources and tools
AlphaCode	—For higher effectiveness and efficiency, a transformer model can be asymmetrically constructed with a shallower encoder and a deeper decoder —To achieve better performances, it is necessary to employ strategies such as massively scaling upsampling, followed by the filtering and clustering of samples into a compact set —The utilization of novel sampling-efficient transformer architectures designed to facilitate large-scale sampling is crucial —Simplifying problem descriptions can effectively improve the model's performance

(Continued)

Table 1. Continued

Models	Findings and Insights
Chinchilla	—The model size and the number of training tokens should be scaled proportionately: for each doubling of the model size, the number of training tokens should be doubled as well
PaLM	—English-centric models produce better translations when translating to English as compared to non-English —Generalized models can have equivalent performance for language translation to specialized small models —Larger models have a higher percentage of training data memorization —Performance has not yet saturated even at 540B scale, which means larger models are likely to perform better
UL2	—Mode-switching training enables better performance on downstream tasks —CoT prompting outperforms standard prompting for UL2
AlexaTM	—Encoder-decoder architecture is more suitable to train LLMs given bidirectional attention to the context than decoder-only —CLM task can be added to benefit the model with efficient ICL —Placing layer norm at the beginning of each transformer layer improves the training stability
U-PaLM	—Training with a MoD outperforms PaLM when trained further for a few more FLOPs —Training with a MoD improves the infilling ability and open-ended text generation diversity
GLM-130B	—Pre-training data with a small proportion of multi-task instruction data improves the overall model performance
CodeGen	—Multi-step prompting for code synthesis leads to a better user intent understanding and code generation
LLaMA	—A constant performance improvement is observed when scaling the model —Smaller models can achieve good performances with more training data and computing time
PanGu-Σ	—Sparse models provide the benefits of large models at a lower computation cost —Randomly Routed Experts reduces catastrophic forgetting effects which in turn is essential for continual learning —Randomly Routed Experts allow extracting a domain-specific sub-model in deployment which is cost-efficient while maintaining a performance similar to the original

(Continued)

Table 1. Continued

Models	Findings and Insights
BloombergGPT	—Pre-training with general-purpose and task-specific data improves task performance without hurting other model capabilities
XuanYuan 2.0	—Combining pre-training and fine-tuning stages in single training avoids catastrophic forgetting
CodeT5+	—CLM is crucial for a model's generation capability in encoder-decoder architectures —Multiple training objectives like span corruption, CLM, matching, and so on complement each other for better performance
StarCoder	—HHH prompt by Anthropic allows the model to follow instructions without fine-tuning
LLaMA-2	—Model trained on unfiltered data is more toxic but may perform better on downstream tasks after fine-tuning —Model trained on unfiltered data requires fewer samples for safety alignment
PaLM-2	—Data quality is important to train better models —Model and data size should be scaled with 1:1 proportions —Smaller models trained for larger iterations outperform larger models
LLaMA-3/3.1	—Increasing batch size gradually stabilizes the training without loss spikes —High-quality data at the final stages of training improve the model performance —Increasing model context length windows stepwise allows it to better adapt to various sequence lengths
Nemotron-40B	—Model aligned iteratively on synthetic data with data generated from the previously aligned model achieves competitive performance
DeepSeek	—Batch size should increase with the increase in compute budget and decrease in learning rate
DeepSeek-v2	—MLA performs better than MHA while requiring a significantly smaller KV cache, therefore achieving faster data generation
DeepSeek-v3	—Multi-token prediction improves model performance

using a small amount of pre-training datasets, including all languages, when fine-tuning for a task using English language data. This allows the model to generate correct non-English outputs.

PanGu-α [111]. An autoregressive model that has a query layer at the end of standard transformer layers, example shown in Figure 8, to predict the next token. Its structure is similar to the transformer

Table 2. Key Insights and Findings from the Study of *Instruction-Tuned* LLMs

Models	Findings and Insights
T0	—Multi-task prompting enables zero-shot generalization and outperforms baselines —Even a single prompt per dataset task is enough to improve performance
WebGPT	—To aid the model in effectively filtering and utilizing relevant information, human labelers play a crucial role in answering questions regarding the usefulness of the retrieved documents —Interacting a fine-tuned language model with a text-based web-browsing environment can improve end-to-end retrieval and synthesis via imitation learning and RL —Generating answers with references can make labelers easily judge the factual accuracy of answers
Tk-INSTRUCT	—Instruction tuning leads to a stronger generalization of unseen tasks —More tasks improve generalization whereas only increasing task instances does not help —Supervised trained models are better than generalized models —Models pre-trained with instructions and examples perform well for different types of inputs
mT0 and BLOOMZ	—Instruction tuning enables zero-shot generalization to tasks never seen before —Multilingual training leads to even better zero-shot generalization for both English and non-English —Training on machine-translated prompts improves performance for held-out tasks with non-English prompts —English only fine-tuning on multilingual PLM is enough to generalize to other pre-trained language tasks
OPT-IML	—Creating a batch with multiple task examples is important for better performance —Only example proportional sampling is not enough, training datasets should also be proportional for better generalization/performance —Fully held-out and partially supervised tasks performance improves by scaling tasks or categories whereas fully supervised tasks have no effect —Including small amounts, i.e., 5% of pre-training data during fine-tuning is effective —Only 1% reasoning data improves the performance, adding more deteriorates performance —Adding dialogue data makes the performance worse
Sparrow	—Labelers' judgment and well-defined alignment rules help the model generate better responses —Good dialogue goals can be broken down into detailed natural language rules for the agent and the raters —The combination of RL with reranking yields optimal performance in terms of preference win rates and resilience against adversarial probing
Flan	—Fine-tuning with CoT improves performance on held-out tasks —Fine-tuning along with CoT data improves reasoning abilities —CoT tuning improves zero-shot reasoning —Performance improves with more tasks —Instruction fine-tuning improves usability which otherwise is challenging for pre-trained models —Improving the model's performance with instruction tuning is compute-efficient —Multi-task prompting enables zero-shot generalization abilities in LLM
WizardCoder	—Fine-tuning with re-written instruction-tuning data into a complex set improves performance
LLaMA-2-Chat	—Model learns to write safe responses with fine-tuning on safe demonstrations, while additional RLHF step further improves model safety and makes it less prone to jailbreak attacks
LIMA	—Less high-quality data are enough for fine-tuned model generalization

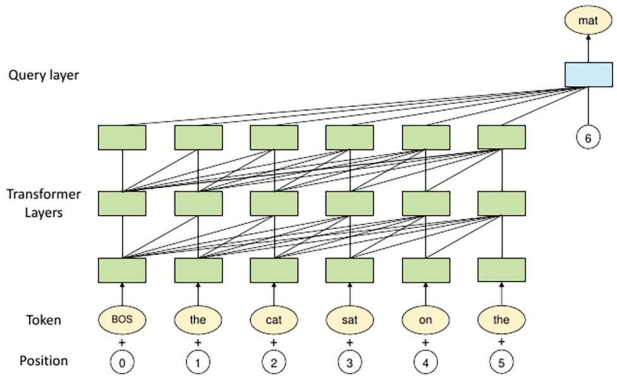

Fig. 8. The image is the article of [111], showing an example of PanGu-α architecture.

layer but with an additional embedding for the next position in the attention mechanism, given in Equation (3).

$$a = p_n W_h^q W_h^k T H_L^T \tag{3}$$

Cost-Efficient Pre-Trained Language Models (CPM-2) [12]. CPM-2 pre-trains bilingual (English and Chinese) 11B and 198B MoE models on the WuDaoCorpus [112] dataset. The tokenization process removes "_" white space tokens in the sentencepiece tokenizer. The models are trained with knowledge inheritance, starting with only the Chinese language in the first stage and then adding English and Chinese data. This trained model gets duplicated multiple times to initialize the 198B MoE model. Moreover, to use the model for downstream tasks, CPM-2 experimented with both complete fine-tuning and prompt fine-tuning as in [40] where only prompt-related parameters are updated by inserting prompts at various positions, front, middle, and back. CPM-2 also proposes the INFMOE, a memory-efficient framework with a strategy to dynamically offload parameters to the CPU for inference at a 100B scale. It overlaps data movement with inference computation for lower inference time.

ERNIE 3.0 [113]. ERNIE 3.0 takes inspiration from multi-task learning to build a modular architecture using Transformer-XL [114] as the backbone. The universal representation module is shared by all the tasks, which serve as the basic block for task-specific representation modules, which are all trained jointly for **natural language understanding (NLU)**, **natural language generation (NLG)**, and knowledge extraction. This LLM is primarily focused on the Chinese language. It claims to train on the largest Chinese text corpora for LLM training and achieved state-of-the-art in 54 Chinese NLP tasks.

Jurassic-1 [115]. A pair of autoregressive language models, including a 7B-parameter J1-Large model and a 178B-parameter J1-Jumbo model. The training vocabulary of Jurassic-1 comprise word pieces, complete words, and multi-word expressions without any word boundaries, where possible out-of-vocabulary instances are interpreted as Unicode bytes. Compared to the GPT-3 counterparts, the Jurassic-1 models apply a more balanced depth-to-width self-attention architecture [116] and an improved tokenizer for a faster prediction based on broader resources, achieving a comparable performance in zero-shot learning tasks and a superior performance in few-shot learning tasks given the ability to feed more examples as a prompt.

HyperCLOVA [117]. A Korean language model with GPT-3 architecture.

Yuan 1.0 [118]. Trained on a Chinese corpus with 5TB of high-quality text collected from the Internet. A Massive Data Filtering System built on Spark is developed to process the raw data via

coarse and fine filtering techniques. To speed up the training of Yuan 1.0 to save energy expenses and carbon emissions, various factors that improve the performance of distributed training are incorporated in architecture and training: like increasing the hidden state size improves pipeline and tensor parallelism performance, larger micro batches improve pipeline parallelism performance, and larger global batch size improve data parallelism performance. In practice, the Yuan 1.0 model performs well on text classification, Winograd Schema, **natural language inference (NLI)**, and **reading comprehension (RC)** tasks.

Gopher [119]. The Gopher family of models ranges from 44M to 280B parameters in size to study the effect of *scale* on the LLMs performance. The 280B model beats GPT-3 [6], Jurrasic-1 [115], MT-NLG [120], and others on 81% of the evaluated tasks.

ERNIE 3.0 TITAN [35]. ERNIE 3.0 Titan extends ERNIE 3.0 by training a larger model with 26× the number of parameters of the latter. This bigger model outperformed other state-of-the-art models in 68 NLP tasks. LLMs produce text with incorrect facts. To have control of the generated text with factual consistency, ERNIE 3.0 Titan adds another task, *Credible and Controllable Generations*, to its multi-task learning setup. It introduces additional self-supervised adversarial and controllable LM losses to the pre-training step, which enables ERNIE 3.0 Titan to beat other LLMs in their manually selected Factual QA task set evaluations.

GPT-NeoX-20B [121]. An autoregressive model that largely follows GPT-3 with a few deviations in architecture design, trained on the Pile dataset without any data deduplication. GPT-NeoX has parallel attention and feed-forward layers in a transformer block, given in Equation (4), that increases throughput by 15%. It uses rotary positional embedding [66], applying it to only 25% of embedding vector dimension as in [122]. This reduces the computation without performance degradation. As opposed to GPT-3, which uses dense and sparse layers, GPT-NeoX-20B uses only dense layers. The hyperparameter tuning at this scale is difficult; therefore, the model chooses hyperparameters from the method [6] and interpolates values between 13B and 175B models for the 20B model. The model training is distributed among GPUs using both tensor and pipeline parallelism.

$$x + Attn(LN_1(x)) + FF(LN_2(x)). \tag{4}$$

OPT [14]. It is a clone of GPT-3, developed to open-source a model that replicates GPT-3 performance. Training of OPT employs dynamic loss scaling [123] and restarts from an earlier checkpoint with a lower learning rate whenever loss divergence is observed. Overall, the performance of OPT-175B models is comparable to the GPT3-175B model.

BLOOM [13]. A causal decoder model trained on the ROOTS corpus to open-source an LLM. The architecture of BLOOM is shown in Figure 9, with differences like ALiBi positional embedding, an additional normalization layer after the embedding layer as suggested by the bitsandbytes[1] library. These changes stabilize training with improved downstream performance.

Generalist Language Model (GLaM) [91]. GLaM represents a family of language models using a sparsely activated decoder-only MoE structure [90, 124]. To gain more model capacity while reducing computation, the experts are sparsely activated where only the best two experts are used to process each input token. The largest GLaM model, GLaM (64B/64E), is about 7× larger than GPT-3 [6], while only part of the parameters are activated per input token. The largest GLaM (64B/64E) model achieves better overall results as compared to GPT-3 while consuming only one-third of GPT-3's training energy.

MT-NLG [120]. A 530B causal decoder based on the GPT-2 architecture that has roughly 3× GPT-3 model parameters. MT-NLG is trained on filtered high-quality data collected from various

[1] https://github.com/TimDettmers/bitsandbytes.

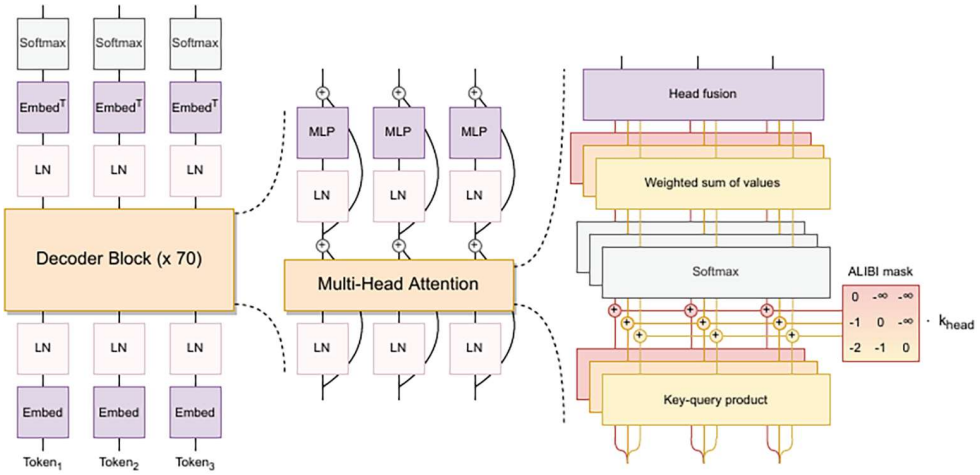

Fig. 9. The BLOOM architecture example sourced from [13].

public datasets and blends various types of datasets in a single batch, which beats GPT-3 on several evaluations.

Chinchilla [96]. A causal decoder trained on the same dataset as the Gopher [119] but with a little different data sampling distribution (sampled from MassiveText). The model architecture is similar to the one used for Gopher, with the exception of AdamW optimizer instead of Adam. Chinchilla identifies the relationship that model size should be doubled for every doubling of training tokens. Over 400 language models ranging from 70 million to over 16 billion parameters on 5–500 billion tokens are trained to get the estimates for compute-optimal training under a given budget. The authors train a 70B model with the same compute budget as Gopher (280B) but with four times more data. It outperforms Gopher [119], GPT-3 [6], and others on various downstream tasks, after fine-tuning.

AlexaTM [125]. An encoder-decoder model, where encoder weights and decoder embeddings are initialized with a pre-trained encoder to speed up training. The encoder stays frozen for the initial 100k steps and is later unfrozen for end-to-end training. The model is trained on a combination of denoising and **causal language modeling (CLM)** objectives, concatenating a [*CLM*] token at the beginning for mode switching. During training, the CLM task is applied for 20% of the time, which improves the ICL performance.

PaLM [15]. A causal decoder with parallel attention and feed-forward layers similar to Equation (4), speeding up training by a factor of 15. Additional changes to the conventional transformer model include SwiGLU activation, RoPE embeddings, multi-query attention that saves computation cost during decoding, and shared input-output embeddings. During training, loss spiking was observed, and to fix it, model training was restarted from a 100-step earlier checkpoint by skipping 200–500 batches around the spike. Moreover, the model was found to memorize around 2.4% of the training data at the 540B model scale, whereas this number was lower for smaller models.

PaLM-2 [126]. A smaller multilingual variant of PaLM, trained for larger iterations on a better quality dataset. PaLM-2 shows significant improvements over PaLM, while reducing training and inference costs due to its smaller size. To lessen toxicity and memorization, it appends special tokens with a fraction of pre-training data, which shows a reduction in generating harmful responses.

U-PaLM [127]. This method trains PaLM for 0.1% additional compute with the UL2 (also named as UL2Restore) objective [128], using the same dataset it outperforms the baseline significantly

on various NLP tasks, including zero-shot, few-shot, **commonsense reasoning (CR)**, CoT, and so on. Training with UL2R involves converting a causal decoder PaLM to a non-causal decoder PaLM and employing 50% sequential denoising, 25% regular denoising, and 25% extreme denoising loss functions.

UL2 [128]. An encoder-decoder architecture trained using a **mixture of denoisers (MoD)** objective. Denoisers include (1) R-Denoiser: a regular span masking, (2) S-Denoiser: which corrupts consecutive tokens of a large sequence, and (3) X-Denoiser: which corrupts a large number of tokens randomly. During pre-training, UL2 includes a denoiser token from R, S, X to represent a denoising setup. It helps improve fine-tuning performance for downstream tasks that bind the task to one of the upstream training modes. This MoD style of training outperforms the T5 model on many benchmarks.

GLM-130B [33]. GLM-130B is a bilingual (English and Chinese) model trained using an autoregressive mask infilling pre-training objective similar to the GLM [129]. This training style makes the model bidirectional as compared to GPT-3, which is unidirectional. As opposed to GLM, the training of GLM-130B includes a small amount of multi-task instruction pre-training data (5% of the total data) along with self-supervised mask infilling. To stabilize the training, it applies embedding layer gradient shrink.

LLaMA [21, 130]. A set of decoder-only language models varying from 7B to 70B parameters. LLaMA models series is the most famous among the community for parameter efficiency and instruction tuning.

LLaMA-1 [130]. Implements efficient causal attention [131] by not storing and computing masked attention weights and key/query scores. Another optimization is reducing the number of activations recomputed in the backward pass, as in [132].

LLaMA-2 [21]. This work is more focused on fine-tuning a safer and better LLaMA-2-Chat model for dialogue generation. The pre-trained model has 40% more training data with a larger context length and grouped-query attention.

LLaMA-3/3.1/3.2 [133]. A collection of models trained on a seven times larger dataset as compared to LLaMA-2 with double the context length, outperforming its previous variants and other models.

PanGu-Σ [92]. An autoregressive model with parameters copied from PanGu-α and extended to a trillion scale with **random routed experts (RRE)**, the architectural diagram is shown in Figure 10. RRE is similar to the MoE architecture, with distinctions at the second level, where tokens are randomly routed to experts in a domain instead of using a learnable gating method. The model has bottom layers densely activated and shared across all domains, whereas top layers are sparsely activated according to the domain. This training style allows for extracting task-specific models and reduces catastrophic forgetting effects in the case of continual learning.

Mixtral8x22b [134]. A MoE model with eight distinct experts routes each token to two experts at each layer and combines the outputs additively.

Snowflake Arctic [135]. Arctic LLM is a hybrid of dense and MoE architecture. The MoE (128×3.66B MLP experts) is parallel to the dense transformer (10B) with only two experts activated. The model has many experts, compared to other MoE LLMs [134, 136], to increase the model capacity and provide an opportunity to choose among many experts for a diverse configuration. The model has 480B parameters, and only 17B are active during a forward pass, reducing the computation significantly.

Grok [136, 137]. Grok is a family of LLMs including Grok-1 and Grok-1.5, released by XAI.

Grok-1 [136]. Grok-1 is a 314B parameters language MoE model (eight experts), where two experts are activated per token.

Grok-1.5 [137]. Grok-1.5 is a MLLM with a larger context length and improved performance.

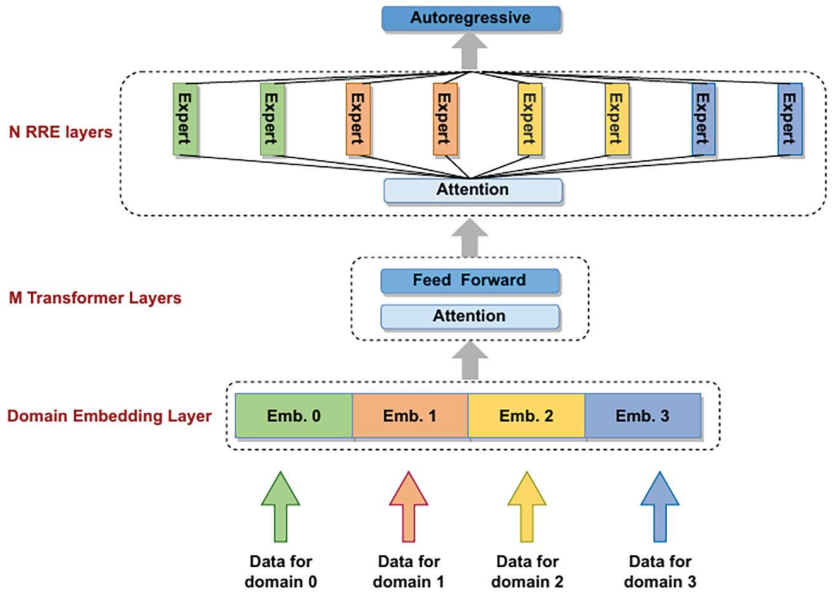

Fig. 10. This example illustrates the PanGu-∑ architecture, as depicted in the image sourced from [92].

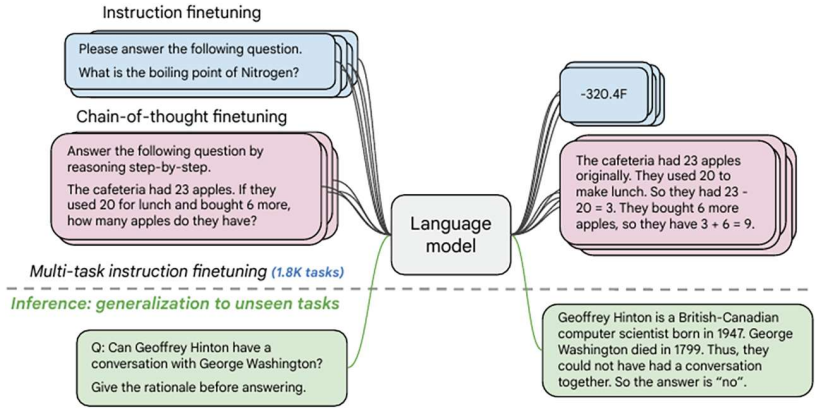

Fig. 11. An example image shows an instance of the Flan training paradigm, taken from [16].

Gemini [138, 139]. Gemini replaces Bard (based on PaLM) with multimodal capabilities and significant LM performance improvements.

Gemini-1 [138]. The first-ever autoregressive model to achieve human-level capabilities on the MMLU benchmark.

Gemini-1.5 [139]. A MLLM with MoE architecture builds on the findings of Gemini-1. The model has a 2M context window and can reason over information up to 10M tokens. Such large context windows were never achieved previously and shown to have a huge impact on performance gain.

Nemotron-4 340B [140]. A decoder-only model that has been aligned on 98% synthetic data and only 2% manually annotated data. Utilizing synthetic data at a large proportion improves the model performance significantly. The article suggested introducing alignment data with a smaller subset of previously seen data during the late stage of the model pre-training, enabling the smooth

transition from the pre-trained stage to the final training stage. To train better instruction-following models, weaker models are trained into stronger models iteratively. The synthetic data generated by the weaker instruction-tuned model are used to train a base model which is later **supervised fine-tuned (SFT)** outperforming the weaker model.

DeepSeek [141]. DeepSeek studies the LLMs scaling laws in detail to determine the optimal non-embedding model size and training data. The experiments were performed for eight budgets ranging from $1e^{17}$ to $3e^{20}$ training FLOPs. Each compute budget was tested against ten different models/data scales. The batch size and learning rates were also fitted for the given compute budget finding that the batch size should increase with the increased compute budget while decreasing the learning rate. Following are the equations for the optimal batch size (B), learning rate (η), model size (M), and data (D):

$$\begin{aligned} B_{opt} &= 0.2920.C^{0.3271} \\ \eta_{opt} &= 0.3118.C^{-0.1250} \\ M_{opt} &= M_{base}.C^{a} \\ D_{opt} &= D_{base}.C^{b} \\ M_{base} = 0.1715, D_{base} &= 5.8316, a = 0.5243, b = 0.4757 \end{aligned} \quad . \tag{5}$$

DeepSeek-v2 [142]. An MoE model that introduces **multi-head latent attention (MLA)** to reduce inference costs, by compressing KV cache into a latent vector. MLA achieves better performance than MHA, and other efficient attention mechanisms such as grouped-query attention, multi-query attention, and so on. Because of MLA, DeepSeek-v2 achieves 5.76 times faster inference throughput as compared to DeepSeek [141].

DeepSeek-v3 [143]. DeepSeek-v3 [143] has a similar architectural design introduced in DeepSeek-v2 [142] with a multi-token prediction objective. The model is trained in FP-8 mixed precision setting with complete computation-communication overlap across nodes.

3.1.2 *Coding.*

CodeGen [144]. CodeGen has a similar architecture to PaLM [15], i.e., parallel attention, MLP layers, and RoPE embeddings. The model is trained on both natural language and programming language data sequentially (trained on the first dataset, then the second, and so on) on the following datasets: (1) PILE, (2) BIGQUERY, and (3) BIGPYTHON. CodeGen proposed a multi-step approach to synthesizing code. The purpose is to simplify the generation of long sequences where the previous prompt and generated code are given as input with the next prompt to generate the next code sequence. CodeGen open-source a Multi-Turn Programming Benchmark to evaluate multi-step program synthesis.

Codex [145]. This LLM is trained on a subset of public Python GitHub repositories to generate code from docstrings. Computer programming is an iterative process where the programs are often debugged and updated before fulfilling the requirements. Similarly, Codex generates 100 versions of a program by repetitive sampling for a given description, which produces a working solution for 77.5% of the problems passing unit tests. Its powerful version powers GitHub Copilot.[2]

AlphaCode [146]. A set of LLMs, ranging from 300M to 41B parameters, designed for competition-level code generation tasks. It uses the multi-query attention [147] to reduce memory and cache costs. Since competitive programming problems highly require deep reasoning and an understanding of complex natural language algorithms, the AlphaCode models are pre-trained on filtered GitHub code in popular languages and then fine-tuned on a new competitive programming dataset named CodeContests. The CodeContests dataset mainly contains problems, solutions, and test cases

[2]https://github.com/features/copilot.

collected from the Codeforces platform.[3] The pre-training employs standard LM objectives, while GOLD [148] with tempering [149] serves as the training objective for the fine-tuning on CodeContests data. To evaluate the performance of AlphaCode, simulated programming competitions are hosted on the Codeforces platform: overall, AlphaCode ranks at the top 54.3% among over 5,000 competitors, where its Codeforces rating is within the top 28% of recently participated users.

CodeT5+ [34]. CodeT5+ is based on CodeT5 [150], with shallow encoder and deep decoder, trained in multiple stages initially unimodal data (code) and later bimodal data (text-code pairs). Each training stage has different training objectives and activates different model blocks encoder, decoder, or both according to the task. The unimodal pre-training includes span denoising and CLM objectives, whereas bimodal pre-training objectives contain contrastive learning, matching, and CLM for text-code pairs. CodeT5+ adds special tokens with the text to enable task modes, for example, [CLS] for contrastive loss, [$Match$] for text-code matching, and so on.

StarCoder [151]. A decoder-only model with the SantaCoder architecture, employing Flash attention to scale up the context length to 8k. The StarCoder trains an encoder to filter names, e-mails, and other personal data from the training data. Its fine-tuned variant outperforms PaLM, LLaMA, and LAMDA on HumanEval and MBPP benchmarks.

3.1.3 *Scientific Knowledge.*

Galactica [152]. A large curated corpus of human scientific knowledge with 48 million papers, textbooks, lecture notes, millions of compounds and proteins, scientific Web sites, encyclopedias, and more is trained using the metaseq library3, which is built on PyTorch and fairscale [153]. The model wraps reasoning datasets with the $< work >$ token to provide step-by-step reasoning context to the model, which has been shown to improve the performance on reasoning tasks.

3.1.4 *Dialog.*

LaMDA [154]. A decoder-only model pre-trained on public dialog data, public dialog utterances, and public web documents, where more than 90% of the pre-training data is in English. LaMDA is trained with the objective of producing responses that exhibit high levels of quality, safety, and groundedness. To achieve this, discriminative and generative fine-tuning techniques are incorporated to enhance the model's safety and quality aspects. As a result, the LaMDA models can be utilized as a general language model performing various tasks.

3.1.5 *Finance.*

BloombergGPT [155]. A non-causal decoder model trained using both financial ("FINPILE" from the Bloomberg archive) and general-purpose datasets. The model's architecture is similar to the BLOOM [13] and OPT [14]. It allocates 50B parameters to different blocks of the model using the approach [116]. For effective training, BloombergGPT packs documents together with $< |endoftext| >$ to use the maximum sequence length, uses warmup batch size starting from 1,024 to 2,048, and manually reduces the learning rate multiple times during the training.

Xuan Yuan 2.0 [156]. A Chinese financial chat model with BLOOM's [13] architecture trained on a combination of general purpose, financial, general purpose instructions, and financial institutions datasets. Xuan Yuan 2.0 combined the pre-training and fine-tuning stages to avoid catastrophic forgetting.

3.2 Fine-Tuned LLMs

Pre-trained LLMs have excellent generalization abilities to unseen tasks. However, because they are generally trained with the objective of next token prediction, LLMs have limited capacity to follow user intent and are prone to generate unethical, toxic, or inaccurate responses [20]. For

[3] https://codeforces.com/.

their effective utilization, LLMs are fine-tuned to follow instructions [16, 17, 97] and generate safe responses [20], which also results in increasing zero-shot, few-shot, and cross-task generalization [16, 18, 97], with minimal compute increment, e.g., 0.2% of the total pre-training for PaLM 540B [16].

We review various fine-tuned LLMs and strategies for effective fine-tuning in this section.

3.2.1 Instruction Tuning with Manually Created Datasets. Numerous hand-crafted instruction-tuning datasets with different design choices are proposed in the literature to instruction-tune LLMs. The performance of fine-tuned LLMs depends on multiple factors, such as dataset, instruction diversity, prompting templates, model size, and training objectives. Keeping this in view, diverse fine-tuned models have emerged in the literature using manually created datasets.

The models T0 [17] and mT0 (multilingual) [158] employ templates to convert existing datasets into prompt datasets. They have shown improvements in generalization to zero-shot and held-out tasks. Tk-Instruct [18] fine-tuned the T5 model with in-context instructions to study generalization on unseen tasks when given in-context instructions during test time. The model outperformed Instruct-GPT, despite being smaller in size, i.e., 11B parameters as compared to 175B of GPT-3.

Increasing Tasks and Prompt Setups. Zero-shot and few-shot performance improves significantly by expanding task collection and prompt styles. OPT-IML [97] and Flan [16], training paradigm shown in Figure 11, curated larger 2k and 1.8k task datasets, respectively. While increasing task size alone is not enough, OPT-IML and Flan add more prompting setups in their datasets, zero-shot, few-shot, and CoT. In continuation, CoT Collection [101] fine-tunes Flan-T5 further on 1.88M CoT samples. Another method [102] uses symbolic tasks with tasks in T0, Flan, and so on.

3.2.2 Instruction Tuning with LLMs Generated Datasets. Generating an instruction-tuning dataset requires carefully writing instructions and input-output pairs, which are often written by humans, smaller in size, and less diverse. To overcome this, self-instruct [19] proposed an approach to prompt available LLMs to generate instruction-tuning datasets. Self-instruct outperformed models trained on manually created dataset SUPER-NATURALINSTRUCTIONS (a dataset with 1,600+ tasks) [18] by 33%. It starts with a seed of 175 tasks, 1 instruction, and 1 sample per task and iteratively generates new instructions (52k) and instances (82k input-output pairs) using GPT-3 [6]. Contrary to this, Dynosaur [159] uses the meta-data of datasets on Huggingface to prompt LLMs to generate multiple task instruction-tuning datasets.

LLaMA Tuned. Various models in the literature instruction-tune LLaMA [160] with GPT-3 [6] or GPT-4 [108] generated datasets. Among these, Alpaca [161], Vicuna [162], and LLaMA-GPT-4 [163] are a few general-purpose fine-tuned models, where Alpaca is trained on 52k samples from text-davinci-003, Vicuna on 70k samples from ShareGPT.com, and LLaMA-GPT-4 by re-creating Alpaca instructions from GPT-4. Goat [164] fine-tunes LLaMA for arithmetic tasks (1 million samples) by generating data from ChatGPT and outperforms GPT-4, PaLM, BLOOM, OPT, and so on, attributing its success to the LLaMA's consistent tokenization of numbers. HuaTuo [165] is a medical knowledge model, fine-tuned with a generated QA dataset of 8k instructions.

Complex Instructions. Evol-Instruct [166, 167] prompts LLMs to convert given instructions into a more complex set. The instructions are iteratively evolved with re-writing instructions in complex wording and creating new instructions. With this style of automated instruction generation, WizardLM [166] (fine-tuned LLaMA on 250k instructions) outperforms Vicuna and Alpaca, and WizardCoder [167] (fine-tuned StarCoder) beats Claude-Plus, Bard, and others.

3.2.3 Aligning with Human Preferences. Incorporating human preferences into LLMs presents a significant advantage in mitigating undesirable behaviors and ensuring accurate outputs. The initial work on alignment, such as Instruct-GPT [20], aligns GPT-3 using a three-step approach, instruction tuning, RM, and fine-tuning with RL. The SFT GPT-3 on demonstrations is queried to generate responses, which human labelers rank according to human values, and a reward model is trained

on the ranked data. Lastly, the GPT-3 is trained with PPO using rewards on the generated data from the reward model. LLaMA 2-Chat [21] improves alignment by dividing RM into helpfulness and safety rewards and using rejection sampling in addition to PPO. The initial four versions of LLaMA 2-Chat are fine-tuned with rejection sampling and then with PPO on top of rejection sampling.

Aligning with Supported Evidence. This style of alignment allows the model to generate responses with proofs and facts, reduces hallucination, and assists humans more effectively, which increases trust in the model's output. Similar to the RLHF training style, a reward model is trained to rank generated responses containing web citations in answers to questions, which is later used to train the model, as in GopherCite [168], WebGPT [169], and Sparrow [170]. The ranking model in Sparrow [170] is divided into two branches, preference reward and rule reward, where human annotators adversarial probe the model to break a rule. These two rewards together rank a response to train with RL.

Aligning Directly with SFT. The PPO in the RLHF pipeline is complex, memory-intensive, and unstable, requiring multiple models, reward, value, policy, and reference models. Avoiding this sophisticated alignment pipeline is possible by incorporating minimal changes in the SFT pipeline as in [171–173], with better or comparable performance to PPO. Direct preference optimization [171] trains a model directly on the human-preferred responses to maximize the likelihood of preferred against unpreferred responses, with per-sample importance weight. Reward ranked fine-tuning RAFT [172] fine-tunes the model on ranked responses by the reward model. Preference ranking optimization [174] and RRHF [173] penalize the model to rank responses with human preferences and supervised loss. On the other hand, chain-of-hindsight [175] provides feedback to the model in language rather than reward, to learn good vs. bad responses.

Aligning with Synthetic Feedback. Aligning LLMs with human feedback is slow and costly. The literature suggests a semi-automated process to align LLMs by prompting LLMs to generate helpful, honest, and ethical responses to the queries, and fine-tuning using the newly created dataset. Constitutional AI [176] replaces human feedback in RLHF with AI, calling it RL from AI feedback. AlpacaFarm [177] designs prompts to imitate human feedback using LLMs APIs. Opposite to constitutional AI, AlpacaFarm injects noise in feedback to replicate human mistakes. Self-Align [98] prompts the LLM with ICL examples, instructing the LLM about what the response should contain to be considered useful and ethical. The same LLM is later fine-tuned with the new dataset.

Aligning with Prompts. LLMs can be steered with prompts to generate desirable responses without training [178, 179]. The self-correction prompting in [179] concatenates instructions and CoT with questions, guiding the model to answer its instruction following a strategy to ensure moral safety before the actual answer. This strategy is shown to reduce the harm in generated responses significantly.

Red-Teaming/Jailbreaking/Adversarial Attacks. LLMs exhibit harmful behaviors, hallucinations, leaking personal information, and other shortcomings through adversarial probing. The models are susceptible to generating harmful responses even though they are aligned for safety [180, 181]. Red-teaming is a common approach to address illicit outputs, where the LLMs are prompted to generate harmful outputs [181, 182]. The dataset collected through red-teaming is used to fine-tune models for safety. While red-teaming largely relies on human annotators, another work [183] red-team LLMs to find prompts that lead to harmful outputs for other LLMs.

3.2.4 Continue Pre-Training. Although fine-tuning boosts a model's performance, it leads to catastrophic forgetting of previously learned information. Concatenating fine-tuning data with a few randomly selected pre-training samples in every iteration avoids network forgetting [156, 184]. This is also effective in adapting LLMs for cases where fine-tuning data is small and the original capacity is to be maintained. Prompt-based continued pre-training [185] trains the model with text and instructions related to tasks and then finally instruction-tunes the model for downstream tasks.

3.2.5 Sample Efficiency. While fine-tuning data is generally many-fold smaller than the pre-training data, it still has to be large enough for acceptable performance [16, 18, 97] and requires proportional computing resources. Studying the effects on performance with less data, existing literature [186, 187] finds that models trained on less data can outperform models trained with more data. In [186], 25% of the total downstream data is found enough for state-of-the-art performance. Selecting coreset-based 0.5% of the total instruction-tuning data improves the model performance by 2% in [187], as compared to the complete data tuning. **Less is more for alignment (LIMA)** [188] uses only 1,000 carefully created demonstrations to fine-tune the model and has achieved comparable performance to GPT-4.

3.3 Increasing Context Window

LLMs are trained with limited context windows due to expensive attention and high memory requirements. A model trained on limited sequence lengths fails to generalize to unseen lengths at inference time [49, 189]. Alternatively, LLMs with ALiBi [65] positional encodings can perform zero-shot length extrapolation. However, ALiBi has less expressive power [66] and inferior performance on multiple benchmarks [46], and many LLMs use RoPE positional embedding that is unable to perform zero-shot extrapolation. A larger context length has benefits such as a better understanding of longer documents, more samples in ICL, execution of bigger reasoning processes, and so on. Expanding context length during fine-tuning is slow, inefficient, and computationally expensive [49]. Therefore, researchers employ various context window extrapolation techniques discussed below.

Position Interpolation. Rather than extrapolating, [49] shows that interpolating position encodings within the pre-trained context window are more effective. The work demonstrates that only 1,000 steps of fine-tuning are enough to achieve better results on larger windows without reducing performance compared to the original context size. Giraffe [46] uses power scaling in RoPE, and YaRN [47] proposed NTK-aware interpolation.

Efficient Attention Mechanism. Dense global attention is one of the major constraints in training larger context window LLMs. Using efficient attention variants, such as local, sparse, and dilated attention, reduces the computation cost significantly. LongT5 [48] proposes **transient global (TGlobal)** attention, applying attention to local and global tokens (windowed token averaging). The model replaces attention in T5 [10] with TGlobal attention, pre-trains the model on 4,098 sequence length, fine-tunes on larger window sizes, as large as 16k, and improves task performance on longer inputs. This shows the extrapolation ability of TGlobal attention with only fine-tuning. COLT5 [190] uses two branches, one with lightweight and the other with heavyweight attention and feed-forward layers. All tokens are processed from the lightweight branch, and only important tokens are routed to the heavyweight branch. LongNet [191] replaces standard attention with dilated attention, expanding sequence length to 1 billion tokens. LongLoRA [192] proposes shift-short attention, used during fine-tuning to reduce dense attention costs. However, the model during inference uses dense attention and achieves similar performance as full attention fine-tuning.

Extrapolation without Training. LM-Infinite [189] and **parallel context windows (PCW)** [193] show length extrapolation is possible using pre-trained LLMs. LM-Infinite suggested Λ-shaped attention applied within the original context window limits. Likewise, PCW chunks larger inputs into the pre-trained context lengths and applies the same positional encodings to each chunk.

3.4 Augmented LLMs

LLMs are capable of learning from the examples concatenated with the input, known as context augmentation, ICL, or few-shot prompting. They show excellent generalization to unseen tasks with few-shot prompting, enabling LLMs to answer queries beyond the capacity acquired during training [6, 55]. These emergent abilities allow for adapting the model without fine-tuning—a

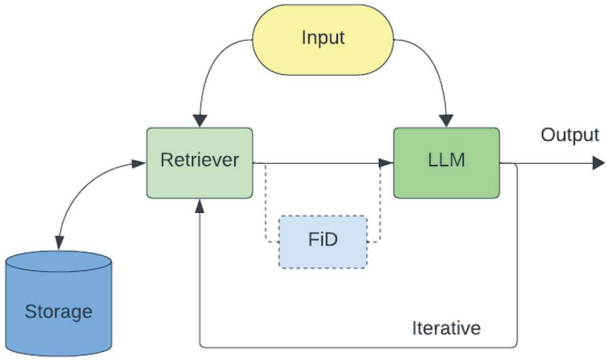

Fig. 12. A flow diagram of retrieval-augmented LLMs. The retriever extracts a similar context to the input and forwards it to the LLM either in simple language or encoded through Fusion-in-Decoder (FiD). Depending on the task, retrieval and generation may repeat multiple times.

costly process. Aside from this, hallucination, producing inaccurate, unsafe, or factually incorrect responses, is common for LLMs, which is avoided by augmenting contextual data. While the user can provide in-context samples in the query [32, 54], here we specifically refer to the methods that access external storage programmatically, calling them augmented LLMs.

The literature suggests various external memory designs to augment LLMs, long-term [194–197], short-term [198], symbolic [199], and non-symbolic [200, 201]. The memory can be maintained in different formats such as documents, vectors, or databases. A few systems maintain intermediate memory representations to retain information across multiple iterations [195, 197], while others extract important information from the datasets and save it in memory for recall [202]. The memory read and write operations are performed either with or without LLMs cooperation [195, 197, 203, 204], acting as a feedback signal in [198]. We discuss different types of augmented LLMs below.

3.4.1 Retrieval-Augmented LLMs. LLMs may have limited memory and outdated information, leading to inaccurate responses. Retrieving relevant information from external up-to-date storage enables the LLMs to accurately answer with references and utilize more information. With retrieval augmentation, smaller models have been shown to perform at par with larger models. For instance, the 11B model can become competitive to 540B PaLM in [25] and 7.5B to 280B Gopher in [196]. **Retrieval-augmented language modeling (RALM)** has two major components, shown in Figure 12, namely: (1) retriever and (2) language model. In RALM, the retriever plays a crucial role in driving LLM response, where incorrect information can steer LLMs to false behavior. This leads to the development of various methods to retrieve accurate information and fuse with the query for better performance.

Zero-Shot Retrieval Augmentation. This kind of augmentation keeps the original LLM architecture and weights unchanged and uses BM25 [205], nearest neighbors, or frozen pre-trained models like Bert [7] as a retriever. The retrieved information is provided as input to the model for response generation, shown to improve performance over LLMs without retrieval [201, 206]. In some scenarios, multiple retrieval iterations are required to complete the task. The output generated in the first iteration is forwarded to the retriever to fetch similar documents. Forward-looking active retrieval [200] initially generates the response and corrects the output by retrieving relevant documents if the response contains low-confidence tokens. Similarly, RepoCoder [207] fetches code snippets recursively for code completion.

Training with Retrieval Augmentation. To reduce failures in **retrieval augmentation generation (RAG)**, researchers train or fine-tune retrievers and LLMs with a retrieval augmentation pipeline. We discuss the literature below based on their focus on the respective training processes of the pipeline.

Training LLM. **Retrieval-enhanced transformer (RETRO)** [196] shows pre-training smaller LLMs with RAG pipeline outperforms larger LLMs, such as GPT-3 trained without RAG. RETRO uses a 2-trillion token subset of MassiveText as a database. The retrieval pipeline divides the input query into subsets and retrieves relevant chunks from the database for each subset, encoded together with input intermediate representations for generating tokens. It uses cross-chunked attention to attend to previous chunks autoregressively. A study on RETRO [208] shows models pre-trained without RAG but fine-tuned using RAG lack the performance gains obtained by pre-training with RAG.

Training Retriever. Quality of responses generated by LLMs is highly dependent on the in-context examples. Therefore, [209–212] train retrievers to retrieve accurate few-shot samples while keeping the LLM frozen for generation. Retrieved samples are ranked to build ground-truth data to train retrievers with contrastive learning in [209, 211]. RoBERTa is trained for downstream tasks in [210] for ICL samples retrieval. REPLUG [212] trains the retriever with supervised signals from the frozen LLM-generated outputs.

Training Retriever and LLM. Further benefits are achieved by training both the retriever and the model in [25, 213, 214]. In this case, the error propagates back to the retriever, updating both the language model and the retriever. While MLM is a common pre-training objective [25, 214], retrieval pre-trained transformer [213] used document chunk prediction as a pre-training objective for long text modeling.

Encoded Context Augmentation. Concatenating retrieved documents with the query becomes infeasible as the sequence length and sample size grow. Encoding the context and fusing it with the decoder (Fusion-in-Decoder) using cross-attention makes it possible to augment more samples without increasing computation costs significantly [25, 196, 213, 215].

Web Augmented. Locally stored memory, but external to LLM, has limited information. However, a large amount of information is available on the Internet, which gets updated regularly. Rather than storing information locally, various methods retrieve query-related context through a web search and forward it to LLMs [169, 216, 217].

3.4.2 Tool-Augmented LLMs. While RAG relies on the retriever to provide context to the LLM to answer queries, tool-augmented LLMs capitalize on the reasoning abilities of LLMs to iteratively plan by dividing tasks into sub-tasks, selecting necessary tools, and taking actions to complete the task [27, 218–220]. A generic pipeline of tool-augmented LLMs is shown in Figure 13, where different modules in Figure 13 are selected in a loop until the task completion.

Zero-Shot Tool Augmentation. LLMs ICL and reasoning abilities enable them to interact with tools without training. Automatic reasoning and tool use [220] builds a task library with demonstrations of reasoning steps and calling external tools. It retrieves similar task examples and provides the context to the LLM for inference. Aside from this, [221] shows tool documentation is enough to teach LLMs to use tools without demonstrations. RestGPT [222] integrates LLMs with RESTful APIs by decomposing tasks into planning and API selection steps. The API selector understands the API documentation to select a suitable API for the task and plan the execution. ToolkenGPT [223] uses tools as tokens by concatenating tool embeddings with other token embeddings. During inference, the LLM generates the tool tokens representing the tool call, stops text generation, and restarts using the tool execution output.

Training with Tool Augmentation. LLMs are trained to interact with diverse tools, enhancing planning abilities to overcome the limitations of zero-shot tool augmentation [27, 224–226]. Gorilla

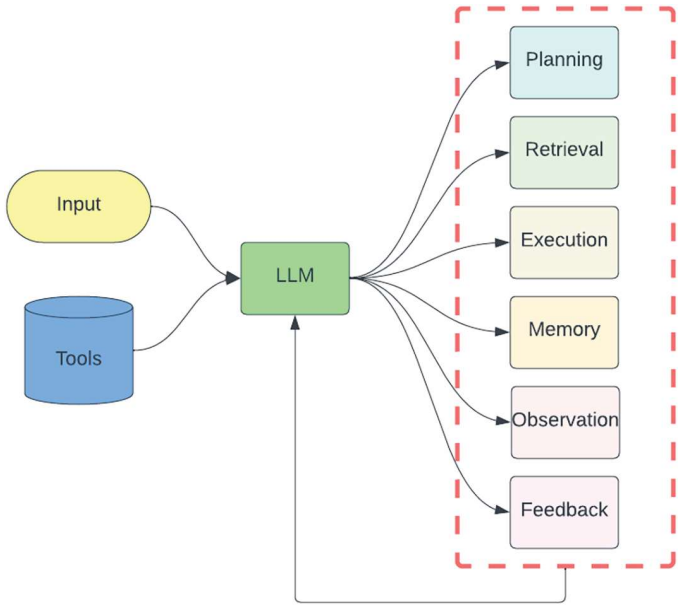

Fig. 13. A basic flow diagram of tool-augmented LLMs. Given an input and a set of available tools, the model generates a plan to complete the task. The tool-augmented LLMs utilize different modules iteratively, such as retriever, tool execution, read-write to memory, feedback, and so on, depending on the task.

[224] instruction-tunes LLaMA with IR from API documentation. It uses the self-instruct [19] data generation pipeline with GPT-4 by providing in-context examples retrieved from API documentation. Tool-augmented language model [27] fine-tunes T5 [10] for tool use with a self-play approach, where it iteratively completes tool manipulation tasks and includes them back in the training set. ToolLLM [226] collects 16k APIs from RapidAPI. It samples APIs from the list to generate an instruction-tuning dataset using ChatGPT in single-tool and multi-tool scenarios. For high-quality datasets, ToolLLM suggested a depth-first search-based decision tree method to generate ground-truths with diverse reasoning and planning.

Multimodal Tool Augmentation. The compositional reasoning capacity of LLMs allows them to manipulate tools in multimodal settings [218, 219, 227]. Following the pipeline shown in Figure 13, the LLM outlines a plan, generally executing in a sequence: Plan $\rightarrow$ Tool selection $\rightarrow$ Execute $\rightarrow$ Inspect $\rightarrow$ Generate, to respond to the user query. Here, the database of tools is rich in modalities, including text, images, and so on. Many of the multimodal tool augmentation systems employ MLLM s [31, 219, 227, 228], while others utilize single modality LLMs and generate a plan on using different modality tools to solve multimodal queries [229].

3.5 LLMs-Powered Agents

AI agents are autonomous entities, capable of planning, decision-making, and performing actions to achieve complex goals. In the early days, AI agents were rule-based, designed for narrow tasks, and had limited capabilities, such as Clippy [230] and Deep Blue [231]. In contrast to this, LLMs abilities to respond to dynamic scenarios have made it possible to incorporate them in diverse applications, including LLMs-powered agents [219, 227], where LLMs behave as the brain of agents. LLMs have been incorporated in web agents [169, 170], coding agents [232], tool agents [27, 226], embodied agents [26], and conversational agents [198], requiring minimal to no fine-tuning. Below,

we summarize the research in LLM-based autonomous agents. For a more detailed discussion, please refer to [233, 234].

LLMs Steering Autonomous Agents. LLMs are the cognitive controllers of the autonomous agents. They generate plans, reason about tasks, incorporate memory to complete tasks, and adapt the outline depending on the feedback from the environment. Depending on the acquired capabilities of LLMs, many methods fine-tune, propose a better prompting approach, or utilize different modules to enhance agents' performance. Modules and strategies employed in autonomous agents are briefly discussed below.

Planning and Reasoning. Completing a complex task requires human-like logical thinking, planning necessary steps, and reasoning current and future directions. Prompting methods like CoTs [103], ToTs [105], and self-consistency [104] are central to agents, eliciting LLMs to reason its actions and choose among different paths for task completion. When LLMs are prompted with a task description and a sequence of actions, they can accurately generate plan actions without any fine-tuning [235]. Reasoning via planning [236] incorporates a re-purposed LLM as a world model to reason about future outcomes and explore alternative paths for task completion. Retroformer [237] uses a retrospective LLM to improve main LLM planning and reasoning capabilities by providing helpful task cues.

Feedback. LLMs in open-loop systems generate plans and assume that the agent will complete them successfully. However, the actual scenario is different with failures and variable responses from the environment. To correctly complete tasks, many methods use LLMs in a closed-loop where the action response is provided as feedback to the LLMs to re-assess and update the plan as required [198, 238–240]. Another direction of research exploits LLMs as reward functions to train RL policies instead of humans [241].

Memory. LLMs can learn from the context provided in the prompt. In addition to internal memory, various systems employ external memory to save the response history. Reflexion [198] maintains an episodic memory to use previous responses as feedback to improve future decision-making. Retroformer [237] improves its responses by employing short-term and long-term memory, where short-term memory contains recent responses and long-term memory keeps summarized failed attempts to add in the prompt as reflection.

Multi-Agents Systems. LLMs can play user-defined roles and behave like a specific domain expert. In multi-agent systems, each LLM is assigned a unique role, simulating human behavior and collaborating with other agents to complete a complex task [232, 242].

LLMs in Physical Environment. LLMs are good at instruction-following, however, utilizing them for physically grounded tasks requires adaptation, as they lack real-world knowledge. This could lead to generating illogical responses for a particular physical situation [26, 243]. SayCan [243] make LLMs aware of the available low-level task operations. LLM (Say) builds a high-level plan to complete the task and a learned affordance function (Can) explores the possibility of executing the plan in the real world. SayCan uses RL to train the language-conditioned affordance function. PaLM-E enables the LLM to solve grounded tasks by training MLLM feeding inputs directly from the sensors.

Manipulation. In the area of manipulation [239, 244], LLMs enhance a robot's dexterity and adaptability, excelling in tasks like object recognition, grasping, and collaboration. They analyze visual and spatial information to determine the most effective approach to interact with objects.

Navigation. LLMs enhance a robot's ability to navigate complex environments with precision and adaptability [245–248]. They generate feasible paths and trajectories for robots, accounting for intricate environmental details [249]. This ability is valuable in scenarios requiring precise and dynamically adaptable navigation in environments like warehouses, transport, healthcare facilities, and residences.

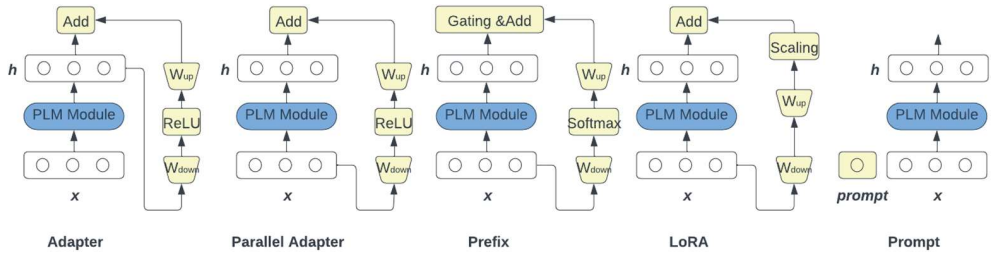

Fig. 14. Illustration of parameter-efficient fine-tuning paradigms, where x is input and h is hidden state, figure courtesy [38]. Parallel adapter and **low-rank adaptation (LoRA)** fall in the adapter tuning category.

3.6 Efficient LLMs

Deploying LLMs in production is expensive. Reducing their running costs while preserving performance is an appealing area of research. This section summarizes the approaches suggested to enhance LLMs' efficiency.

3.6.1 Parameter Efficient Fine-Tuning. Fine-tuning LLMs with tens or hundreds of billions of parameters, such as GPT-3 (175B), BLOOM (176B), MT-NLG (540B), and so on, is computationally intensive and time-consuming. To avoid complete model fine-tuning, numerous **parameter-efficient fine-tuning (PEFT)** techniques [38–41, 250] try to achieve acceptable model fine-tuning performance at reduced costs. As compared to full fine-tuning [251], PEFT performs better in low-resource setups, achieves comparable performance on medium-resource scenarios, and performs worse than full fine-tuning under high-resource availability. An overview of different PEFT approaches is shown in Figure 14.

Adapter Tuning. Adds a few trainable parameters within the transformer block. The adapter layer is a sequence of feature downscaling, non-linearity, and upscaling [106]. Variants of adapter tuning inject adapter layers sequentially [106] and in parallel [38], whereas the mixture of adapter (AdaMix) [252] employs multiple adapter modules in a single layer. AdaMix routes input instances randomly to one of the multiple downscale and upscale modules. The mixture of adapters is averaged out for inference to avoid additional latency. LoRA [253] learns low-rank decomposed matrices to freeze original weights. The learned weights are fused with the original weights for inference, avoiding latency.

Prompt Tuning. Prompting is an effective way to adapt a pre-trained LLM for the downstream task. However, manual prompts bring uncertainty in the model's prediction, where a change in a single word drops the performance [250]. Prompt tuning alleviates this problem by fine-tuning only 0.001–3% additional parameters [254]. It concatenates trainable prompt parameters with the model embeddings [40, 250, 254]. Task-specific fixed discrete prompts are concatenated with input embeddings in [40]. As discrete prompts bring instability, prompts are encoded through a learnable mapping in P-Tuning [250], naming continuous prompts, which are appended with the discrete prompts. Only the prompt encoder is trainable in the model. In an extension of P-Tuning, continuous prompts are concatenated with each layer of the network in [254]. Progressive prompts [255] avoid catastrophic forgetting and transfer previously learned knowledge by sequentially adding trainable prompt embeddings to the previously frozen task embeddings.

Prefix Tuning. A set of trainable task-specific prefix vectors are appended to the frozen transformer layers in prefix tuning [41]. The prefix vectors are virtual tokens attended by the context tokens on the right. In addition, adaptive prefix tuning [256] applies a gating mechanism to control the information from the prefix and actual tokens.

Bias Tuning. Fine-tuning only bias terms in small to medium training data has been found effective in BitFit [257]. This method achieves full fine-tuning performance for tasks with less training data and comparable performance with more training data.

3.6.2 Quantization. LLMs require extensive computing and memory for inference. Deploying a 175B parameter GPT-3 model needs at least five 80GB A100 GPUs and 350GB of memory to store in FP16 format [44]. Such demanding requirements for deploying LLMs make it harder for smaller organizations to utilize them. Model compression is an effective solution but comes at the cost of degraded performance, especially at large scales greater than 6B. These models exhibit very large magnitude outliers that do not exist in smaller models [258], making it challenging and requiring specialized methods for quantizing LLMs [44, 259].

Post-Training Quantization. Minimal or no training is required in this type of quantization, without significantly compromising the model performance. LLM-8-bit [258] uses full-precision matrix multiplication for weights associated with outlier features and 8-bit for remaining features. The lower precision multiplication outputs are converted to FP-16 and concatenated with others. The quantized models have homogeneous word embeddings, which may degrade their performance. To fix this, token-level knowledge distillation is employed in [45] along with independent quantization scaling factors for each module due to varying weight distribution. Feature distributions are asymmetric and appear in different channels; outlier suppression [260] shifts and scales per-channel activation distributions for effective quantization. SmoothQuant [44] quantizes activations and weights to INT8 format by smoothing activations and migrating the quantization difficulty toward weights. It multiplies the inverse of the smoothing factor with weights, which introduces a few outliers in the weights but is easier to quantify than unsmoothed activations. OPTQ [259] uses the optimal brain compression [261] algorithm to quantize the model layer-by-layer and update weights to compensate for quantization error. To improve speed and performance, OPTQ updates weights in arbitrary order, employs lazy updates, and uses better Cholesky kernels. Outlier-aware weight quantization [262] uses the OPTQ algorithm for quantization but assigns higher precision to vulnerable weights, causing outliers and lower precision for others.

Quantization-Aware Training. To compensate for performance degradation, a quantized model is fine-tuned in **quantization-aware training (QAT)** [263–265]. Alpha Tuning quantizes the model using binary coding quantization [266] and fine-tunes only quantization scaling factors. This approach improves performance over PEFT of the pre-trained model. Similarly, parameter-efficient and quantization-aware adaptation [267] reduces the precision of fully connected layers and fine-tunes only quantization scaling parameters. LLM-QAT [265] generates training data from the pre-trained network and trains a quantized student model with knowledge distillation. QLoRA [264] fine-tunes 4-bit quantized pre-trained LLM with LoRA [253] using a 4-bit normal float, which shows better performance over a 4-bit integer and float.

3.6.3 Pruning. Pruning is an alternative approach to quantization to compress model size, thereby reducing LLMs deployment costs significantly. Compared to task-agnostic pruning, task-specific pruning is easily achievable with good performance, where a model is fine-tuned on the downstream task and pruned for faster inference. It is possible to prune LLMs for individual tasks, but the cost of pruning and deploying task-specific models is high. To overcome this, many structured and unstructured pruning methods for LLMs have been proposed to maintain reasonable performance across all tasks while shrinking the model size [42, 268, 269].

Unstructured Pruning. This kind of pruning removes less important weights without maintaining any structure. Existing LLM pruning methods take advantage of the unique characteristics of LLMs, uncommon for smaller models, where a small subset of hidden states are activated with large magnitude [258]. Pruning by weights and activations (Wanda) [268] prunes weights in every row

based on importance, calculated by multiplying the weights with the norm of input. The pruned model does not require fine-tuning, thereby saving computational costs. Outlier weighed layerwise sparsity [270] extends Wanda with non-uniform layer pruning. It shows that the number of outliers varies for different layers; therefore, the model should have variable pruning ratios for better performance for every layer. Contrastive pruning [43] iteratively prunes the model by training the sparse model using contrastive loss between pre-trained, fine-tuned, and snapshots of previous sparse models to learn task-specific and task-agnostic knowledge.

Structured Pruning. Here, the parameters are removed in groups, rows, columns, or matrices, which speeds up the inference because of effective hardware tensor core utilization [268]. LLM-Pruner [42] employs a three-stage structured pruning strategy, identifying the groups of hidden states causing each other to activate during the forward pass, keeping important groups and removing less important ones, and fine-tuning the pruned model with LoRA. Sparsity-induced mask learning [271] prunes the network using learnable masks. Similarly, another method prunes LLMs by learning masks and removing unimportant rank-1 components of the factorized weight matrix [269].

3.7 MLLMs

Inspired by the success of LLMs in NLP applications, an increasing number of research works are now facilitating LLMs to perceive different modalities of information like image [272–274], video [275–277], audio [277–279], and so on. MLLMs present substantial benefits compared to standard LLMs that process only text. By incorporating information from various modalities, MLLMs can achieve a deeper understanding of context, leading to more intelligent responses infused with a variety of expressions. Importantly, MLLMs align closely with human perceptual experiences, leveraging the synergistic nature of our multisensory inputs to form a comprehensive understanding of the world [26, 279]. Coupled with a user-friendly interface, MLLMs can offer intuitive, flexible, and adaptable interactions, allowing users to engage with intelligent assistants through a spectrum of input methods. According to the ways of constructing models, current MLLMs can be generally divided into three streams: pre-training, fine-tuning, and prompting. In this section, we will discuss more details of these main streams, as well as the important application of MLLMs in visual reasoning.

Pre-Training. This stream of MLLMs intends to support different modalities using unified end-to-end models. For instance, Flamingo [272] applies gated cross-attention to fuse vision and language modalities, which are collected from pre-trained and frozen visual encoder and LLM, respectively. Moreover, BLIP-2 [273] proposes a two-stage strategy to pre-train a Querying Transformer (Q-Former) for the alignment between vision and language modalities: in the first stage, vision-language representation learning is bootstrapped from a frozen visual encoder; and in the second stage, a frozen LLM bootstraps vision-to-language generative learning for zero-shot image-to-text generation. Similarly, MiniGPT-4 [280] deploys pre-trained and frozen ViT [281], Q-Former and Vicuna LLM [162], only training the linear projection layer for vision and language modalities alignment.

Fine-Tuning. Derived from instruction tuning [16] for NLP tasks [16, 20, 97], researchers are fine-tune pre-trained LLMs using multimodal instructions. Following this method, LLMs can be easily and effectively extended as multimodal chatbots [29, 274, 280] and multimodal task solvers [30, 282, 283]. The key issue of this stream of MLLMs is to collect multimodal instruction-following data for fine-tuning [58]. To address this issue, the solutions of benchmark adaptation [282, 284, 285], self-instruction [19, 31, 286], and hybrid composition [283, 287] are employed, respectively. To mitigate the gap between the original language modality and additional modalities, the learnable interface is introduced to connect different modalities from frozen pre-trained models. Particularly,

the learnable interface is expected to work in a parameter-efficient tuning manner: e.g., LLaMA-Adapter [288] applies an efficient transformer-based adapter module for training, and LaVIN [287] dynamically learns the multimodal feature weights using a mixture-of-modality adapter. Different from the learnable interface, the expert models can directly convert multimodalities into language: e.g., VideoChat-Text [275] incorporates Whisper [289], a speech recognition expert model, to generate the captions of given videos for the understanding of following LLMs.

Prompting. Different from the fine-tuning technique that directly updates the model parameters given task-specific datasets, the prompting technique provides certain context, examples, or instructions to the model, fulfilling specialized tasks without changing the model parameters. Since prompting can significantly reduce the need for large-scale multimodal data, this technique is widely used to construct MLLMs. Particularly, to solve multimodal CoT problems [103], LLMs are prompted to generate both the reasoning process and the answer given multimodal inputs [290]. On this front, different learning paradigms are exploited in practice: for example, Multimodal CoT [290] involves two stages of rationale generation and answer inference, where the input of the second stage is a combination of the original input and the output of the first stage; and CoT-PT [291] applies both prompt tuning and specific visual bias to generate a chain of reasoning implicitly. In addition to CoT problems, LLMs can also be prompted with multimodal descriptions and tools, effectively dividing complex tasks into sub-tasks [292, 293].

Visual Reasoning Application. Recent visual reasoning systems [219, 294–296] tend to apply LLMs for better visual information analysis and visual-language integration. Different from previous works [297, 298] that rely on limited VQA datasets and small-scale neural networks, current LLM-aided methods offer benefits of stronger generalization ability, emergent ability, and interactivity [58]. To realize visual reasoning with the help of LLMs, prompting and fine-tuning techniques can also be utilized: for example, PointClip V2 [295] applies LLMs to generate 3D-specific prompts, which are encoded as textual features and then combined with visual features for 3D recognition; and GPT4Tools [31] employs LoRA [253] to fine-tune LLMs following tool-related instructions. Serving as a controller [296], decision maker [299], or semantics refiner [294, 300], LLMs significantly facilitates the progress of visual reasoning research.

3.8 Summary and Discussion

3.8.1 Architecture. Due to the gigantic scale of LLMs, minor changes in architecture and training strategies have a big impact on performance and stability. Here, we summarize key architectural modules used in various LLMs, leading to better performance, reduced training time and memory, and better training stability.

Layer Normalization. The performance and training stability of LLMs are affected significantly by layer normalization. Pre-norm, that is normalizing inputs rather than outputs, is more common among LLMs stabilizing the training [6, 111, 130]. BLOOM [13] and AlexaTM [125] utilize an additional layer normalization before embedding layer to stabilize the training of large-scale models, while the model's zero-shot generalization ability can be negatively impacted [13]. However, another study [33] finds that pre-norm degrades fine-tuned model performance as compared to post-norm, and there are no stability benefits of pre-norm beyond the 100B scale. Therefore, GLM-130B [33] used deep-norm which is a variant of post-norm for better downstream task performance after fine-tuning.

Positional Encoding. Like other building blocks of the model, positional encoding also affects the performance and training stability of LLMs. BLOOM [13] finds ALiBi outperforms learned and rotary positional encodings. Contrary to this, GLM-130B [33] identifies rotary positional encoding as being better than ALiBi. So, there is no conclusion in the literature about positional encodings yet.

Parallel Attention. In this type of attention, feed-forward and attention layers are parallel to each other rather than sequential in a transformer block. It has been shown to reduce training time by 15%. There is no evidence of performance drop due to this change in the literature and it is used by the models PaLM [15], GPT-NeoX [121], and CodeGen [144].

Multi-Query Attention. It has shared key and value attention heads in a transformer block while query attention heads are projected as usual. This reduces memory usage and speeds up sampling in autoregressive decoding. No performance degradation has been observed with this change and it makes the training efficient allowing larger batch sizes. Multi-query attention is used in [15, 146].

MoE. This type of architecture enables easily scaling models to trillions of parameters [91, 92]. Only a few experts are activated during the computation making them compute-efficient. The performance of MoE models is better than dense models for the same amount of data and requires less computation during fine-tuning to achieve performance similar to dense models as discussed in [91]. MoE architectures are less prone to catastrophic forgetting, therefore are more suited for continual learning [92]. Extracting smaller sub-models for downstream tasks is possible without losing any performance, making MoE architecture hardware-friendly [92].

Sparse vs. Dense Activated. GPT-3 [6] uses sparse transformers [67] whereas GLaM [91] and PanGu-$\sum$ [92] use MoE [124] architectures to lower computational costs and increase the model size and capacity. According to the literature, sparse modules do not degrade the model's performance [67]. However, more experiments are required to verify this statement.

3.8.2 Training Strategies. Training models at a huge scale require tricks to reduce training costs, avoid loss divergence, and achieve better performance. We summarize and discuss some of these key tricks used in different LLMs.

Mixed Precision. It is a famous method for LLMs to reduce memory usage and improve training efficiency. In mixed precision, forward and backward passes are performed in FP16 format whereas optimizer states and master weights are kept in FP32 format [123]. A drawback associated with this format change is training instability due to a smaller value range resulting in loss spikes [33]. An alternative to FP16 is BF16 which has a comparatively larger range and performs precision-sensitive operations like gradient accumulation and softmax in FP32 [13]. BF16 has better performance and training stability but uses more memory and is supported on specific hardware, for example, A100 GPUs. Therefore, its adoption in LLMs is limited.

Training Instability. Loss divergence or spiking is a common issue in LLMs that occurs multiple times during training. This happens in the presence of gradient clipping [15]. To mitigate this problem, many approaches suggest restarting training from an earlier checkpoint [15, 33, 91], skipping 200–500 earlier data batches at the point of divergence in [15] and re-shuffling batches in [91]. The embedding layer gradient shrink proves to further stabilize the training as its gradient norm is significantly larger than the other layers [33]. Another suggestion to improve training stability for larger models is not to use *biases* in dense and norm layers as in [15].

Weight Initialization. It plays a significant role in model convergence and training stability. GPT-NeoX [121] initializes feed-forward layers before residuals with $\frac{2}{L\sqrt{d}}$ as in [157] and other layers with the small initialization scheme [301]. This avoids activations growing exponentially with increasing depth. MT-NLG [120] found higher variance for weight initialization leads to unstable training, hence validating small initialization scheme [301]. Various models perform random weight initialization which can cause bad initialization, Galactica [152] suggests a longer warmup to negate the effect.

Learning Rate. A suitable learning rate is important for stable training. It is suggested to use a lower value [13, 15, 127] with warmup and decay (cosine or linear). Usually, the learning rate is within the range $1e^{-4}$ to $8e^{-4}$. Moreover, MT-NLG (530B) [120] and GPT-NeoX (20B) [121] suggest

interpolating learning rates based on the model size using the GPT-3 [6] models ranging between 13B and 175B. This avoids tuning the learning rate hyperparameter.

Training Parallelism. 3D parallelism, a combination of data, pipeline, and tensor parallelism, is the most utilized training parallelism approach in LLMs [13–15, 33, 115, 118, 120]. In addition to 3D parallelism, BLOOM [13] uses a zero optimizer [37] to shard optimizer states. PanGu-α [111] and PanGu-Σ [92] go beyond 3D parallelism and apply 5D parallelism which additionally contains optimizer parallelism and rematerialization.

Mode Switching. It adds task-related tokens at the beginning of the text during training. These tokens refer to the NLU and NLG tasks which are shown to improve downstream task performance in [125, 127, 128]. During fine-tuning and inference, tokens are appended based on the downstream tasks.

Controllable Text Generation. Generating credible and controlled text from a pre-trained model is challenging. GPT-3 [6] and other LLMs use ICL to control generated text. While ICL helps in controlling the generated text, ERNIE 3.0 Titan [35] suggests using adversarial loss to rank its generated text for credibility and soft prompts such as genre, topic, keywords, sentiment, and length for better control on generated text.

3.8.3 Supervised Models vs. Generalized Models. Although generalized models are capable of performing diverse tasks with good performance they have not yet outperformed models trained in supervised settings. The supervised trained models are still state-of-the-art in various NLP tasks by a large margin as shown in [6, 15, 18].

3.8.4 Zero-Shot vs. Few-Shot. LLMs perform well in zero-shot and few-shot settings. But the performance difference between zero-shot and few-shot is large for pre-trained models [6, 15], naming LLMs as meta-learners [6]. LLMs zero-shot evaluations underperform unsupervised methods in neural **machine translation (MT)** [6]. The literature shows pre-training is not enough for good zero-shot performance [15, 16]. To improve the zero-shot performance the literature suggests using instruction fine-tuning that improves the zero-shot performance significantly and outperforms baselines. Instruction fine-tuning has also been shown to improve zero-shot generalization to unseen tasks. Another model, Flan-PaLM [16], unlocks zero-shot reasoning with CoT training.

3.8.5 Encoder vs. Decoder vs. Encoder-Decoder. Traditionally, these architectures perform well for different tasks, for example, encoder-only for NLU tasks, decoder-only for NLG, and encoder-decoder for sequence2sequence modeling. Encoder-only models are famous for smaller models such as Bert [7], RoBERTa [302], and so on., whereas LLMs are either decoder-only [6, 13, 121] or encoder-decoder [10, 11, 125]. While decoder-only models are good at NLG tasks, various LLMs, PaLM [15], OPT [14], GPT-3 [6], BLOOM [13], and LLaMA [160] are decoder-only models with significant performance gains on both NLU and NLG tasks. In contradiction to this, T5 [10] and UL2 [128] identify encoder-decoder models out-performing decoder-only models. In another study, PaLM [15] finds increasing the size of decoder-only models can reduce the performance gap between decoder-only and encoder-decoder architectures.

Although decoder-only architectures have become a trend for LLMs, many recently proposed approaches [125, 128] use mode-switching tokens in text with encoder-decoder architectures to enable task-specific modes. Similarly, CodeT5+ [34] uses an encoder-decoder architecture with multiple training objectives for different tasks, activating the encoder, decoder, or both according to the tasks. These variations in architecture and training objectives allow a model to perform well in different settings. Because of this dynamic configuration, the future of LLMs can be attributed to encoder-decoder architectures.

Table 3. Summary of Pre-Trained LLMs (>10B)

Models	Publication Venue	License Type	Model Creators	Purpose	No. of Params	Commercial Use	Steps Trained	Data/ Tokens	Data Cleaning	No. of Processing Units	Processing Unit Type	Training Time	Calculated Train. Cost	Training Parallelism	Library
T5 [10]	JMLR'20	Apache-2.0	Google	General	11B	✓	1M	1T	Heur+Dedup	1,024	TPU v3	-	-	D+M	Mesh TensorFlow
GPT-3 [6]	NeurIPS'20		OpenAI	General	175B	×	-	300B	Dedup+QF	-	V100	-	-	M	-
mT5 [11]	NAACL'21	Apache-2.0	Google	General	13B	✓	1M	1T	-	-	-	-	-	-	-
PanGu-α [111]	arXiv'21	Apache-2.0	Huawei	General	200B	✓	260k	1.1TB	Heur+Dedup	2,048	Ascend 910	-	-	D+OP+P+O+R	MindSpore
CPM-2 [12]	AI Open'21	MIT	Tsinghua	General	198B	✓	1M	2.6TB	Dedup	-	-	-	-	D+M	JAXFormer
Codex [145]	arXiv'21	-	OpenAI	Coding	12B	×	-	100B	Heur	-	-	-	-	-	-
ERNIE 3.0 [113]	arXiv'21	-	Baidu	General	10B	×	120k*	375B	Heur+Dedup	384	V100	-	-	M*	PaddlePaddle
Jurassic-1 [115]	White-Paper'21	Apache-2.0	AI21	General	178B	✓	-	300B	-	800	GPU	-	-	D+M+P	Megatron+DS
HyperCLOVA [117]	EMNLP'21	-	Naver	General	82B	×	-	300B	Clf+Dedup+PF	1,024	A100	321 h	1.32 Mil	M	Megatron
Yuan 1.0 [118]	arXiv'21	Apache-2.0	-	General	245B	✓	26k*	180B	Heur+Clf+Dedup	2,128	GPU	-	-	D+T+P	-
Gopher [119]	arXiv'21	-	Google	General	280B	×	-	300B	QF+Dedup	4,096	TPU v3	920 h	13.19 Mil	D+M	JAX+Haiku
ERNIE 3.0 Titan [35]	arXiv'21	-	Baidu	General	260B	×	-	300B	Heur+Dedup	-	Ascend 910	-	-	D+M+P+D*	PaddlePaddle
GPT-NeoX-20B [121]	BigScience'22	Apache-2.0	EleutherAI	General	20B	✓	150k	825GB	None	96	40G A100	-	-	M	Megatron+DS+PyTorch
OPT [14]	arXiv'22	MIT	Meta	General	175B	✓	150k	180B	Dedup	992	80G A100	-	-	D+T	Megatron
BLOOM [13]	arXiv'22	RAIL-1.0	BigScience	General	176B	✓	-	366B	Dedup+PR	384	80G A100	2,520 h	3.87 Mil	D+T+P	Megatron+DS
Galactica [152]	arXiv'22	Apache-2.0	Meta	Science	120B	×	225k	106B	Dedup	128	80GB A100	-	-	-	Metaseq
GLaM [91]	ICML'22	-	Google	General	1.2T	×	600k*	600B	Clf*	1,024	TPU v4	-	-	M	GSPMD
LaMDA [154]	arXiv'22	-	Google	Dialog	137B	×	3M	2.81T	Filtered	1,024	TPU v3	1,384 h	4.96 Mil	D+M	Lingvo
MT-NLG [120]	arXiv'22	Apache-v2.0	MS.+Nvidia	General	530B	×	-	270B	-	4,480	80G A100	-	-	D+T+P	Megatron+DS
AlphaCode [146]	Science'22	Apache-v2.0	Google	Coding	41B	✓	205k	967B	Heur+Dedup	-	TPU v4	-	-	M	JAX+Haiku
Chinchilla [96]	arXiv'22	-	Google	General	70B	×	-	1.4T	QF+Dedup	-	TPUv4	-	-	-	JAX+Haiku
PaLM [15]	arXiv'22	-	Google	General	540B	×	255k	780B	Heur	6,144	TPU v4	-	-	D+M	JAX+T5X
AlexaTM [125]	arXiv'22	Apache v2.0	Amazon	General	20B	×	500k	1.1T	Filtered	128	A100	2,880 h	1.47 Mil	M	DS
U-PaLM [127]	arXiv'22	-	Google	General	540B	×	20k	-	-	512	TPU v4	120 h	0.25 Mil	-	-
UL2 [128]	ICLR'23	Apache-2.0	Google	General	20B	✓	2M	1T	-	512	TPU v4	-	-	M	JAX+T5X
GLM [33]	ICLR'23	Apache-2.0	Multiple	General	130B	×	-	400B	-	768	40G A100	1,440 h	3.37 Mil	M	-
CodeGen [144]	ICLR'23	Apache-2.0	Salesforce	Coding	16B	✓	650k	577B	Heur+Dedup	-	TPU v4	-	-	D+M	JAXFormer
LLaMA [130]	arXiv'23	-	Meta	General	65B	×	350k	1.4T	Clf+Heur+Dedup	2,048	80G A100	504 h	4.12 Mil	D+M	xFormers
PanGu-Σ [92]	arXiv'23	-	Huawei	General	1.085T	×	-	329B	-	512	Ascend 910	2,400 h	-	D+OP+P+O+R	MindSpore
BloombergGPT [155]	arXiv23	-	Bloomberg	Finance	50B	×	139k	569B	Dedup	512	40G A100	1,272 h	1.97 Mil	M	PyTorch
Xuan Yuan 2.0 [156]	arXiv23	RAIL-1.0	Du Xiaoman	Finance	176B	✓	-	366B	Filtered	-	80GB A100	-	-	P	DS
CodeT5+ [34]	arXiv'23	BSD-3	Salesforce	Coding	16B	✓	110k	51.5B	Dedup	16	40G A100	-	-	-	DS
StarCoder [151]	arXiv'23	OpenRAIL-M	BigCode	Coding	15.5B	✓	250k	1T	Dedup+QF+PF	512	80G A100	624 h	1.28 Mil	D+T+P	Megatron-LM
LLaMA-2 [21]	arXiv'23	LLaMA-2.0	Meta	General	70B	✓	500k	2T	Minimal Filtering	-	80G A100	1.7Mh	-	-	-
PaLM-2 [126]	arXiv'23	-	Google	General	-	×	-	-	Ddedup+PF+QF	-	-	-	-	-	-
LLaMA-3.1 [133]	arXiv'24	LLaMA-3.0	Meta	General	405B	✓	1.2M	15T	Dedup+QF	16k	80G H100	30.84Mh	-	D+T+P+C	PyTorch
Mixtral 8x22B [134]	web'24	Apache-2.0	Mistral AI	General	141B	✓	-	-	-	-	-	-	-	-	-
Snowflake Arctic [135]	web'24	Apache-2.0	Snowflake	General	480B	✓	-	3.5T	-	-		-	-	T+P	DS
Nemotron-4 340B [140]	web'24	Nvidia	Nvidia	General	340B	✓	-	9T	-	6,144	80G H100	-	-	D+T+P	-
DeepSeek [141]	arXiv'24	MIT	DeepSeek	General	67B	✓	-	2T	Dedup+QF	-	-	300.6Kh	-	D+T+P	DS
DeepSeek-v2 [142]	arXiv'24	MIT	DeepSeek	General	67B	✓	-	8.1T	QF	-	H800	172.8Kh	-	D+P	HAI-LLM
DeepSeek-v3 [143]	arXiv'25	MIT	DeepSeek	General	671B	✓	-	14.8T	QF	-	H800	2,788Kh	5.576 Mil	D+P+E	HAI-LLM

Only the LLMs discussed individually in the previous sections are summarized. "Data/Tokens" is the model's pre-training data, which is either the number of tokens or the data size. "Data Cleaning" indicates whether data cleaning is performed or not. This includes heuristics (Heur), deduplication (Dedup), quality filtering (QF), and privacy filtering (PF). "Cost" is the calculated training cost obtained by multiplying the GPUs/TPUs hourly rate with the number of GPUs and the training time. The actual cost may vary due to many reasons such as using in-house GPUs or getting a discounted rate, re-training, number of employees working on the problem, and so on. "Training Parallelism" indicates distributed training using data parallelism (D), tensor parallelism (T), pipeline parallelism (P), context parallelism (C), model parallelism (M), expert parallelism (E), optimizer parallelism (OP), and rematerialization (R), where for "Library" column, "DS" is a short form for Deep Speed. In column "Commercial Use," we assumed a model is for non-commercial purposes if its license is unavailable.

4 Model Configurations

We provide different statistics of pre-trained and instruction-tuned models in this section. This includes information such as publication venue, license type, model creators, steps trained, parallelism, and so on in Tables 3 and 4. Architecture details of pre-trained LLMs are available in Table 5. Providing these details for instruction-tuned models is unnecessary because it fine-tunes pre-trained models for instruction datasets. Hence, architectural details are the same as the baselines. Moreover, optimization settings for various LLMs are available in Tables 6 and 7. We do not include details on precision, warmup, and weight decay in Table 7. These details are not as important as others to mention for instruction-tuned models and are not provided by the papers.

5 Datasets and Evaluation

Generating training and evaluation datasets is expensive because of the large-scale data demand of LLMs. Hence, datasets for training and benchmarking these models are topics of key importance. A summary of datasets commonly used by LLMs is provided next.

Table 4. Summary of Instruction-Tuned LLMs (>10B)

Models	Publication Venue	License Type	Model Creators	Purpose	No. of Params	Commercial Use	Pre-trained Models	Steps Trained	Data/ Tokens	No. of Processing Units	Processing Unit Type	Train. Time	Calculated Train. Cost	Train. Parallelism	Library
WebGPT [169]	arXiv'21	-	OpenAI	General	175B	×	GPT-3	-	-	-	-	-	-	-	-
T0 [17]	ICLR'22	Apache-2.0	BigScience	General	11B	✓	T5	-	250B	512	TPU v3	270 h	0.48 Mil	-	-
Tk-Instruct [18]	EMNLP'22	MIT	AI2+	General	11B	✓	T5	1,000	-	256	TPU v3	4 h	0.0036 Mil	-	Google T5
OPT-IML [97]	arXiv'22	-	Meta	General	175B	×	OPT	8k	2B	128	40G A100	-	-	D+T	Megatron
Flan-U-PaLM [16]	ICLR'22	Apache-2.0	Google	General	540B	✓	U-PaLM	30k	-	512	TPU v4	-	-	-	JAX+T5X
mT0 [158]	ACL'23	Apache-2.0	HuggingFace+	General	13B	✓	mT5	-	-	-	-	-	-	-	-
Sparrow [170]	arXiv'22	-	Google	Dialog	70B	×	Chinchilla	-	-	64	TPU v3	-	-	M	-
WizardCoder [167]	arXiv'23	Apache-2.0	HK Bapt.	Coding	15B	×	StarCoder	200	S-78k	-	-	-	-	-	-
Alpaca [161]	GitHub '23	Apache-2.0	Stanford	General	13B	✓	LLaMA	3-Epoch	S-52k	8	80G A100	3 h	600	FSDP	PyTorch
Vicuna [162]	GitHub '23	Apache-2.0	LMSYS	General	13B	✓	LLaMA	3-Epoch	S-125k	-	-	-	-	FSDP	PyTorch
LIMA [188]	arXiv'23	-	Meta+	General	65B	-	LLaMA	15-Epoch	S-1000	-	-	-	-	-	-
Koala [303]	GitHub '23	Apache-2.0	UC-Berkley	General	13B	×	LLaMA	2-Epoch	S-472k	8	A100	6 h	100	-	JAX/FLAX

All abbreviations are the same as Table 3. Entries in "Data/Tokens" starting with "S-" represent the number of training samples.

Table 5. Architecture Details of LLMs

Models	Type	Training Objective	Attention	Vocab	Tokenizer	Norm	PE	Activation	Bias	nL	nH	HS
T5 (11B)	Enc-Dec	Span Corruption	Standard	32k	SentencePiece	Pre-RMS	Relative	ReLU	×	24	128	1,024
GPT3 (175B)	Causal-Dec	Next Token	Dense+Sparse	-	-	Layer	Learned	GeLU	✓	96	96	12,288
mT5 (13B)	Enc-Dec	Span Corruption	Standard	250k	SentencePiece	Pre-RMS	Relative	ReLU	-	-	-	-
PanGu-α (200B)	Causal-Dec	Next Token	Standard	40k	BPE	Layer	-	-	-	64	128	16,384
CPM-2 (198B)	Enc-Dec	Span Corruption	Standard	250k	SentencePiece	Pre-RMS	Relative	ReLU	-	24	64	-
Codex (12B)	Causal-Dec	Next Token	Standard	-	BPE+	Pre-Layer	Learned	GeLU	-	96	96	12,288
ERNIE 3.0 (10B)	Causal-Dec	Next Token	Standard	-	WordPiece	Post-Layer	Relative	GeLU	-	48	64	4,096
Jurassic-1 (178B)	Causal-Dec	Next Token	Standard	256k	SentencePiece*	Pre-Layer	Learned	GeLU	✓	76	96	13,824
HyperCLOVA (82B)	Causal-Dec	Next Token	Dense+Sparse	-	BPE*	Pre-Layer	Learned	GeLU	-	64	80	10,240
Yuan 1.0 (245B)	Causal-Dec	Next Token	Standard	-	-	-	-	-	-	76	-	16,384
Gopher (280B)	Causal-Dec	Next Token	Standard	32k	SentencePiece	Pre-RMS	Relative	GeLU	✓	80	128	16,384
ERNIE 3.0 Titan (260B)	Causal-Dec	Next Token	Standard	-	WordPiece	Post-Layer	Relative	GeLU	-	48	192	12,288
GPT-NeoX-20B	Causal-Dec	Next Token	Parallel	50k	BPE	Layer	Rotary	GeLU	✓	44	64	-
OPT (175B)	Causal-Dec	Next Token	Standard	-	BPE	-	-	ReLU	✓	96	96	-
BLOOM (176B)	Causal-Dec	Next Token	Standard	250k	BPE	Layer	ALiBi	GeLU	✓	70	112	14,336
Galactica (120B)	Causal-Dec	Next Token	Standard	50k	BPE+custom	Layer	Learned	GeLU	×	96	80	10,240
GLaM (1.2T)	MoE-Dec	Next Token	Standard	256k	SentencePiece	Layer	Relative	GeLU	✓	64	128	32,768
LaMDA (137B)	Causal-Dec	Next Token	Standard	32k	BPE	Layer	Relative	GeGLU	-	64	128	8,192
MT-NLG (530B)	Causal-Dec	Next Token	Standard	50k	BPE	Pre-Layer	Learned	GeLU	✓	105	128	20,480
AlphaCode (41B)	Enc-Dec	Next Token	Multi-query	8k	SentencePiece	-	-	-	-	64	128	6,144
Chinchilla (70B)	Causal-Dec	Next Token	Standard	32k	SentencePiece-NFKC	Pre-RMS	Relative	GeLU	✓	80	64	8,192
PaLM (540B)	Causal-Dec	Next Token	Parallel+Multi-query	256k	SentencePiece	Layer	RoPE	SwiGLU	×	118	48	18,432
AlexaTM (20B)	Enc-Dec	Denoising	Standard	150k	SentencePiece	Pre-Layer	Learned	GeLU	✓	78	32	4,096
Sparrow (70B)	Causal-Dec	Pref.&Rule RM	-	32k	SentencePiece-NFKC	Pre-RMS	Relative	GeLU	✓	16*	64	8,192
U-PaLM (540B)	Non-Causal-Dec	MoD	Parallel+Multi-query	256k	SentencePiece	Layer	RoPE	SwiGLU	×	118	48	18,432
UL2 (20B)	Enc-Dec	MoD	Standard	32k	SentencePiece	-	-	-	-	64	16	4,096
GLM (130B)	Non-Causal-Dec	AR Blank Infilling	Standard	130k	SentencePiece	Deep	RoPE	GeGLU	✓	70	96	12,288
CodeGen (16B)	Causal-Dec	Next Token	Parallel	-	BPE	Layer	RoPE	-	-	34	24	-
LLaMA (65B)	Causal-Dec	Next Token	Standard	32k	BPE	Pre-RMS	RoPE	SwiGLU	-	80	64	8,192
PanGu-Σ (1085B)	Causal-Dec	Next Token	Standard	-	BPE	Fused Layer	-	FastGeLU	-	40	40	5,120
BloombergGPT (50B)	Causal-Dec	Next Token	Standard	131k	Unigram	Layer	ALiBi	GeLU	✓	70	40	7,680
Xuan Yuan 2.0 (176B)	Causal-Dec	Next Token	Self	250k	BPE	Layer	ALiBi	GeLU	✓	70	112	14,336
CodeT5+ (16B)	Enc-Dec	SC+NT+Cont.+Match	Standard	-	Code-Specific	-	-	-	-	-	-	-
StarCoder (15.5B)	Causal-Dec	FIM	Multi-query	49k	BPE	-	Learned	-	-	40	48	6,144
LLaMA-2 (70B)	Causal-Dec	Next Token	Grouped-query	32k	BPE	Pre-RMS	RoPE	SwiGLUE	-	-	-	-
PaLM-2	-	MoD	Parallel	-	-	-	-	-	-	-	-	-
LLaMA-3.1 (405B)	Causal-Dec	Next Token	Grouped-query	128k	BPE	Pre-RMS	RoPE	SwiGLU	-	126	128	16,384
Nemotron-4 (340B)	Causal-Dec	Next Token	Standard	256k	SentencePiece	-	RoPE	ReLU	×	96	96	18,432
DeepSeek (67B)	Causal-Dec	Next Token	Grouped-query	100k	BBPE	Pre-RMS	RoPE	SwiGLU	-	95	64	8,192
DeepSeek-v2 (67B)	MoE-Dec	Next Token	Multi-Head Latent	100k	BBPE	Pre-RMS	RoPE	SwiGLU	-	60	128	5,120
DeepSeek-v3 (671B)	MoE-Dec	Multi Token	Multi-Head Latent	128k	BBPE	Pre-RMS	RoPE	SwiGLU	-	61	128	7,168

Here, "PE" is the positional embedding, "nL" is the number of layers, "nH" is the number of attention heads, "HS" is the size of hidden states.

5.1 Training Datasets

The performance of LLMs largely depends on the training data's quality, size, and diversity. Preparing training datasets of high quality at a large scale is laborious. Researchers have suggested various pre-training and fine-tuning datasets to enhance LLMs capabilities. We summarize these efforts in Table 8. While numerous training datasets are available in the literature, we cover the most widely used ones in our summary.

Table 6. Summary of Optimization Settings Used for Pre-Trained LLMs

Models	Batch Size	Sequence Length	LR	Warmup	LR Decay	Optimizers			Precision			Weight Decay	Grad Clip	Dropout
						AdaFactor	Adam	AdamW	FP16	BF16	Mixed			
T5 (11B)	2^{11}	512	0.01	×	Inverse square root	✓			-	-	-	-	-	✓
GPT3 (175B)	32K	-	6e-5	✓	Cosine		✓		✓			✓	✓	-
mT5 (13B)	1,024	1,024	0.01	-	Inverse square root	✓			-	-	-	-	-	✓
PanGu-α (200B)	-	1,024	2e-5	-	-	-	-	-	-	✓	-	-	-	-
CPM-2 (198B)	1,024	1,024	0.001	-	-	✓			-	-	-	-	-	✓
Codex (12B)	-	-	6e-5	✓	Cosine		✓		✓			✓	-	-
ERNIE 3.0 (12B)	6,144	512	1e-4	✓	Linear		✓		-	-	-	✓	-	-
Jurassic-1 (178B)	3.2M	2,048	6e-5	✓	Cosine		✓		✓			✓	✓	-
HyperCLOVA (82B)	1,024	-	6e-5	-	Cosine			✓	-	-	-	✓	-	-
Yuan 1.0 (245B)	<10M	2,048	1.6e-4	✓	Cosine decay to 10%		✓		-	-	-	✓	-	-
Gopher (280B)	3M	2,048	4e-5	✓	Cosine decay to 10%		✓			✓		-	✓	-
ERNIE 3.0 Titan (260B)	-	512	1e-4	✓	Linear		✓		✓			✓	✓	-
GPT-NeoX-20B	1,538	2,048	0.97e-5	✓	Cosine			✓	✓			✓	✓	×
OPT (175B)	2M	2,048	1.2e-4	-	Linear			✓	✓			✓	✓	✓
BLOOM (176B)	2,048	2,048	6e-5	✓	Cosine		✓			✓		✓	✓	×
Galactica (120B)	2M	2,048	7e-6	✓	Linear decay to 10%			✓	-	-	-	✓	✓	✓
GLaM (1.2T)	1M	1,024	0.01	-	Inverse square root	✓				FP32 + ✓		-	✓	×
LaMDA (137B)	256K	-	-	-	-	-	-	-	-	-	-	-	-	-
MT-NLG (530B)	1,920	2,048	5e-5	✓	Cosine decay to 10%		✓			✓		✓	✓	-
AlphaCode (41B)	2,048	1,536 + 768	1e-4	✓	Cosine decay to 10%			✓		✓		✓	✓	-
Chinchilla (70B)	1.5M	2,048	1e-4	✓	Cosine decay to 10%			✓		✓		-	-	-
PaLM (540B)	2,048	2,048	0.01	-	Inverse square root	✓			-	-	-	✓	✓	×
AlexaTM (20B)	2M	1,024	1e-4	-	Linear decay to 5%		✓			✓		✓	-	✓
U-PaLM (540B)	32	2,048	1e-4	-	Cosine	✓			-	-	-	-	-	-
UL2 (20B)	1,024	1,024	-	-	Inverse square root	-	-	-	-	-	-	×	-	-
GLM (130B)	4,224	2,048	8e-5	✓	Cosine			✓	✓			✓	✓	✓
CodeGen (16B)	2M	2,048	5e-5	✓	Cosine		✓		-	-	-	✓	✓	-
LLaMA (65B)	4M Tokens	2,048	1.5e-4	✓	Cosine decay to 10%			✓	-	-	-	✓	✓	-
PanGu-Σ (1.085T)	512	1,024	2e-5	✓	-		✓				✓	-	-	-
BloombergGPT (50B)	2,048	2,048	6e-5	✓	Cosine			✓			✓	✓	✓	×
Xuan Yuan 2.0 (176B)	2,048	2,048	6e-5	✓	Cosine		✓		✓			✓	✓	-
CodeT5+ (16B)	2,048	1,024	2e-4	-	Linear			✓			✓	✓	-	-
StarCoder (15.5B)	512	8k	3e-4	✓	COSINE		✓			✓		✓	-	-
LLaMA-2 (70B)	4M Tokens	4k	1.5e-4	✓	Cosine			✓		✓		✓	✓	-
LLaMA-3.1 (405B)	16M	8,192	8e-5	✓	Linear+cosine			✓		✓		-	-	-
Nemotron-4 (340B)	2.304	4,096	-	-	Linear	-	-	-		✓		-	-	×
DeepSeek (67B)	4,608	4,096	3.2e-4	✓	Cosine			✓		✓		✓	✓	-
DeepSeek-v2 (67B)	9,216	4k	2.4e-4	✓	Step-decay			✓	-	-	-	✓	✓	-
DeepSeek-v3 (671B)	15,360	4k	2.2e-4	-	Cosine			✓			✓	✓	✓	-

The values for weight decay, gradient clipping, and dropout are 0.1, 1.0, and 0.1, respectively, for most of the LLMs.

Table 7. Summary of Optimization Settings Used for Instruction-Tuned LLMs

Models	Batch Size	Sequence Length	LR	Warmup	LR_Decay	Optimizers			Grad Clip	Dropout
						AdaFactor	Adam	AdamW		
WebGPT (175B)	BC:512, RM:32	-	6e-5	-	-		✓		-	-
T0 (11B)	1,024	1,280	1e-3	-	-	✓			-	✓
Tk-Instruct (11B)	1,024	-	1e-5	-	Constant	-	-	-	-	-
OPT-IML (175B)	128	2,048	5e-5	×	Linear		✓		✓	✓
Flan-U-PaLM (540B)	32	-	1e-3	-	Constant	✓			-	✓
Sparrow (70B)	RM: 8 + 16, RL:16	-	2e-6	✓	Cosine decay to 10%	✓			✓	×
WizardCoder (15B)	512	2,048	2e-5	✓	Cosine	-	-	-	-	-
Alpaca (13B)	128	512	1e-5	✓	Cosine	-	-	✓	✓	×
Vicuna (13B)	128	-2,048	2e-5	✓	Cosine			✓	-	×
LIMA (65B)	32	2,048	1e-5	×	Linear			✓	-	✓

Values for gradient clipping and dropout are the same as the pre-trained models, while no model uses weight decay for instruction tuning.

5.2 Evaluation Datasets and Tasks

The evaluation of LLMs is important in gauging their proficiency and limitations. This process measures the model's ability to comprehend, generate, and interact with human language across a spectrum of tasks. Evaluating a LM is divided into two broader categories: (1) NLU and (2) NLG. It is emphasized that tasks in NLU and NLG are softly categorized and are often used interchangeably in the literature.

Table 8. Details of Various Well-Known Pre-Training and Fine-Tuning Datasets

Dataset	Type	Size/Samples	Tasks	Source	Creation	Comments
C4 [10]	Pre-train	806GB	-	Common Crawl	Automated	A clean, multilingual dataset with billions of tokens
mC4 [11]	Pre-train	38.49TB	-	Common Crawl	Automated	A multilingual extension of the C4 dataset, mC4 identifies over 100 languages using cld3 from 71 monthly web scrapes of Common Crawl
PILE [304]	Pre-train	825GB	-	Common Crawl, PubMed Central, OpenWebText2, ArXiv, GitHub, Books3, and others	Automated	A massive dataset comprised of 22 constituent sub-datasets
ROOTs [305]	Pre-train	1.61TB	-	498 Hugging Face datasets	Automated	46 natural and 13 programming languages
MassiveText [119]	Pre-train	10.5TB	-	MassiveWeb, Books, News, Wikipedia, GitHub, C4	Automated	99% of the data is in English
Wikipedia [306]	Pre-train	-	-	Wikipedia	Automated	Dump of Wikipedia
RedPajama [307]	Pre-train	5TB	-	CommonCrawl, C4, Wikipedia, GitHub, Books, StackExchange	Automated	Open-source replica of LLaMA dataset
PushShift.io Reddit	Pre-train	21.1GB	-	Reddit	Automated	Submissions and comments on Reddit from 2005 to 2019
BigPython [144]	Pre-train	5.5TB	Coding	GitHub	Automated	-
Pool of Prompt (P3) [17]	Instructions	12M	62	PromptSource	Manual	A Subset of PromptSource, created from 177 datasets including summarization, QA, classification, and so on
xP3 [158]	Instructions	81M	71	P3+Multilingual datasets	Manual	Extending P3 to total 46 languages
Super-NaturalInstructions (SNI) [18]	Instructions	12.4M	1,616	Multiple datasets	Manual	Extending P3 with additional multilingual datasets, total 46 languages
Flan [16]	Instructions	15M	1,836	Muffin+T0-SF+NIV2	Manual	Total 60 languages
OPT-IML [97]	Instructions	18.1M	1,667	-	Manual	-
Self-Instruct [19]	Instructions	82k	175	-	Automated	Generated 52k instructions with 82k samples from 175 seed tasks using GPT-3
Alpaca [161]	Instructions	52k	-	-	Automated	Employed self-instruct method to generate data from text-davinci-003
Vicuna [162]	Instructions	125k	-	ShareGPT	Automated	Conversations shared by users on ShareGPT using public APIs
LLaMA-GPT-4 [163]	Instructions	52k	-	Alpaca	Automated	Recreated Alpaca dataset with GPT-4 in English and Chinese
Unnatural Instructions [308]	Instructions	68k	-	15-Seeds (SNI)	Automated	-
LIMA [188]	Instructions	1k	-	Multiple datasets	Manual	Carefully created samples to test performance with fine-tuning on less data
Anthropic-HH-RLHF [309]	Alignment	142k	-	-	Manual	
Anthropic-HH-RLHF-2 [181]	Alignment	39k	-	-	Manual	

Here, alignment means aligning with human preferences.

NLU. It measures the language understanding capacity of LMs. It encompasses multiple tasks, including sentiment analysis, text classification, NLI, QA, CR, **mathematical reasoning (MR)**, RC, and so on.

NLG. It assesses the language generation capabilities of LLMs by understanding the provided input context. It includes tasks such as summarization, sentence completion, MT, dialogue generation, and so on.

Numerous datasets are proposed for each task, evaluating LLMs against different characteristics. To provide an overview of evaluation datasets, we briefly discuss a few famous datasets within each category and offer a comprehensive list of datasets in Table 9. Moreover, we show a detailed

Table 9. Categorized Evaluation Datasets Used in Evaluating LLMs

Type	Datasets/Benchmarks
Multi-Task	MMLU [310], SuperGLUE [2], BIG-bench [311], GLUE [312], BBH [311], CUGE [313], ZeroCLUE [314], FewCLUE [315], Blended Skill Talk [316], HELM [317], KLUE-STS [318]
Language Understanding	CoQA [319], WiC [320], Wikitext103 [321], PG19 [322], LCQMC [323], QQP [324], WinoGender [325], CB [326], FinRE [327], SanWen [328], AFQMC [314], BQ Corpus [329], CNSS [330], CKBQA 13 [331], CLUENER [314], Weibo [332], AQuA [333], OntoNotes [334], HeadQA [335], Twitter Dataset [336]
Story Cloze and Sentence Completion	StoryCloze [337], LAMBADA [338], LCSTS [339], AdGen [340], E2E [341], CHID [342], CHID-FC [315]
Physical Knowledge and World Understanding	PIQA [343], TriviaQA [344], ARC [345], ARC-Easy [345], ARC-Challenge [345], PROST [346], OpenBookQA [347], WebNLG [348], DogWhistle Insider & Outsider [349]
Contextual Language Understanding	RACE [350], RACE-Middle [350], RACE-High [350], QuAC [351], StrategyQA [352], Quiz Bowl [353], cMedQA [354],cMedQA2 [355], MATINF-QA [356]
CR	WinoGrande [357], HellaSwag [358], COPA [359], WSC [360], CSQA [361], SIQA [362], C^3 [363], CLUEWSC2020 [314], CLUEWSC [314], CLUEWSC-FC [315], ReCoRD [364]
RC	SQuAD [365], BoolQ [366], SQUADv2 [367], DROP [368], RTE [369], WebQA [370], CMRC2017 [371], CMRC2018 [372], CMRC2019 [373], COTE-BD [374], COTE-DP [374], COTE-MFW [374], MultiRC [375], Natural Questions [376], CNSE [330], DRCD [377], DuReader [378], $\text{Dureader}_{\text{robust}}$ [379], DuReader-QG [378], SciQ [380], Sogou-log [381], $\text{Dureader}_{\text{robust}}$-QG [379], QA4MRE [382], KorQuAD 1.0 [383], CAIL2018-Task1 & Task2 [384]
MR	MATH [385], Math23k [386], GSM8K [387], MathQA [388], MGSM [389], MultiArith [390], ASDiv [391], MAWPS [392], SVAMP [393]
Problem-Solving	HumanEval [145], DS-1000 [394], MBPP [395], APPS [385], CodeContests [146]
NLI and Logical Reasoning	ANLI [396], MNLI-m [397], MNLI-mm [397],QNLI [365], WNLI [360], OCNLI [314], CMNLI [314], ANLI R1 [396], ANLI R2 [396], ANLI R3 [396], HANS [398], OCNLI-FC [315], LogiQA [399], StrategyQA [352]
Cross-Lingual Understanding	MLQA [400], XNLI [401], PAWS-X [402], XSum [403], XCOPA [404], XWinograd [405], TyDiQA-GoldP [406], MLSum [407]
Truthfulness and Fact Checking	TruthfulQA [408], MultiFC [409], Fact Checking on Fever [410]
Biases and Ethics in AI	ETHOS [411], StereoSet [412], BBQ [413], Winobias [414], CrowS-Pairs [415]
Toxicity	RealToxicityPrompts [416], CivilComments toxicity classification [417]
Language Translation	WMT [418], WMT20 [419], WMT20-enzh [419], EPRSTMT [315], CCPM [420]
Scientific Knowledge	AminoProbe [152], BioLAMA [152], Chemical Reactions [152], Galaxy Clusters [152], Mineral Groups [152]
Dialogue	Wizard of Wikipedia [421], Empathetic Dialogues [422], DPC-generated [96] dialogues, ConvAI2 [423], KdConv [424]
Topic Classification	TNEWS-FC [315], YNAT [318], KLUE-TC [318], CSL [314], CSL-FC [315], IFLYTEK [425]

overview of the training datasets and evaluation tasks and benchmarks used by various pre-trained LLMs in Table 10 and fine-tuned LLMs in Table 11. We also compare the top-performing LLMs in various NLP tasks in Table 12.

5.2.1 *Multi-Task.*

MMLU [310]. A benchmark that measures the knowledge acquired by models during pre-training and evaluates models in zero-shot and few-shot settings across 57 subjects, testing both world knowledge and problem-solving ability.

SuperGLUE [2]. A more challenging and diverse successor to the **general language understanding evaluation (GLUE)** [312] benchmark, SuperGLUE includes a variety of language understanding tasks, such as QA, NLI, and co-reference resolution. It is designed to provide a rigorous test of language understanding and requires significant progress in areas like sample-efficient, transfer, multi-task, and unsupervised or self-supervised learning.

Behavior of Intelligent Generative Models Benchmark (BIG-Bench) [311]. The BIG-bench is a large-scale benchmark designed to test the abilities of LLMs across a wide range of tasks, including reasoning, creativity, ethics, and understanding of specific domains.

GLUE [312]. The GLUE benchmark is a collection of resources for training, evaluating, and analyzing NLU systems. It includes a variety of tasks that test a wide range of linguistic phenomena, making it a comprehensive tool for evaluating language understanding in AI.

Table 10. An Illustration of Training Datasets and Evaluation Tasks Employed by Pre-Trained LLMs

Models	Training Dataset	Benchmark			QA	Clf	NLI	MT	Cloze/ Completion	RC	CR	MR	Coding	Truthful/ Bias/ Toxicity/ Mem.
		BIG-Bench	MMLU	Super GLUE										
T5	C4 [10]			✓	✓		✓	✓	✓	✓	✓	✓		
GPT-3	Common Crawl, WebText, Books Corpora, Wikipedia			✓	✓			✓	✓	✓				✓
mT5	mC4 [11]				✓		✓	✓						
PanGu-α	1.1TB Chinese Text Corpus				✓		✓		✓	✓	✓			
CPM-2	WuDaoCorpus [112]									✓		✓		
Codex	54 million public repositories from GitHub												✓	
ERNIE-3.0	Chinese text corpora, Baidu Search, Web text, QA-long, QA-short, Poetry and Couplet Domain-specific data from medical, law, and financial area Baidu knowledge graph with more than 50 million facts			✓	✓	✓	✓	✓	✓	✓		✓		
Jurassic-1	Wikipedia, OWT, Books, C4, Pile [304], arXiv, GitHub				✓		✓		✓	✓				
HyperCLOVA	Korean blogs, Community sites, News, KiN Korean Wikipedia, Wikipedia (English and Japanese), Modu-Corpus: Messenger, News, Spoken and written language corpus, Web corpus							✓						
Yuan 1.0	Common Crawl, SogouT, Sogou News, Baidu Baike, Wikipedia, Books				✓	✓	✓			✓				
Gopher	Subsets of MassiveWeb Books, C4, News, GitHub and Wikipedia samples from MassiveText	✓	✓	✓	✓						✓	✓		✓
ERNIE-3.0 TITAN	Same as ERNIE 3.0 and ERNIE 3.0 adversarial dataset, ERNIE 3.0 controllable dataset				✓	✓	✓		✓	✓				
GPT-NeoX-20B	Pile [304]			✓	✓		✓		✓		✓	✓		
OPT	RoBERTa [302], Pile [304], PushShift.io Reddit [426]				✓	✓					✓			✓
BLOOM	ROOTs [13]			✓			✓	✓	✓				✓	✓
Galactica	arXiv, PMC, Semantic Scholar, Wikipedia, StackExchange, LibreText, Open Textbooks, RefSeq Genome, OEIS, LIPID MAPS, NASAExoplanet, Common Crawl, ScientificCC, AcademicCC, GitHub repositories Khan Problems, GSM8K, OneSmallStep	✓	✓		✓							✓		✓
GLaM	Filtered Webpages, Social media conversations Wikipedia, Forums, Books, News				✓		✓		✓	✓	✓			
LaMDA	Infiniset : Public documents, Dialogs, Utterances													✓
MT-NLG	Two snapshots of Common Crawl and Books3, OpenWebText2, Stack Exchange, PubMed Abstracts, Wikipedia, PG-19 [242], BookCorpus2, NIH ExPorter, Pile, CC-Stories, RealNews						✓		✓	✓	✓			✓
AlphaCode	Selected GitHub repositories, CodeContests: Codeforces, Description2Code, CodeNet												✓	
Chinchilla	MassiveWeb, MassiveText Books, C4, News, GitHub, Wikipedia	✓	✓		✓					✓	✓			✓
PaLM	Web pages, books, Wikipedia, news, articles, source code, social media conversations	✓			✓			✓			✓		✓	✓
AlexaTM	Wikipedia, mC4			✓			✓	✓			✓			✓
U-PaLM	Same as PaLM	✓		✓	✓		✓		✓	✓	✓			
UL2	-			✓	✓	✓	✓					✓		✓
GLM-130B	-	✓	✓						✓					
CodeGen	Pile, BigQuery, BigPython												✓	
LLaMA	CommonCrawl, C4, GitHub, Wikipedia, Books, arXiv, StackExchange		✓		✓					✓	✓	✓	✓	✓
PanGu-Σ	WuDaoCorpora, CLUE, Pile, C4, Python code				✓	✓	✓	✓	✓				✓	
BloombergGPT	inPile, Pile, C4, Wikipedia	✓	✓				✓		✓	✓	✓			✓
CodeT5+	CodeSearchNet, GitHub Code											✓	✓	
StarCoder	The Stack v1.2		✓									✓	✓	✓
LLaMA-2	✓	✓		✓						✓	✓	✓	✓	
PaLM-2	Web documents, Code, Books, Maths, Conversation			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Here, "QA" is question answering, "Clf" is classification, "NLI" is natural language inference, "MT" is machine translation, "RC" is reading comprehension, "CR" is commonsense reasoning, "MR" is mathematical reasoning, and "Mem." is memorization.

5.2.2 *Language Understanding.*

WinoGrande [357]. A large-scale dataset inspired by the original **Winograd schema challenge (WSC)** [360] tests models on their ability to resolve pronoun ambiguity and encourages the development of models that understand the broad context in natural language text.

Table 11. An Illustration of Training Datasets and Evaluation Benchmarks Used in Fine-Tuned LLMs

Models	Training Dataset	BIG-Bench	MMLU	BBH	RAFT	Flan	SNI	PromptSource	TyDiQA	HumanEval	MBPP	Truthful/ Bias/ Toxicity
T0	Pool of Prompts	✓										
WebGPT	ELI5 [427], ELI5 fact-check [169], TriviaQA [344], ARC-Challenge [345], ARC-Easy [345], Hand-written data, Demonstrations of humans, Comparisons between model-generated answers											✓
Tk-INSTRUCT	SNI [18]						✓					
mT0	xP3 [158]											
OPT-IML	PromptSource [17], Flan [16], SNI [428], UnifiedSKG [429], CrossFit [430], ExMix [431], T5 [10], Reasoning		✓	✓	✓	✓	✓	✓				
Flan	Muffin, T0-SF, NIv2, CoT		✓	✓					✓			
WizardCoder	Code Alpaca									✓	✓	

"SNI" is a short of Super-NaturalInsturctions.

Table 12. Performance Comparison of Top-Performing LLMs across Various NLU and NLG Tasks

Task	Dataset/Benchmark	Top-1		Top-2		Top-3	
		Model (Size)	Score (N-shots)	Model (Size)	Score (N-shots)	Model (Size)	Score (N-shots)
Multi-Task	BIG-bench (B)	Chinchilla (70B)	65.1 (5-shot)	Gopher (280B)	53.97 (5-shot)	PaLM (540B)	53.7 (5-shot)
	MMLU (B)	GPT-4 (-)	86.4 (5-shot)	Gemini (Ultra)	83.7 (5-shot)	Flan-PaLM-2$_{(f)}$ (Large)	81.2 (5-shot)
Language Understanding	SuperGLUE (B)	ERNIE 3.0 (12B)	90.6 (-)	PaLM$_{(f)}$ (540B)	90.4 (-)	T5 (11B)	88.9 (-)
Story Comprehension and Generation	HellaSwag	GPT-4 (-)	95.3 (10-shot)	Gemini (Ultra)	87.8 (10-shot)	PaLM-2 (Large)	86.8 (one shot)
	StoryCloze	GPT3 (175B)	87.7 (few-shot)	PaLM-2 (Large)	87.4 (one shot)	OPT (175B)	79.82 (-)
Physical Knowledge and World Understanding	PIQA	PaLM-2 (Large)	85.0 (one shot)	LLaMA (65B)	82.8 (zero shot)	MT-NLG (530B)	81.99 (zero shot)
	TriviaQA	PaLM-2 (Large)	86.1 (one shot)	LLaMA-2 (70B)	85.0 (one shot)	PaLM (540B)	81.4 (one shot)
Contextual Language Understanding	LAMBADA	PaLM (540B)	89.7 (few-shot)	MT-NLG (530B)	87.15 (few-shot)	PaLM-2 (Large)	86.9 (one shot)
CR	WinoGrande	GPT-4 (-)	87.5 (5-shot)	PaLM-2 (Large)	83.0 (one shot)	PaLM (540B)	81.1 (zero shot)
	SIQA	LLaMA (65B)	52.3 (zero shot)	Chinchilla (70B)	51.3 (zero shot)	Gopher (280B)	50.6 (zero shot)
RC	BoolQ	PaLM$_{(f)}$ (540B)	92.2 (-)	T5 (11B)	91.2 (-)	PaLM-2 (Large)	90.9 (one shot)
Truthfulness	Truthful-QA	LLaMA (65B)	57 (-)				
MR	MATH	Gemini (Ultra)	53.2 (4-shot)	PaLM-2 (Large)	34.3 (4-shot)	LLaMA-2 (65B)	13.5 (4-shot)
	GSM8K	GPT-4 (-)	92.0 (5-shot)	PaLM-2 (Large)	80.7 (8-shot)	U-PaLM (540B)	58.5 (-)
Problem-Solving and Logical Reasoning	HumanEval	Gemini$_{(f)}$ (Ultra)	74.4 (zero shot)	GPT-4 (-)	67.0 (zero shot)	Code LLaMA (34B)	48.8 (zero shot)

Here, "N-Shots" indicate the number of example prompts provided to the model during the evaluation, representing its capability in few-shot or zero-shot learning settings, "f" represents the fine-tuned version, and "B" represents the benchmark.

CoQA [319]. A conversational QA dataset, CoQA, challenges models with questions that rely on conversation history and require free-form text answers. Its diverse content from seven domains makes it a rigorous test for models' ability to handle a wide range of topics and conversational contexts.

WiC [320]. This dataset assesses a model's ability to discern word meanings based on context, aiding in tasks related to Word Sense Disambiguation.

Wikitext103 [321]. With over 100 million tokens from Wikipedia's top articles, this dataset is a rich resource for tasks that require understanding long-term dependencies, such as LM and translation.

PG19 [322]. This is a digital library of diverse books from Project Gutenberg. It is specifically designed to facilitate research in unsupervised learning and LM, with a special focus on long-form content.

C4 [10]. A clean, multilingual dataset, C4 offers billions of tokens from web-crawled data. It is a comprehensive resource for training advanced Transformer models in various languages.

Large-Scale Chinese Question Matching Corpus (LCQMC) [323]. The LCQMC is a dataset for evaluating the performance of models in semantic matching tasks. It contains pairs of questions in

Chinese and their matching status, making it a valuable resource for research in Chinese language understanding.

5.2.3 *Story Cloze and Sentence Completion.*

StoryCloze [337]. It introduces a new "StoryCloze Test," a CR framework for evaluating story understanding, generation, and script learning. It considers a model's ability to understand and generate coherent and sensible stories.

LAMBADA [338]. This dataset evaluates contextual text understanding through a word prediction task. Models must predict the last word of a passage, which is easy for humans when given the whole passage, but not when given only the last sentence.

5.2.4 *Physical Knowledge and World Understanding.*

PIQA [343]. A dataset that probes the physical knowledge of models, aiming to understand how well they are learning about the real world.

TriviaQA [344]. A dataset that tests models on RC and open-domain QA tasks, with a focus on IR-style QA.

ARC [345]. A larger version of the ARC-Challenge, this dataset contains both easy and challenging grade-school level, multiple-choice science questions. It is a comprehensive test of a model's ability to understand and answer complex questions.

ARC-Easy [345]. A subset of the ARC dataset, ARC-Easy, contains questions that are answered correctly by either a retrieval-based algorithm or a word co-occurrence algorithm. It is a great starting point for models beginning to explore advanced QA.

ARC-Challenge [345]. A rigorous QA dataset, ARC-Challenge includes complex, grade-school level questions that demand reasoning beyond simple retrieval, testing the true comprehension capabilities of models.

5.2.5 *Contextual Language Understanding.*

RACE [350]. The RACE dataset is a RC dataset collected from English examinations in China, which benchmarks AI models for understanding and answering questions on long and complex passages, simulating the challenge of a real-world examination.

RACE-Middle [350]. Another subset of the RACE [350] dataset, RACE-Middle, contains middle school-level English exam questions. It offers a slightly less challenging but academically oriented evaluation of a model's comprehension skills.

RACE-High [350]. A subset of the RACE [350] dataset, RACE-High consists of high school-level English exam questions. It is designed to evaluate the comprehension ability of models in a more academic and challenging context.

QuAC [351]. This dataset simulates an information-seeking dialog between students and teachers using hidden Wikipedia text. It introduces unique challenges not found in machine comprehension datasets, making it a valuable resource for advancing dialog systems.

5.2.6 *CR.*

HellaSwag [358]. A dataset that challenges models to pick the best ending to a context uses Adversarial Filtering to create a "Goldilocks" zone of complexity, where generated text is absurd to humans but often misclassified by models.

COPA [404]. This dataset evaluates a model's progress in open-domain commonsense causal reasoning. Each question comprises a premise and two alternatives, and the model must select the more plausible alternative, testing a model's ability to understand and reason about cause and effect.

WSC [360]. The WSC is a RC task in which a system must resolve references in a text, often requiring world knowledge and reasoning about the text.

CSQA [361]. The CommonsenseQA is a QA dataset that requires commonsense knowledge to evaluate the ability of AI models to understand and answer questions.

5.2.7 RC.

BoolQ [366]. A dataset derived from Google search queries, BoolQ challenges models to answer binary (yes/no) questions. The questions are naturally occurring and are paired with a paragraph from a Wikipedia article containing the answer. It is a test of RC and reasoning.

SQUADv2 [367]. The **Stanford question answering dataset (SQuAD)** [365] is a collection of questions posed by crowd workers on a set of Wikipedia articles, where the answer to every question is a segment of text from the corresponding reading passage. SQuADv2 combines the original SQuAD1.1 dataset with over 50,000 unanswerable questions. The aim is to evaluate a model's ability to understand and answer questions based on a given context and to determine when a question is unanswerable.

Discrete Reasoning Over the Content of Paragraphs (DROP) [368]. DROP is designed to test a model's ability to understand a wide variety of reading phenomena. It encourages comprehensive and reliable evaluation of RC capabilities.

Recognizing Textual Entailment (RTE) [369]. The RTE datasets come from a series of annual competitions on textual entailment, predicting whether a given sentence logically follows from another and evaluating a model's understanding of logical relationships in a text.

WebQA [370]. A dataset for open-domain QA, WebQA offers a large collection of web-based question answer pairs. It is designed to assess the ability of AI models to understand and answer questions based on web content.

CMRC2018 [372]. This dataset is a test of Chinese language models' ability to reason comprehensively and is designed with a challenging span-extraction format that pushes the boundaries of machine performance.

5.2.8 MR.

MATH [385]. This dataset is a platform for evaluating the mathematical problem-solving abilities of AI models. It contains a diverse set of math problems, ranging from arithmetic to calculus, and is designed to test the model's ability to understand and solve complex mathematical problems.

Math23k [386]. This one challenges a model's ability to understand and solve mathematical word problems. It contains 23,000 Chinese arithmetic word problems that require models to perform reasoning and computation based on the problem description.

GSM8K [387]. A dataset of diverse grade-school math word problems, testing a model's ability to perform multi-step MR.

5.2.9 Problem-Solving and Logical Reasoning.

ANLI [396]. A large-scale dataset designed to test the robustness of machine learning models in NLI is created through an iterative, adversarial process where humans try to generate examples that models cannot correctly classify.

HumanEval [145]. A dataset for evaluating the problem-solving ability of AI models, which includes a diverse set of tasks that require various cognitive abilities, making it a comprehensive tool for assessing general intelligence in AI.

StrategyQA [352]. A QA dataset that requires reasoning over multiple pieces of evidence to evaluate the strategic reasoning ability of AI models, pushing the boundaries of what machines can understand and answer.

5.2.10 Cross-Lingual Understanding.

XNLI [401]. A cross-lingual benchmark, XNLI extends the MultiNLI [432] corpus to 15 languages, including low-resource ones like Urdu. It tests models on cross-lingual sentence understanding, with 112,500 annotated pairs across three categories: entailment, contradiction, and neutral.

Cross-lingual Paraphrase Adversaries from Word Scrambling (PAWS-X) [402]. PAWS-X is a multilingual version of the PAWS [433] dataset for paraphrase identification. It includes examples in seven languages and is designed to evaluate the performance of cross-lingual paraphrase identification models.

5.2.11 *Truthfulness.*

Truthful-QA [408]. A unique benchmark that measures a language model's truthfulness when generating answers. The dataset includes questions across various categories like health, law, and politics, some designed to test the model against common human misconceptions.

5.2.12 *Biases and Ethics in AI.*

ETHOS [411]. ETHOS is a hate speech detection dataset built from YouTube and Reddit comments. It is a tool in the fight against online hate speech, offering binary and multi-label variants for robust content moderation.

StereoSet [412]. StereoSet is a comprehensive dataset designed to measure and evaluate the presence of stereotypical biases in language models. It focuses on four key domains: gender, profession, race, and religion. Contrasting stereotypical bias against LM ability provides a valuable tool for understanding and mitigating biases in LLMs.

6 Applications

Applying LLMs to a variety of downstream tasks has become a popular trend in both AI-related research communities and industries, with many emerging uses being discovered and explored daily. LLMs, which are capable of understanding and generating human-like text, have found meaningful applications across a variety of fields. This section provides an overview of LLM applications in medicine, education, science, mathematics, law, finance, robotics, and coding. While each of these domains poses different challenges, LLMs open up opportunities to make significant contributions to these domains through their generalizability.

General Purpose. LLMs are being widely considered as general-purpose tools for a wide variety of tasks [434]. This is due to their inherent ability to understand, generate, and manipulate human-like text in a contextually relevant manner. This allows them to perform tasks ranging from simple language translation and QA to more complex tasks like summarization, text generation, and even programming help [435]. The utility of LLMs is further enhanced by their ability to adapt to the specific style and tone of the text they are processing, making the outputs more user-friendly and context-aware. In everyday applications, LLMs can be used as personal assistants, helping users draft e-mails or schedule appointments [436]; they can also be deployed in customer service to handle common questions or applied to generate content for digital platforms like Web sites by creating human-like text based on given prompts [437]. Moreover, LLMs play a crucial role in data analysis, where they can filter large volumes of text data, summarize key points, and find patterns that would take humans much longer to identify [438]. Despite their wide-ranging applications, it is essential to remember that LLMs, similar to any AI system, are only as good as the data they have been trained on.

Medicine. The application of LLMs in the field of medicine is reshaping healthcare delivery and research. For example, LLMs are increasingly used in clinical decision support systems to provide physicians with evidence-based treatment recommendations [439–441]. By analyzing patient data and medical literature, they can help identify potential diagnoses, suggest appropriate tests, and recommend optimal treatment strategies. Moreover, LLMs can also enhance patient interactions with healthcare systems; e.g., they can be used in chatbot applications [442–444] to answer patient queries about symptoms or medications, schedule appointments, and even provide essential health advice. For medical research, LLMs are used to extract and filter information from a considerable

amount of medical literature, identify relevant studies, summarize findings, and even predict future research trends [445–447]. For medical education, LLMs can help create training materials, generate exam questions, provide detailed explanations of complex medical topics, and offer personalized feedback to students [448–451]. They can also simulate patient interactions, enabling students to practice and improve their clinical skills. At a broader level, LLMs can assist in public health initiatives by analyzing media data to detect disease outbreaks, monitor public sentiment toward health policies, and disseminate health information in a clear and understandable manner [452]. LLMs can be employed to support public health initiatives, addressing related issues such as data privacy, the necessity for explainability, and the potential risk of propagating biases [453, 454].

Education. The integration of LLMs into the educational sector offers opportunities to enhance learning experiences, teacher support, and educational content development. For students, by analyzing their learning styles, performance, and preferences, LLMs can provide customized study materials and practice questions to develop personalized learning experiences [455]. For teachers, LLMs can help to create lesson plans and grade assignments and generate diverse and inclusive educational content, significantly saving more time for teaching and student interaction [456, 457]. In language learning, LLMs serve as advanced conversational partners capable of simulating conversations in multiple languages, correcting grammar, enhancing vocabulary, and aiding pronunciation for the needs of fluency in practice [458]. Furthermore, LLMs improve accessibility in education by providing support for students with disabilities. They can generate real-time transcriptions for the hearing impaired, offer reading assistance for the visually impaired, and simplify complex texts for those with learning disabilities [454]. As LLMs continue to evolve, their applications in education can benefit more students and teachers from different perspectives in practice.

Science. Similar to medical applications, LLMs can expedite the research process by quickly analyzing and summarizing scientific literature. By briefing comprehensible and accessible research summaries, LLMs can assist researchers in staying up-to-date with the latest findings, even in fields outside their area of expertise [459, 460]. In addition, LLMs can aid scientists in formulating new hypotheses and research questions since their ability to process large-scale datasets allows them to unveil insights that might not be immediately apparent to human researchers [461]. Moreover, for scientific writing, LLMs can help researchers draft documents, suggest improvements, and ensure adherence to specific formatting guidelines [462, 463]. This not only saves time but also improves the clarity of scientific communication, enabling interdisciplinary teams to work together more effectively.

Maths. In addition to providing mathematical research and education support, LLMs can assist in solving mathematical problems by giving step-by-step explanations and guiding users through complex proofs and calculations. They can help identify errors in reasoning or computation and suggest corrections, serving as an invaluable tool for both learning and verification purposes [464, 465]. LLMs can be employed to check the validity of mathematical proofs, offering a preliminary filter before human review. While they are not a substitute for the meticulous work of mathematicians, they can help simplify the process of proof verification [466, 467]. Moreover, LLMs enhance accessibility to mathematics by translating complex concepts and findings into understandable language for non-specialists [468], where the gap between theoretical mathematics and applied contexts such as physics, engineering, and economics can be bridged.

Law. LLMs can assist with the thematic analysis of legal documents, including generating initial coding for datasets, identifying themes, and classifying data according to these themes. This collaborative effort between legal experts and LLMs has proved to be effective in analyzing legal texts such as court opinions on theft, improving both the efficiency and quality of the research [469]. Additionally, LLMs have been evaluated for their ability to generate explanations of legal terms, focusing on improving factual accuracy and relevance by incorporating sentences from case law. By feeding

relevant case law into the LLM, the augmented models can generate higher-quality explanations with less factually incorrect information [470]. Moreover, LLMs can be trained with specialized domain knowledge to perform legal reasoning tasks [471] and answer legal questions [472].

Finance. LLMs like BloombergGPT [155], trained on extensive proprietary financial datasets, exhibit superior performance on financial tasks. This indicates the value of domain-specific training in creating LLMs that can more accurately understand and process industry-specific language and concepts. The introduction of FinGPT [473] as an open-source model offers transparent and accessible resources to develop novel applications such as robo-advising, algorithmic trading, and low-code solutions, ultimately expanding the capabilities of financial services. Both BloombergGPT and FinGPT show the adaptability of LLMs to the financial domain, with the former showing the power of custom datasets and the latter emphasizing a data-centric approach and low-rank adaptation techniques for customization. Moreover, LLMs demonstrate an ability to break down complex financial tasks into actionable plans, enabling end-to-end solutions that were previously unfeasible with a single model [474].

Robotics. In robotics research, LLMs have promising applications, such as enhancing human-robot interaction [28, 475–477], task planning [240], motion planning [249], navigation [249, 478], object manipulation [239], personalized robots [479], and so on. LLMs enable robots to understand the environment effectively and generate plans to complete tasks collaboratively [26, 243]. They can facilitate continuous learning by allowing robots to access and integrate information from a wide range of sources, helping robots acquire new skills, adapt to changes, and refine their paths [227, 236, 237].

7 Deployment

Bias, privacy, hallucination, security, adversarial attacks, multilingual support, and helpful response are a few of the issues that need to be fixed before deploying LLMs in production. Much research has been carried out in these areas [11, 178, 182, 309, 480], which requires training or fine-tuning of the model. However, serving LLMs for inference comes with additional challenges, such as power management, real-time response, hardware resources, and hardware failures. Researchers have explored this direction of LLMs to find optimal strategies for serving LLMs on both cloud infrastructure and edge devices.

Deployment on Cloud. Cloud infrastructure is divided into homogeneous and heterogeneous types. Homogeneous hardware has the same type of GPUs across nodes, whereas heterogeneous hardware has different types of GPUs with varying memory, computing capacity, and network bandwidth, making deployment harder. Implementing model and pipeline parallelism is easier for homogeneous hardware; however, simply implementing these parallelism approaches for distributed LLM inference is naive. The deployment of LLMs on homogeneous hardware can be further optimized in memory, throughput, and latency space by efficiently managing the KV cache [481], scheduling prefill and decode phases [482, 483], iterative scheduling, and selective batching [483]. On the other hand, the workload in the real world is not consistent, and the heterogeneous cluster has been shown to benefit from it, where optimal task scheduling [484–486] achieves better throughput and latency against homogeneous hardware in the same price budget.

Deployment on Edge. Edge devices have limited memory and computing power compared to cloud hardware. Deploying LLMs on edge devices with billions of parameters is not possible. However, researchers have suggested alternative ways to deploy LLMs by shrinking model size through quantization [44, 45, 258, 487], knowledge distillation [487], and pruning [42, 43, 487]; introducing collaborative edge computing [488]; designing efficient accelerators for edge devices [489]; and training smaller LLMs [480].

8 Challenges and Future Directions

LLMs such as GPT-4 and its predecessors have significantly advanced NLP. Nevertheless, they also bring along a set of challenges. The computational cost, adversarial robustness, and interpretability are among the technical challenges that are intrinsic to these models. Furthermore, as these models are scaled up to handle more complex tasks or to operate in more complex or dynamic environments, new challenges in scalability, privacy, and real-time processing emerge. On the frontier of foundational research, integrating multimodality and the effectiveness of transfer learning are being keenly explored. Additionally, the continuous learning aspect of these models, which aims to have models that can adapt to new information over time, presents a fresh set of challenges. These challenges not only underscore the technical intricacies involved but also highlight the broader impact and the future trajectory of LLMs in real-world applications. The following sections delve into these challenges, shedding light on the ongoing and potential efforts to address them.

Computational Cost. Training LLMs requires extensive computational resources, which increases production costs and raises environmental concerns due to substantial energy consumption during large-scale training [490]. Hence, it is important to put more research effort into developing efficient models such as DeepSeekv3 [143], suggesting MLA, KV caching, multi-token prediction, FP-8 precision training, and so on, to minimize training costs.

Bias and Fairness. LLMs can inherit and amplify societal biases in their training data. These biases can manifest in the model's outputs, leading to potential ethical and fairness issues [491]. LLMs are aligned to show no bias and be fair in their output [6, 309], but red-teaming [181, 182] identified that LLMs can be provoked to produce undesired responses.

Overfitting. Although LLMs possess substantial learning capabilities, they are susceptible to overfitting noisy and peculiar patterns within their extensive training data. Consequently, this may cause them to generate illogical responses [492]. The debate about memorization vs. generalization in LLMs is about finding the right balance. Memorization allows the model to remember specific details from its training data, ensuring it can provide accurate answers to precise questions. However, generalization enables the model to make inferences and produce responses for inputs it has not seen before, which is essential for handling various real-world tasks. Striking the right balance is the challenge: too much memorization can lead to overfitting, making the model inflexible and struggling with new inputs [493].

Economic and Research Inequality. The high cost of training and deploying LLMs may make their development concentrated within well-funded organizations, potentially worsening economic, and research inequalities in AI [494]. New architectures and algorithms should be developed that reduce the cost of training and deploying LLMs.

Reasoning and Planning. Some reasoning and planning tasks, even as seemingly simple as commonsense planning, which humans find easy, remain well beyond the current capabilities of LLMs evaluated using an assessment framework. This is not entirely unexpected, considering that LLMs primarily generate text completions based on likelihood and offer no solid guarantees in terms of reasoning abilities [495].

Hallucinations. LLMs exhibit "hallucinations," where they generate responses that, while sounding plausible, are incorrect or do not align with the provided information [496]. Hallucinations can be categorized into three categories.

- —Input-conflicting hallucination, wherein LLMs produce content that diverges from the input given by users.
- —Context-conflicting hallucination, where LLMs generate content that contradicts information they have generated earlier.

—Fact-conflicting hallucination involves LLM's generation of content that does not align with established world knowledge.

Augmented LLMs [169, 194] reduce hallucinations but require collecting facts and context from the database, slowing response times.

Prompt Engineering. Prompts serve as inputs to LLMs, and their syntax and semantics play a crucial role in determining the model's output. The prompt variations, sometimes counter-intuitive to humans, can result in significant changes in model output and are addressed through prompt engineering, which involves designing natural language queries to guide LLMs responses effectively [32, 497].

Limited Knowledge. Information acquired during pre-training is limited and may become obsolete after some time. Re-training the model using updated data is costly. To generate factually accurate responses, people use a retrieval augmentation pipeline [201]. However, pre-trained models are not trained with RAG [6, 21]; hence, adapting the training pipeline is necessary [25, 196]. This requires suggesting better training and utilization methods in the future.

Model Safety. Although LLMs are aligned for safety, they are prone to generating harmful, fake, misleading, and inappropriate content, leaking private information, and misuse. Prompt injection [498], jailbreaking [180], and adversarial attacks [499, 500] can easily bypass LLMs security. Red-teaming [181, 182] identifies these vulnerabilities to align LLMs for better safety. Designing better defense mechanisms to ensure LLMs are safe, reliable, and trustworthy for complex AI applications demands more research in this area [501, 502].

Multimodality. Multimodal learning, where LLMs are trained on diverse data like text, images, and videos, aims to create models with richer understanding but faces challenges in data alignment, fusion strategies, and higher computational demands.

Catastrophic Forgetting. LLMs are often pre-trained on large datasets and then fine-tuned on domain-specific data, reducing training resources. However, they face issues like domain adaptation and catastrophic forgetting, which hinder retaining original knowledge when learning new tasks. Continual learning should be explored further to avoid catastrophic forgetting.

Adversarial Robustness. LLMs have shown great capabilities in various tasks but are vulnerable to adversarial attacks, where slight, deliberate input alterations can mislead them. Especially with models like BERT, adversarial fine-tuning can enhance robustness, although it sometimes compromises generalization [499]. As LLMs integrate more into complex systems, examining their security properties becomes crucial, given the emerging field of adversarial attacks on LLMs within trustworthy ML [503]. This vulnerability is notable in safety-critical domains, necessitating robust training methods and adversarial evaluation tools to ensure LLM reliability [500].

Interpretability and Explainability. The "black-box" nature of LLMs poses challenges in understanding their decision-making, which is crucial for broader acceptance and trust, especially in sensitive domains. Despite their advanced capabilities, the lack of insight into their operation limits their effectiveness and trustworthiness [504, 505]. Efforts are being made to make LLMs more explainable to promote user trust and ensure responsible AI usage. Understanding the logic behind LLMs' responses is essential for fostering trust and ensuring they align with human values and legal standards.

Privacy Concerns. Privacy concerns in LLMs have escalated with their growth in complexity and size, particularly around data sharing and potential misuse. There is a risk of malicious content creation, filter bypass, and data privacy issues, especially in e-commerce, where protecting customer privacy is crucial. If models are trained on private data, additional concerns arise if such models are made publicly available. LLMs tend to memorize phrases from their training sets, which an adversary could exploit to extract sensitive data, posing a threat to personal privacy [506, 507].

Real-Time Processing. Real-time processing in LLMs is pivotal for various applications, especially with the rising popularity of mobile AI applications and concerns regarding information security and privacy. However, LLMs often have hundreds of layers and billions of parameters, which impede real-time processing due to the high computational demands and limited weight storage on hardware platforms, particularly in edge computing environments [508]. MobileLLM [480] suggested smaller LLMs with million-scale parameters optimized for mobile devices with better architectural design. More development in this area is needed to employ LLMs in smaller devices.

Long-Term Dependencies. LLMs have shown considerable progress in understanding and generating text, yet they often struggle with preserving context and handling long-term dependencies, particularly in complex, multi-turn conversations or long documents. This limitation can lead to incoherent or irrelevant responses. Research efforts like LongT5 [48] and LongLoRA [192] could help LLMs understand the longer context.

Hardware Acceleration. The growth of LLMs presents significant hardware challenges due to the increasing computational and memory demands associated with training and deploying these models. GPUs have played a crucial role in meeting the hardware requirements for training LLMs, with the networking industry also evolving to optimize hardware for training workloads. However, the growing size of LLMs, which has been outpacing hardware progress, makes model inference increasingly costly. Model quantization is a promising approach to bridge the widening gap between LLM size and hardware capacity [509]. Although specialized hardware acceleration like GPUs or TPUs can significantly reduce the computational cost, making real-time applications more feasible, they may not fully resolve all limitations, necessitating further advancements in hardware technology.

Regulatory and Ethical Frameworks. The rapid advancements in AI have given rise to sophisticated LLMs like OpenAI's GPT-4 [108] and Google's Gemini. These developments underscore the imperative for regulatory oversight to manage the ethical and social challenges accompanying LLMs' widespread use [510]. For instance, LLMs can generate content that can be used positively or negatively, emphasizing the need for proactive ethical frameworks and policy measures to guide their responsible use and assign accountability for their outputs [511]. Auditing is identified as a promising governance mechanism to ensure that AI systems, including LLMs, are designed and deployed ethically, legally, and technically robust [512].

Energy Efficiency. LLMs are gigantic, requiring massive computation and hardware resources for training and inference. This demands high electricity requirements and contributes to environmental pollution with CO_2 emissions. With such large requirements, research in LLMs is mostly limited to big AI companies. To democratize LLMs research and utilization, we need to develop efficient transformer architectures and algorithms such as an MoE, sparse attention mechanisms, quantization, pruning, knowledge distillation, low-powered GPUs, and efficient accelerators [42, 45, 133, 142, 143, 513].

Multilingual Capabilities. LLMs have excellent English capabilities but suffer significantly in other languages. Although the training data used in the LLaMA [133] and GPT models [6, 108] are multilingual, the coverage of other languages is limited compared to English [11]. Furthermore, the evaluation benchmarks are more centered on English [2, 310, 311] than multilingual [400, 401]. Hence, to improve multilingual capabilities, we need to prepare datasets that cover multiple languages equally proportioned to English.

9 Conclusion

This article has comprehensively reviewed the developments in LLMs. It summarizes the significant findings of LLMs in the existing literature and provides a detailed analysis of the design aspects, including architectures, datasets, and training pipelines. We identified crucial architectural components and training strategies employed in LLMs. These aspects are presented as summaries and

discussions throughout the article. We have discussed the performance differences of LLMs in zero-shot and few-shot settings, explored the impact of fine-tuning, and compared supervised and generalized models and encoder vs. decoder vs. encoder-decoder architectures. A comprehensive review of MLLMs, retrieval-augmented LLMs, LLMs-powered agents, efficient LLMs, datasets, evaluation, applications, and challenges is also provided. This article is anticipated to serve as a valuable resource for researchers, offering insights into the recent advancements in LLMs and providing fundamental concepts and details to develop better LLMs.

References

[1] A. Chernyavskiy, D. Ilvovsky, and P. Nakov. 2021. Transformers: "The end of history" for natural language processing? In *Proceedings of the Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference (ECML PKDD '21)*. Springer, 677–693.

[2] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. 2019. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 32

[3] D. Adiwardana, M.-T. Luong, D. R. So, J. Hall, N. Fiedel, R. Thoppilan, Z. Yang, A. Kulshreshtha, G. Nemade, Y. Lu, et al. 2020. Towards a human-like open-domain chatbot. arXiv:2001.09977. Retrieved from https://arxiv.org/abs/2001.09977

[4] B. A. Y. Arcas. 2022. Do large language models understand us? *Daedalus* 151, 2 (2022), 183–197.

[5] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog* 1, 8 (2019), 9.

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. 2020. Language models are few-shot learners. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 33, 1877–1901.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805. Retrieved from https://arxiv.org/abs/1810.04805

[8] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2227–2237.

[9] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. 2019. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv:1910.13461. Retrieved from https://arxiv.org/abs/1910.13461

[10] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research* 21, 1 (2020), 5485–5551.

[11] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel. 2020. MT5: A massively multilingual pre-trained text-to-text transformer. arXiv:2010.11934. Retrieved from https://arxiv.org/abs/2010.11934

[12] Z. Zhang, Y. Gu, X. Han, S. Chen, C. Xiao, Z. Sun, Y. Yao, F. Qi, J. Guan, P. Ke, et al. 2021. CPM-2: Large-scale cost-effective pre-trained language models. *AI Open* 2 (2021), 216–224.

[13] T. L. Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilić, D. Hesslow, R. Castagné, A. S. Luccioni, F. Yvon, M. Gallé, et al. 2022. BLOOM: A 176B-parameter open-access multilingual language model. arXiv:2211.05100. Retrieved from https://arxiv.org/abs/2211.05100

[14] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, et al. 2022. OPT: Open pre-trained transformer language models. arXiv:2205.01068. Retrieved from https://arxiv.org/abs/2205.01068

[15] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. 2022. PaLM: Scaling language modeling with pathways. arXiv:2204.02311. Retrieved from https://arxiv.org/abs/2204.02311

[16] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma, et al. 2022. Scaling instruction-finetuned language models. arXiv:2210.11416. Retrieved from https://arxiv.org/abs/2210.11416

[17] V. Sanh, A. Webson, C. Raffel, S. H. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, T. L. Scao, A. Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. arXiv:2110.08207. Retrieved from https://arxiv.org/abs/2110.08207

[18] Y. Wang, S. Mishra, P. Alipoormolabashi, Y. Kordi, A. Mirzaei, A. Naik, A. Ashok, A. S. Dhanasekaran, A. Arunkumar, D. Stap, et al. 2022. Super-natural instructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 5085–5109.

[19] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi. 2022. Self-instruct: Aligning language model with self generated instructions. arXiv:2212.10560. Retrieved from https://arxiv.org/abs/2212.10560

[20] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. 2022. Training language models to follow instructions with human feedback. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 35, 27730–27744.

[21] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. arXiv:2307.09288. Retrieved from https://arxiv.org/abs/2307.09288

[22] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, et al. 2022. Emergent abilities of large language models. arXiv:2206.07682. Retrieved from https://arxiv.org/abs/2206.07682

[23] T. Webb, K. J. Holyoak, and H. Lu. 2023. Emergent analogical reasoning in large language models. *Nature Human Behaviour* 7, 9 (2023), 1526–1541.

[24] D. A. Boiko, R. MacKnight, and G. Gomes. 2023. Emergent autonomous scientific research capabilities of large language models. arXiv:2304.05332. Retrieved from https://arxiv.org/abs/2304.05332

[25] G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin, S. Riedel, and E. Grave. 2022. Few-shot learning with retrieval augmented language models. arXiv:2208.03299. Retrieved from https://arxiv.org/abs/2208.03299

[26] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. 2023. PaLM-E: An embodied multimodal language model. arXiv:2303.03378. Retrieved from https://arxiv.org/abs/2303.03378

[27] A. Parisi, Y. Zhao, and N. Fiedel. 2022. TALM: Tool augmented language models. arXiv:2205.12255. Retrieved from https://arxiv.org/abs/2205.12255

[28] B. Zhang and H. Soh. 2023. Large language models as zero-shot human models for human-robot interaction. arXiv:2303.03548. Retrieved from https://arxiv.org/abs/2303.03548

[29] Q. Ye, H. Xu, G. Xu, J. Ye, M. Yan, Y. Zhou, J. Wang, A. Hu, P. Shi, Y. Shi, et al. 2023. mPLUG-Owl: Modularization empowers large language models with multimodality. arXiv:2304.14178. Retrieved from https://arxiv.org/abs/2304.14178

[30] W. Wang, Z. Chen, X. Chen, J. Wu, X. Zhu, G. Zeng, P. Luo, T. Lu, J. Zhou, Y. Qiao, et al. 2023. VisionLLM: Large language model is also an open-ended decoder for vision-centric tasks. arXiv:2305.11175. Retrieved from https://arxiv.org/abs/2305.11175

[31] R. Yang, L. Song, Y. Li, S. Zhao, Y. Ge, X. Li, and Y. Shan. 2023. GPT4Tools: Teaching large language model to use tools via self-instruction. arXiv:2305.18752. Retrieved from https://arxiv.org/abs/2305.18752

[32] E. Saravia. 2022. Prompt Engineering Guide. Retrieved December 2022 from https://github.com/dair-ai/Prompt-Engineering-Guide

[33] A. Zeng, X. Liu, Z. Du, Z. Wang, H. Lai, M. Ding, Z. Yang, Y. Xu, W. Zheng, X. Xia, et al. 2022. GLM-130B: An open bilingual pre-trained model. arXiv:2210.02414. Retrieved from https://arxiv.org/abs/2210.02414

[34] Y. Wang, H. Le, A. D. Gotmare, N. D. Bui, J. Li, and S. C. Hoi. 2023. CodeT5+: Open code large language models for code understanding and generation. arXiv:2305.07922. Retrieved from https://arxiv.org/abs/2305.07922

[35] S. Wang, Y. Sun, Y. Xiang, Z. Wu, S. Ding, W. Gong, S. Feng, J. Shang, Y. Zhao, C. Pang, et al. 2021. ERNIE 3.0 Titan: Exploring larger-scale knowledge enhanced pre-training for language understanding and generation. arXiv:2112.12731. Retrieved from https://arxiv.org/abs/2112.12731

[36] J. Rasley, S. Rajbhandari, O. Ruwase, and Y. He. 2020. Deep speed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 3505–3506.

[37] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He. 2020. ZeRO: Memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '20)*. IEEE, 1–16.

[38] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. arXiv:2110.04366. Retrieved from https://arxiv.org/abs/2110.04366

[39] Z. Hu, Y. Lan, L. Wang, W. Xu, E.-P. Lim, R. K.-W. Lee, L. Bing, and S. Poria. 2023. LLM-Adapters: An adapter family for parameter-efficient fine-tuning of large language models. arXiv:2304.01933. Retrieved from https://arxiv.org/abs/2304.01933

[40] B. Lester, R. Al-Rfou, and N. Constant. 2021. The power of scale for parameter-efficient prompt tuning. arXiv:2104.08691. Retrieved from https://arxiv.org/abs/2104.08691

[41] X. L. Li and P. Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. arXiv:2101.00190. Retrieved from https://arxiv.org/abs/2101.00190

[42] X. Ma, G. Fang, and X. Wang. 2023. LLM-Pruner: On the structural pruning of large language models. arXiv:2305.11627. Retrieved from https://arxiv.org/abs/2305.11627

[43] R. Xu, F. Luo, C. Wang, B. Chang, J. Huang, S. Huang, and F. Huang. 2022. From dense to sparse: Contrastive pruning for better pre-trained language model compression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36, 11547–11555.

[44] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han. 2023. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the International Conference on Machine Learning*, Vol. 202, PMLR, 38087–38099.

[45] C. Tao, L. Hou, W. Zhang, L. Shang, X. Jiang, Q. Liu, P. Luo, and N. Wong. 2022. Compression of generative pre-trained language models via quantization. arXiv:2203.10705. Retrieved from https://arxiv.org/abs/2203.10705

[46] A. Pal, D. Karkhanis, M. Roberts, S. Dooley, A. Sundararajan, and S. Naidu. 2023. Giraffe: Adventures in expanding context lengths in LLMS. arXiv:2308.10882. Retrieved from https://arxiv.org/abs/2308.10882

[47] B. Peng, J. Quesnelle, H. Fan, and E. Shippole. 2023. YaRN: Efficient context window extension of large language models. arXiv:2309.00071. Retrieved from https://arxiv.org/abs/2309.00071

[48] M. Guo, J. Ainslie, D. Uthus, S. Ontanon, J. Ni, Y.-H. Sung, and Y. Yang. 2021. LongT5: Efficient text-to-text transformer for long sequences. arXiv:2112.07916. Retrieved from https://arxiv.org/abs/2112.07916

[49] S. Chen, S. Wong, L. Chen, and Y. Tian. 2023. Extending context window of large language models via positional interpolation. arXiv:2306.15595. Retrieved from https://arxiv.org/abs/2306.15595

[50] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al. 2023. A survey of large language models. arXiv:2303.18223. Retrieved from https://arxiv.org/abs/2303.18223

[51] U. Naseem, I. Razzak, S. K. Khan, and M. Prasad. 2021. A comprehensive survey on word representation models: From classical to state-of-the-art word representation language models. *Transactions on Asian and Low-Resource Language Information Processing* 20, 5 (2021), 1–35.

[52] B. Min, H. Ross, E. Sulem, A. P. B. Veyseh, T. H. Nguyen, O. Sainz, E. Agirre, I. Heinz, and D. Roth. 2021. Recent advances in natural language processing via large pre-trained language models: A survey. arXiv:2111.01243. Retrieved from https://arxiv.org/abs/2111.01243

[53] C. Zhou, Q. Li, C. Li, J. Yu, Y. Liu, G. Wang, K. Zhang, C. Ji, Q. Yan, L. He, et al. 2023. A comprehensive survey on pretrained foundation models: A history from BERT to ChatGPT. arXiv:2302.09419. Retrieved from https://arxiv.org/abs/2302.09419

[54] Q. Dong, L. Li, D. Dai, C. Zheng, Z. Wu, B. Chang, X. Sun, J. Xu, and Z. Sui. 2022. A survey for in-context learning. arXiv:2301.00234. Retrieved from https://arxiv.org/abs/2301.00234

[55] J. Huang and K. C.-C. Chang. 2022. Towards reasoning in large language models: A survey. arXiv:2212.10403. Retrieved from https://arxiv.org/abs/2212.10403

[56] Y. Wang, W. Zhong, L. Li, F. Mi, X. Zeng, W. Huang, L. Shang, X. Jiang, and Q. Liu. 2023. Aligning large language models with human: A survey. arXiv:2307.12966. Retrieved from https://arxiv.org/abs/2307.12966

[57] X. Zhu, J. Li, Y. Liu, C. Ma, and W. Wang. 2023. A survey on model compression for large language models. arXiv:2308.07633. Retrieved from https://arxiv.org/abs/2308.07633

[58] S. Yin, C. Fu, S. Zhao, K. Li, X. Sun, T. Xu, and E. Chen. 2023. A survey on multimodal large language models. arXiv:2306.13549. Retrieved from https://arxiv.org/abs/2306.13549

[59] J. J. Webster and C. Kit. 1992. Tokenization as the initial phase in NLP. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING '92)*, Vol. 4.

[60] T. Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Long Papers*, Vol. 1, 66–75.

[61] R. Sennrich, B. Haddow, and A. Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Long Papers*, Vol. 1, 1715–1725.

[62] M. Schuster and K. Nakajima. 2012. Japanese and Korean voice search. In *Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5149–5152.

[63] S. J. Mielke, Z. Alyafeai, E. Salesky, C. Raffel, M. Dey, M. Gallé, A. Raja, C. Si, W. Y. Lee, B. Sagot, et al. 2021. Between words and characters: A brief history of open-vocabulary modeling and tokenization in NLP. arXiv:2112.10508. Retrieved from https://arxiv.org/abs/2112.10508

[64] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. 2017. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 30.

[65] O. Press, N. Smith, and M. Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *Proceedings of the International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=R8sQPpGCv0

[66] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu. 2021. RoFormer: Enhanced transformer with rotary position embedding. arXiv:2104.09864. Retrieved from https://arxiv.org/abs/2104.09864

[67] R. Child, S. Gray, A. Radford, and I. Sutskever. 2019. Generating long sequences with sparse transformers. arXiv:1904.10509. Retrieved from https://arxiv.org/abs/1904.10509

[68] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 35, 16344–16359.

[69] K. Hornik, M. Stinchcombe, and H. White. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* 2, 5 (1989), 359–366.
[70] V. Nair and G. E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 807–814.
[71] D. Hendrycks and K. Gimpel. 2016. Gaussian error linear units (GELUs). arXiv:1606.08415. Retrieved from https://arxiv.org/abs/1606.08415
[72] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting, *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.
[73] D. Krueger, T. Maharaj, J. Kramár, M. Pezeshki, N. Ballas, N. R. Ke, A. Goyal, Y. Bengio, A. Courville, and C. Pal. 2016. Zoneout: Regularizing RNNS by randomly preserving hidden activations. arXiv:1606.01305. Retrieved from https://arxiv.org/abs/1606.01305
[74] N. Shazeer. 2020. GLU variants improve transformer. arXiv:2002.05202. Retrieved from https://arxiv.org/abs/2002.05202
[75] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the International Conference on Machine Learning*. PMLR, 933–941.
[76] J. L. Ba, J. R. Kiros, and G. E. Hinton. 2016. Layer normalization. arXiv:1607.06450. Retrieved from https://arxiv.org/abs/1607.06450
[77] B. Zhang and R. Sennrich. 2019. Root mean square layer normalization. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 32.
[78] A. Baevski and M. Auli. 2018. Adaptive input representations for neural language modeling. arXiv:1809.10853. Retrieved from https://arxiv.org/abs/1809.10853
[79] H. Wang, S. Ma, L. Dong, S. Huang, D. Zhang, and F. Wei. 2022. DeepNet: Scaling transformers to 1,000 layers. arXiv:2203.00555. Retrieved from https://arxiv.org/abs/2203.00555
[80] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro. 2019. Megatron-LM: Training multi-billion parameter language models using model parallelism. arXiv:1909.08053. Retrieved from https://arxiv.org/abs/1909.08053
[81] BMTrain. 2025. BMTrain: Efficient Training for Big Models. Retrieved from https://github.com/OpenBMB/BMTrain
[82] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45.
[83] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, et al. 2018. JAX: Composable transformations of Python+ Numpy programs (2018).
[84] S. Li, J. Fang, Z. Bian, H. Liu, Y. Liu, H. Huang, B. Wang, and Y. You. 2021. Colossal-AI: A unified deep learning system for large-scale parallel training. arXiv:2110.14883. Retrieved from https://arxiv.org/abs/2110.14883
[85] J. He, J. Qiu, A. Zeng, Z. Yang, J. Zhai, and J. Tang. 2021. FastMoE: A fast mixture-of-expert training system. arXiv:2103.13262. Retrieved from https://arxiv.org/abs/2103.13262
[86] Huawei Technologies Co., Ltd. 2022. Huawei MindSpore AI development framework. In *Artificial Intelligence Technology*. Springer, 137–162.
[87] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 32.
[88] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. 2016. TensorFlow: A system for large-scale machine learning. In *Proceedings of the Operating Systems Design and Implementation*, Vol. 16, 265–283.
[89] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. 2015. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. arXiv:1512.01274. Retrieved from https://arxiv.org/abs/1512.01274
[90] W. Fedus, B. Zoph, and N. Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research* 23, 1 (2022), 5232–5270.
[91] N. Du, Y. Huang, A. M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou, A. W. Yu, O. Firat, et al. 2022. GLaM: Efficient scaling of language models with mixture-of-experts. In *Proceedings of the International Conference on Machine Learning*. PMLR, 5547–5569.
[92] X. Ren, P. Zhou, X. Meng, X. Huang, Y. Wang, W. Wang, P. Li, X. Zhang, A. Podolskiy, G. Arshinov, et al. 2023. Pangu-$\sum$: Towards trillion parameter language model with sparse heterogeneous computing. arXiv:2303.10845. Retrieved from https://arxiv.org/abs/2303.10845
[93] T. Wang, A. Roberts, D. Hesslow, T. Le Scao, H. W. Chung, I. Beltagy, J. Launay, and C. Raffel. 2022. What language model architecture and pretraining objective works best for zero-shot generalization? In *Proceedings of the International Conference on Machine Learning*. PMLR, 22964–22984.

[94] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H.-W. Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 32.

[95] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. 2020. Scaling laws for neural language models. arXiv:2001.08361. Retrieved from https://arxiv.org/abs/2001.08361

[96] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. D. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, et al. 2022. Training compute-optimal large language models. arXiv:2203.15556. Retrieved from https://arxiv.org/abs/2203.15556

[97] S. Iyer, X. V. Lin, R. Pasunuru, T. Mihaylov, D. Simig, P. Yu, K. Shuster, T. Wang, Q. Liu, P. S. Koura, et al. 2022. OPT-IML: Scaling language model instruction meta learning through the lens of generalization. arXiv:2212.12017. Retrieved from https://arxiv.org/abs/2212.12017

[98] Z. Sun, Y. Shen, Q. Zhou, H. Zhang, Z. Chen, D. Cox, Y. Yang, and C. Gan. 2023. Principle-driven self-alignment of language models from scratch with minimal human supervision. arXiv:2305.03047. Retrieved from https://arxiv.org/abs/2305.03047

[99] A. Askell, Y. Bai, A. Chen, D. Drain, D. Ganguli, T. Henighan, A. Jones, N. Joseph, B. Mann, N. DasSarma, et al. 2021. A general language assistant as a laboratory for alignment. arXiv:2112.00861. Retrieved from https://arxiv.org/abs/2112.00861

[100] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving. 2019. Fine-tuning language models from human preferences. arXiv:1909.08593. Retrieved from https://arxiv.org/abs/1909.08593

[101] S. Kim, S. J. Joo, D. Kim, J. Jang, S. Ye, J. Shin, and M. Seo. 2023. The CoT collection: Improving zero-shot and few-shot learning of language models via chain-of-thought fine-tuning. arXiv:2305.14045. Retrieved from https://arxiv.org/abs/2305.14045

[102] Q. Liu, F. Zhou, Z. Jiang, L. Dou, and M. Lin. 2023. From zero to hero: Examining the power of symbolic tasks in instruction tuning. arXiv:2304.07995. Retrieved from https://arxiv.org/abs/2304.07995

[103] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, and D. Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 35, 24824–24837.

[104] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. arXiv:2203.11171. Retrieved from https://arxiv.org/abs/2203.11171

[105] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. arXiv:2305.10601. Retrieved from https://arxiv.org/abs/2305.10601

[106] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the International Conference on Machine Learning*. PMLR, 2790–2799.

[107] S. McCandlish, J. Kaplan, D. Amodei, and OpenAI Dota Team. 2018. An empirical model of large-batch training. arXiv:1812.06162. Retrieved from https://arxiv.org/abs/1812.06162

[108] OpenAI, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Alemen, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. 2023. GPT-4 technical report. arXiv:2303.08774. Retrieved from https://arxiv.org/abs/2303.08774

[109] A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford, et al. 2024. GPT-4o system card. arXiv:2410.21276. Retrieved from https://arxiv.org/abs/2410.21276

[110] OpenAI O3-Mini System Card. 2025. Retrieved from https://cdn.openai.com/o3-mini-system-card-feb10.pdf

[111] W. Zeng, X. Ren, T. Su, H. Wang, Y. Liao, Z. Wang, X. Jiang, Z. Yang, K. Wang, X. Zhang, et al. 2021. PanGu-α: Large-scale autoregressive pretrained Chinese language models with auto-parallel computation. arXiv:2104.12369. Retrieved from https://arxiv.org/abs/2104.12369

[112] S. Yuan, H. Zhao, Z. Du, M. Ding, X. Liu, Y. Cen, X. Zou, Z. Yang, and J. Tang. 2021. WuDaoCorpora: A super large-scale Chinese corpora for pre-training language models. *AI Open* 2 (2021), 65–68.

[113] Y. Sun, S. Wang, S. Feng, S. Ding, C. Pang, J. Shang, J. Liu, X. Chen, Y. Zhao, Y. Lu, et al. 2021. Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. arXiv:2107.02137. Retrieved from https://arxiv.org/abs/2107.02137

[114] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. arXiv:1901.02860. Retrieved from https://arxiv.org/abs/1901.02860

[115] O. Lieber, O. Sharir, B. Lenz, and Y. Shoham. 2021. *Jurassic-1: Technical Details and Evaluation*. White Paper. AI21 Labs.

[116] Y. Levine, N. Wies, O. Sharir, H. Bata, and A. Shashua. 2020. Limits to depth efficiencies of self-attention. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 33, 22640–22651.

[117] B. Kim, H. Kim, S.-W. Lee, G. Lee, D. Kwak, D. H. Jeon, S. Park, S. Kim, S. Kim, D. Seo, et al. 2021. What changes can large-scale language models bring? Intensive study on HyperCLOVA: Billions-scale Korean generative pretrained transformers. arXiv:2109.04650. Retrieved from https://arxiv.org/abs/2109.04650

[118] S. Wu, X. Zhao, T. Yu, R. Zhang, C. Shen, H. Liu, F. Li, H. Zhu, J. Luo, L. Xu, et al. 2021. Yuan 1.0: Large-scale pre-trained language model in zero-shot and few-shot learning. arXiv:2110.04725. Retrieved from https://arxiv.org/abs/2110.04725
[119] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. arXiv:2112.11446. Retrieved from https://arxiv.org/abs/2112.11446
[120] S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhumoye, G. Zerveas, V. Korthikanti, et al. 2022. Using DeepSpeed and Megatron to train Megatron-Turing NLG 530B, a large-scale generative language model. arXiv:2201.11990. Retrieved from https://arxiv.org/abs/2201.11990
[121] S. Black, S. Biderman, E. Hallahan, Q. Anthony, L. Gao, L. Golding, H. He, C. Leahy, K. McDonell, J. Phang, et al. 2022. Gpt-neox-20b: An open-source autoregressive language model. arXiv:2204.06745. Retrieved from https://arxiv.org/abs/2204.06745
[122] W. Ben and K. Aran. 2021. GPT-J-6B: A 6 billion parameter autoregressive language model.
[123] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, et al. 2017. Mixed precision training. arXiv:1710.03740. Retrieved from https://arxiv.org/abs/1710.03740
[124] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. arXiv:1701.06538. Retrieved from https://arxiv.org/abs/1701.06538
[125] S. Soltan, S. Ananthakrishnan, J. FitzGerald, R. Gupta, W. Hamza, H. Khan, C. Peris, S. Rawls, A. Rosenbaum, A. Rumshisky, et al. 2022. AlexaTM 20B: Few-shot learning using a large-scale multilingual Seq2Seq model. arXiv:2208.01448. Retrieved from https://arxiv.org/abs/2208.01448
[126] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen, et al. 2023. PaLM 2 technical report. arXiv:2305.10403. Retrieved from https://arxiv.org/abs/2305.10403
[127] Y. Tay, J. Wei, H. W. Chung, V. Q. Tran, D. R. So, S. Shakeri, X. Garcia, H. S. Zheng, J. Rao, A. Chowdhery, et al. 2022. Transcending scaling laws with 0.1% extra compute. arXiv:2210.11399. Retrieved from https://arxiv.org/abs/2210.11399
[128] Y. Tay, M. Dehghani, V. Q. Tran, X. Garcia, J. Wei, X. Wang, H. W. Chung, D. Bahri, T. Schuster, S. Zheng, et al. 2022. UL2: Unifying language learning paradigms. In *Proceedings of the 11th International Conference on Learning Representations*.
[129] Z. Du, Y. Qian, X. Liu, M. Ding, J. Qiu, Z. Yang, and J. Tang. 2022. GLM: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, Long Papers*, Vol. 1, 320–335.
[130] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. 2023. LLaMA: Open and efficient foundation language models. arXiv:2302.13971. Retrieved from https://arxiv.org/abs/2302.13971
[131] M. N. Rabe and C. Staats. 2021. Self-attention does not need o(n^2) memory. arXiv:2112.05682. Retrieved from https://arxiv.org/abs/2112.05682
[132] V. A. Korthikanti, J. Casper, S. Lym, L. McAfee, M. Andersch, M. Shoeybi, and B. Catanzaro. 2023. Reducing activation recomputation in large transformer models. *Proceedings of Machine Learning and Systems* 5 (2023), 341–353.
[133] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. 2024. The Llama 3 Herd of models. arXiv:2407.21783. Retrieved from https://arxiv.org/abs/2407.21783
[134] Mistral AI. 2024. Retrieved from https://mistral.ai/news/mixtral-8x22b/
[135] Snowflake-Labs. 2025. Retrieved from https://github.com/Snowflake-Labs/snowflake-arctic
[136] Xai-org. 2025. Retrieved from https://github.com/xai-org/grok-1
[137] Grok-1.5. 2024. Retrieved from https://x.ai/blog/grok-1.5
[138] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, et al. 2023. Gemini: A family of highly capable multimodal models, arXiv:2312.11805. Retrieved from https://arxiv.org/abs/2312.11805
[139] M. Reid, N. Savinov, D. Teplyashin, D. Lepikhin, T. Lillicrap, J.-B. Alayrac, R. Soricut, A. Lazaridou, O. Firat, J. Schrittwieser, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv:2403.05530. Retrieved from https://arxiv.org/abs/2403.05530
[140] B. Adler, N. Agarwal, A. Aithal, D. H. Anh, P. Bhattacharya, A. Brundyn, J. Casper, B. Catanzaro, S. Clay, J. Cohen, et al. 2024. Nemotron-4 340B technical report. arXiv:2406.11704. Retrieved from https://arxiv.org/abs/2406.11704
[141] X. Bi, D. Chen, G. Chen, S. Chen, D. Dai, C. Deng, H. Ding, K. Dong, Q. Du, Z. Fu, et al. 2024. DeepSeek LLM: Scaling open-source language models with longtermism. arXiv:2401.02954. Retrieved from https://arxiv.org/abs/2401.02954
[142] A. DeepSeek-AI, B. Liu, B. Feng, B. Wang, B. Wang, B. Liu, C. Zhao, C. Deng, C. Ruan, D. Dai, et al. 2024. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. arXiv:2405.04434. Retrieved from https://arxiv.org/abs/2405.04434

[143] A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, et al. 2024. DeepSeek-V3 technical report. arXiv:2412.19437. Retrieved from https://arxiv.org/abs/2412.19437

[144] E. Nijkamp, B. Pang, H. Hayashi, L. Tu, H. Wang, Y. Zhou, S. Savarese, C. Xiong. 2022. CodeGen: An open large language model for code with multi-turn program synthesis. arXiv:2203.13474. Retrieved from https://arxiv.org/abs/2203.13474

[145] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. 2021. Evaluating large language models trained on code. arXiv:2107.03374. Retrieved from https://arxiv.org/abs/2107.03374

[146] Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. Dal Lago, et al. 2022. Competition-level code generation with AlphaCode. *Science* 378, 6624 (2022), 1092–1097.

[147] N. Shazeer. 2019. Fast transformer decoding: One write-head is all you need. arXiv:1911.02150. Retrieved from https://arxiv.org/abs/1911.02150

[148] R. Y. Pang and H. He. 2020. Text generation by learning from demonstrations. arXiv:2009.07839. Retrieved from https://arxiv.org/abs/2009.07839

[149] R. Dabre and A. Fujita. 2020. Softmax tempering for training neural machine translation models. arXiv:2009.09372. Retrieved from https://arxiv.org/abs/2009.09372

[150] Y. Wang, W. Wang, S. Joty, and S. C. Hoi. 2021. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. arXiv:2109.00859. Retrieved from https://arxiv.org/abs/2109.00859

[151] R. Li, L. B. Allal, Y. Zi, N. Muennighoff, D. Kocetkov, C. Mou, M. Marone, C. Akiki, J. Li, J. Chim, et al. 2023. StarCoder: May the source be with you! arXiv:2305.06161. Retrieved from https://arxiv.org/abs/2305.06161

[152] R. Taylor, M. Kardas, G. Cucurull, T. Scialom, A. Hartshorn, E. Saravia, A. Poulton, V. Kerkez, and R. Stojnic. 2022. Galactica: A large language model for science. arXiv:2211.09085. Retrieved from https://arxiv.org/abs/2211.09085

[153] FairScale Authors. 2021. FairScale: A General Purpose Modular PyTorch Library for High Performance and Large Scale Training. Retrieved from https://github.com/facebookresearch/fairscale

[154] R. Thoppilan, D. De Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H.-T. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, et al. 2022. LaMDA: Language models for dialog applications. arXiv:2201.08239. Retrieved from https://arxiv.org/abs/2201.08239

[155] S. Wu, O. Irsoy, S. Lu, V. Dabravolski, M. Dredze, S. Gehrmann, P. Kambadur, D. Rosenberg, and G. Mann. 2023. BloombergGPT: A large language model for finance. arXiv:2303.17564. Retrieved from https://arxiv.org/abs/2303.17564

[156] X. Zhang, Q. Yang, and D. Xu. 2023. *XuanYuan 2.0: A large Chinese financial chat model with hundreds of billions parameters.* arXiv:2305.12002. Retrieved from https://arxiv.org/abs/2305.12002

[157] W. Ben. 2021. Mesh-Transformer-JAX: Model-parallel implementation of transformer language model with JAX.

[158] N. Muennighoff, T. Wang, L. Sutawika, A. Roberts, S. Biderman, T. L. Scao, M. S. Bari, S. Shen, Z.-X. Yong, H. Schoelkopf, et al. 2022. Crosslingual generalization through multitask finetuning. arXiv:2211.01786. Retrieved from https://arxiv.org/abs/2211.01786

[159] D. Yin, X. Liu, F. Yin, M. Zhong, H. Bansal, J. Han, and K.-W. Chang. 2023. Dynosaur: A dynamic growth paradigm for instruction-tuning data curation. arXiv:2305.14327. Retrieved from https://arxiv.org/abs/2305.14327

[160] P. Gao, J. Han, R. Zhang, Z. Lin, S. Geng, A. Zhou, W. Zhang, P. Lu, C. He, X. Yue, et al. 2023. LLaMA-Adapter V2: Parameter-efficient visual instruction model. arXiv:2304.15010. Retrieved from https://arxiv.org/abs/2304.15010

[161] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto. 2023. Stanford Alpaca: An instruction-Following LLaMA Model. Retrieved from https://github.com/tatsu-lab/stanford_alpaca

[162] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. Retrieved March 2023 from https://lmsys.org/blog/2023-03-30-vicuna/

[163] B. Peng, C. Li, P. He, M. Galley, and J. Gao. 2023. Instruction tuning with GPT-4. arXiv:2304.03277. Retrieved from https://arxiv.org/abs/2304.03277

[164] T. Liu and B. K. H. Low. 2023. Goat: Fine-tuned llama outperforms GPT-4 on arithmetic tasks. arXiv:2305.14201. Retrieved from https://arxiv.org/abs/2305.14201

[165] H. Wang, C. Liu, N. Xi, Z. Qiang, S. Zhao, B. Qin, and T. Liu. 2023. HuaTuo: Tuning LLaMA model with Chinese medical knowledge. arXiv:2304.06975. Retrieved from https://arxiv.org/abs/2304.06975

[166] C. Xu, Q. Sun, K. Zheng, X. Geng, P. Zhao, J. Feng, C. Tao, and D. Jiang. 2023. WizardLM: Empowering large language models to follow complex instructions. arXiv:2304.12244. Retrieved from https://arxiv.org/abs/2304.12244

[167] Z. Luo, C. Xu, P. Zhao, Q. Sun, X. Geng, W. Hu, C. Tao, J. Ma, Q. Lin, and D. Jiang. 2023. WizardCoder: Empowering code large language models with Evol-Instruct. arXiv:2306.08568. Retrieved from https://arxiv.org/abs/2306.08568

[168] J. Menick, M. Trebacz, V. Mikulik, J. Aslanides, F. Song, M. Chadwick, M. Glaese, S. Young, L. Campbell-Gillingham, G. Irving, et al. 2022. Teaching language models to support answers with verified quotes. arXiv:2203.11147. Retrieved from https://arxiv.org/abs/2203.11147

[169] R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, et al. 2021. WebGPT: Browser-assisted question-answering with human feedback. arXiv:2112.09332. Retrieved from https://arxiv.org/abs/2112.09332

[170] A. Glaese, N. McAleese, M. Trębacz, J. Aslanides, V. Firoiu, T. Ewalds, M. Rauh, L. Weidinger, M. Chadwick, P. Thacker, et al. 2022. Improving alignment of dialogue agents via targeted human judgements. arXiv:2209.14375. Retrieved from https://arxiv.org/abs/2209.14375

[171] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. arXiv:2305.18290. Retrieved from https://arxiv.org/abs/2305.18290

[172] H. Dong, W. Xiong, D. Goyal, R. Pan, S. Diao, J. Zhang, K. Shum, and T. Zhang. 2023. RAFT: Reward ranked finetuning for generative foundation model alignment. arXiv:2304.06767. Retrieved from https://arxiv.org/abs/2304.06767

[173] Z. Yuan, H. Yuan, C. Tan, W. Wang, S. Huang, and F. Huang. 2023. RRHF: Rank responses to align language models with human feedback without tears. arXiv:2304.05302. Retrieved from https://arxiv.org/abs/2304.05302

[174] F. Song, B. Yu, M. Li, H. Yu, F. Huang, Y. Li, and H. Wang. 2023. Preference ranking optimization for human alignment. arXiv:2306.17492. Retrieved from https://arxiv.org/abs/2306.17492

[175] H. Liu, C. Sferrazza, and P. Abbeel. 2023. Languages are rewards: Hindsight finetuning using human feedback. arXiv:2302.02676. Retrieved from https://arxiv.org/abs/2302.02676

[176] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, et al. 2022. Constitutional AI: Harmlessness from AI feedback. arXiv:2212.08073. Retrieved from https://arxiv.org/abs/2212.08073

[177] Y. Dubois, X. Li, R. Taori, T. Zhang, I. Gulrajani, J. Ba, C. Guestrin, P. Liang, and T. B. Hashimoto. 2023. AlpacaFarm: A simulation framework for methods that learn from human feedback. arXiv:2305.14387. Retrieved from https://arxiv.org/abs/2305.14387

[178] C. Si, Z. Gan, Z. Yang, S. Wang, J. Wang, J. Boyd-Graber, and L. Wang. 2022. Prompting GPT-3 to be reliable. arXiv:2210.09150. Retrieved from https://arxiv.org/abs/2210.09150

[179] D. Ganguli, A. Askell, N. Schiefer, T. Liao, K. Lukošiūtė, A. Chen, A. Goldie, A. Mirhoseini, C. Olsson, D. Hernandez, et al. 2023. The capacity for moral self-correction in large language models. arXiv:2302.07459. Retrieved from https://arxiv.org/abs/2302.07459

[180] A. Wei, N. Haghtalab, and J. Steinhardt. 2023. Jailbroken: How does LLM safety training fail? arXiv:2307.02483. Retrieved from https://arxiv.org/abs/2307.02483

[181] D. Ganguli, L. Lovitt, J. Kernion, A. Askell, Y. Bai, S. Kadavath, B. Mann, E. Perez, N. Schiefer, K. Ndousse, et al. 2022. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. arXiv:2209.07858. Retrieved from https://arxiv.org/abs/2209.07858

[182] S. Casper, J. Lin, J. Kwon, G. Culp, and D. Hadfield-Menell. 2023. Explore, establish, exploit: Red teaming language models from scratch. arXiv:2306.09442. Retrieved from https://arxiv.org/abs/2306.09442

[183] E. Perez, S. Huang, F. Song, T. Cai, R. Ring, J. Aslanides, A. Glaese, N. McAleese, and G. Irving. 2022. Red teaming language models with language models. arXiv:2202.03286. Retrieved from https://arxiv.org/abs/2202.03286

[184] T. Scialom, T. Chakrabarty, and S. Muresan. 2022. Fine-tuned language models are continual learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 6107–6122.

[185] Z. Shi and A. Lipani. 2023. Don't stop pretraining? Make prompt-based fine-tuning powerful learner. arXiv:2305.01711. Retrieved from https://arxiv.org/abs/2305.01711

[186] H. Gupta, S. A. Sawant, S. Mishra, M. Nakamura, A. Mitra, S. Mashetty, and C. Baral. 2023. Instruction tuned models are quick learners. arXiv:2306.05539. Retrieved from https://arxiv.org/abs/2306.05539

[187] H. Chen, Y. Zhang, Q. Zhang, H. Yang, X. Hu, X. Ma, Y. Yanggong, and J. Zhao. 2023. Maybe only 0.5% data is needed: A preliminary exploration of low training data instruction tuning. arXiv:2305.09246. Retrieved from https://arxiv.org/abs/2305.09246

[188] C. Zhou, P. Liu, P. Xu, S. Iyer, J. Sun, Y. Mao, X. Ma, A. Efrat, P. Yu, L. Yu, et al. 2023. LIMA: Less is more for alignment. arXiv:2305.11206. Retrieved from https://arxiv.org/abs/2305.11206

[189] C. Han, Q. Wang, W. Xiong, Y. Chen, H. Ji, and S. Wang. 2023. LM-infinite: Simple on-the-fly length generalization for large language models. arXiv:2308.16137. Retrieved from https://arxiv.org/abs/2308.16137

[190] J. Ainslie, T. Lei, M. de Jong, S. Ontañón, S. Brahma, Y. Zemlyanskiy, D. Uthus, M. Guo, J. Lee-Thorp, Y. Tay, et al. 2023. CoLT5: Faster long-range transformers with conditional computation. arXiv:2303.09752. Retrieved from https://arxiv.org/abs/2303.09752

[191] J. Ding, S. Ma, L. Dong, X. Zhang, S. Huang, W. Wang, and F. Wei. 2023. LongNet: Scaling transformers to 1,000,000,000 tokens. arXiv:2307.02486. Retrieved from https://arxiv.org/abs/2307.02486

[192] Y. Chen, S. Qian, H. Tang, X. Lai, Z. Liu, S. Han, and J. Jia. 2023. LongLoRA: Efficient fine-tuning of long-context large language models. arXiv:2309.12307. Retrieved from https://arxiv.org/abs/2309.12307

[193] N. Ratner, Y. Levine, Y. Belinkov, O. Ram, I. Magar, O. Abend, E. Karpas, A. Shashua, K. Leyton-Brown, and Y. Shoham. 2023. Parallel context windows for large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 6383–6402.

[194] W. Wang, L. Dong, H. Cheng, X. Liu, X. Yan, J. Gao, and F. Wei. 2023. Augmenting language models with long-term memory. arXiv:2306.07174. Retrieved from https://arxiv.org/abs/2306.07174
[195] X. Xu, Z. Gou, W. Wu, Z.-Y. Niu, H. Wu, H. Wang, and S. Wang. 2022. Long time no see! Open-domain conversation with long-term persona memory. arXiv:2203.05797. Retrieved from https://arxiv.org/abs/2203.05797
[196] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. Van Den Driessche, J.-B. Lespiau, B. Damoc, A. Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *Proceedings of the International Conference on Machine Learning*. PMLR, 2206–2240.
[197] W. Zhong, L. Guo, Q. Gao, and Y. Wang. 2023. MemoryBank: Enhancing large language models with long-term memory. arXiv:2305.10250. Retrieved from https://arxiv.org/abs/2305.10250
[198] N. Shinn, F. Cassano, B. Labash, A. Gopinath, K. Narasimhan, and S. Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. arXiv:2303.11366. Retrieved from https://arxiv.org/abs/2303.11366
[199] C. Hu, J. Fu, C. Du, S. Luo, J. Zhao, and H. Zhao. 2023. ChatDB: Augmenting LLMs with databases as their symbolic memory. arXiv:2306.03901. Retrieved from https://arxiv.org/abs/2306.03901
[200] Z. Jiang, F. F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig. 2023. Active retrieval augmented generation. arXiv:2305.06983. Retrieved from https://arxiv.org/abs/2305.06983
[201] O. Ram, Y. Levine, I. Dalmedigos, D. Muhlgay, A. Shashua, K. Leyton-Brown, and Y. Shoham. 2023. In-context retrieval-augmented language models. arXiv:2302.00083. Retrieved from https://arxiv.org/abs/2302.00083
[202] X. Li and X. Qiu. 2023. MoT: Pre-thinking and recalling enable ChatGPT to self-improve with memory-of-thoughts. arXiv:2305.05181. Retrieved from https://arxiv.org/abs/2305.05181
[203] D. Schuurmans. 2023. Memory augmented large language models are computationally universal. arXiv:2301.04589. Retrieved from https://arxiv.org/abs/2301.04589
[204] A. Modarressi, A. Imani, M. Fayyaz, and H. Schütze. 2023. RET-LLM: Towards a general read-write memory for large language models. arXiv:2305.14322. Retrieved from https://arxiv.org/abs/2305.14322
[205] S. Robertson and H. Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.
[206] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, and D. Zhou. 2022. Rationale-augmented ensembles in language models. arXiv:2207.00747. Retrieved from https://arxiv.org/abs/2207.00747
[207] F. Zhang, B. Chen, Y. Zhang, J. Liu, D. Zan, Y. Mao, J.-G. Lou, and W. Chen. 2023. RepoCoder: Repository-level code completion through iterative retrieval and generation. arXiv:2303.12570. Retrieved from https://arxiv.org/abs/2303.12570
[208] B. Wang, W. Ping, P. Xu, L. McAfee, Z. Liu, M. Shoeybi, Y. Dong, O. Kuchaiev, B. Li, C. Xiao, et al. 2023. Shall we pretrain autoregressive language models with retrieval? A comprehensive study. arXiv:2304.06762. Retrieved from https://arxiv.org/abs/2304.06762
[209] L. Wang, N. Yang, and F. Wei. 2023. Learning to retrieve in-context examples for large language models. arXiv:2307.07164. Retrieved from https://arxiv.org/abs/2307.07164
[210] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, and W. Chen. 2021. What makes good in-context examples for GPT-3? arXiv:2101.06804. Retrieved from https://arxiv.org/abs/2101.06804
[211] O. Rubin, J. Herzig, and J. Berant. 2021. Learning to retrieve prompts for in-context learning. arXiv:2112.08633. Retrieved from https://arxiv.org/abs/2112.08633
[212] W. Shi, S. Min, M. Yasunaga, M. Seo, R. James, M. Lewis, L. Zettlemoyer, and W. Yih. 2023. REPLUG: Retrieval-augmented black-box language models. arXiv:2301.12652. Retrieved from https://arxiv.org/abs/2301.12652
[213] O. Rubin and J. Berant. 2023. Long-range language modeling with self-retrieval. arXiv:2306.13421. Retrieved from https://arxiv.org/abs/2306.13421
[214] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the International Conference on Machine Learning*. PMLR, 3929–3938.
[215] S. Hofstätter, J. Chen, K. Raman, and H. Zamani. 2023. FiD-Light: Efficient and effective retrieval-augmented text generation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1437–1447.
[216] M. Komeili, K. Shuster, and J. Weston. 2021. Internet-augmented dialogue generation. arXiv:2107.07566. Retrieved from https://arxiv.org/abs/2107.07566
[217] A. Lazaridou, E. Gribovskaya, W. Stokowiec, and N. Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering. arXiv:2203.05115. Retrieved from https://arxiv.org/abs/2203.05115
[218] D. Gao, L. Ji, L. Zhou, K. Q. Lin, J. Chen, Z. Fan, and M. Z. Shou. 2023. AssistGPT: A general multi-modal assistant that can plan, execute, inspect, and learn. arXiv:2306.08640. Retrieved from https://arxiv.org/abs/2306.08640
[219] P. Lu, B. Peng, H. Cheng, M. Galley, K.-W. Chang, Y. N. Wu, S.-C. Zhu, and J. Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models. arXiv:2304.09842. Retrieved from https://arxiv.org/abs/2304.09842

[220] B. Paranjape, S. Lundberg, S. Singh, H. Hajishirzi, L. Zettlemoyer, and M. T. Ribeiro. 2023. ART: Automatic multi-step reasoning and tool-use for large language models. arXiv:2303.09014. Retrieved from https://arxiv.org/abs/2303.09014
[221] C.-Y. Hsieh, S.-A. Chen, C.-L. Li, Y. Fujii, A. Ratner, C.-Y. Lee, R. Krishna, and T. Pfister. 2023. Tool documentation enables zero-shot tool-usage with large language models. arXiv:2308.00675. Retrieved from https://arxiv.org/abs/2308.00675
[222] Y. Song, W. Xiong, D. Zhu, C. Li, K. Wang, Y. Tian, and S. Li. 2023. RestGPT: Connecting large language models with real-world applications via RESTful APIs. arXiv:2306.06624. Retrieved from https://arxiv.org/abs/2306.06624
[223] S. Hao, T. Liu, Z. Wang, and Z. Hu. 2023. ToolkenGPT: Augmenting frozen language models with massive tools via tool embeddings. arXiv:2305.11554. Retrieved from https://arxiv.org/abs/2305.11554
[224] S. G. Patil, T. Zhang, X. Wang, and J. E. Gonzalez. 2023. Gorilla: Large language model connected with massive APIs. arXiv:2305.15334. Retrieved from https://arxiv.org/abs/2305.15334
[225] Q. Xu, F. Hong, B. Li, C. Hu, Z. Chen, and J. Zhang. 2023. On the tool manipulation capability of open-source large language models. arXiv:2305.16504. Retrieved from https://arxiv.org/abs/2305.16504
[226] Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian, et al. 2023. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. arXiv:2307.16789. Retrieved from https://arxiv.org/abs/2307.16789
[227] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang. 2023. HuggingGPT: Solving AI tasks with ChatGPT and its friends in Hugging Face. arXiv:2303.17580. Retrieved from https://arxiv.org/abs/2303.17580
[228] Y. Liang, C. Wu, T. Song, W. Wu, Y. Xia, Y. Liu, Y. Ou, S. Lu, L. Ji, S. Mao, et al. 2023. TaskMatrix.AI: Completing tasks by connecting foundation models with millions of APIs. arXiv:2303.16434. Retrieved from https://arxiv.org/abs/2303.16434
[229] D. Surís, S. Menon, and C. Vondrick. 2023. ViperGPT: Visual inference via Python execution for reasoning. arXiv:2303.08128. Retrieved from https://arxiv.org/abs/2303.08128
[230] A. Maedche, S. Morana, S. Schacht, D. Werth, and J. Krumeich. 2016. Advanced user assistance systems. *Business & Information Systems Engineering* 58, 5 (2016), 367–370.
[231] M. Campbell, A. J. Hoane Jr, and F. Hsu. 2002. Deep blue. *Artificial Intelligence* 134, 1–2 (2002), 57–83.
[232] S. Hong, X. Zheng, J. Chen, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, et al. 2023. MetaGPT: Meta programming for multi-agent collaborative framework. arXiv:2308.00352. Retrieved from https://arxiv.org/abs/2308.00352
[233] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, et al. 2023. The rise and potential of large language model based agents: A survey. arXiv:2309.07864. Retrieved from https://arxiv.org/abs/2309.07864
[234] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, et al. 2023. A survey on large language model based autonomous agents. arXiv:2308.11432. Retrieved from https://arxiv.org/abs/2308.11432
[235] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *Proceedings of the International Conference on Machine Learning*. PMLR, 9118–9147.
[236] S. Hao, Y. Gu, H. Ma, J. J. Hong, Z. Wang, D. Z. Wang, and Z. Hu. 2023. Reasoning with language model is planning with world model. arXiv:2305.14992. Retrieved from https://arxiv.org/abs/2305.14992
[237] W. Yao, S. Heinecke, J. C. Niebles, Z. Liu, Y. Feng, L. Xue, R. Murthy, Z. Chen, J. Zhang, D. Arpit, et al. 2023. Retroformer: Retrospective large language agents with policy gradient optimization. arXiv:2308.02151. Retrieved from https://arxiv.org/abs/2308.02151
[238] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, T. Jackson, N. Brown, L. Luu, S. Levine, K. Hausman, and B. Ichter. 2022. Inner monologue: Embodied reasoning through planning with language models. In *Proceedings of the 6th Annual Conference on Robot Learning*. Retrieved from https://openreview.net/forum?id=3R3Pz5i0tye.
[239] C. Jin, W. Tan, J. Yang, B. Liu, R. Song, L. Wang, and J. Fu. 2023. AlphaBlock: Embodied finetuning for vision-language reasoning in robot manipulation. arXiv:2305.18898. Retrieved from https://arxiv.org/abs/2305.18898
[240] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. 2023. ProgPrompt: Generating situated robot task plans using large language models. In *Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 11523–11530.
[241] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik, et al. 2023. Language to rewards for robotic skill synthesis. arXiv:2306.08647. Retrieved from https://arxiv.org/abs/2306.08647
[242] X. Tang, A. Zou, Z. Zhang, Y. Zhao, X. Zhang, A. Cohan, and M. Gerstein. 2023. MedAgents: Large language models as collaborators for zero-shot medical reasoning. arXiv:2311.10537. Retrieved from https://arxiv.org/abs/2311.10537
[243] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, et al. 2023. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*. PMLR, 287–318.
[244] H. Ha, P. Florence, and S. Song. 2023. Scaling up and distilling down: Language-guided robot skill acquisition. arXiv:2307.14535. Retrieved from https://arxiv.org/abs/2307.14535

[245] A. Rajvanshi, K. Sikka, X. Lin, B. Lee, H.-P. Chiu, and A. Velasquez. 2023. SayNav: Grounding large language models for dynamic planning to navigation in new environments. arXiv:2309.04077. Retrieved from https://arxiv.org/abs/2309.04077

[246] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su. 2022. LLM-Planner: Few-shot grounded planning for embodied agents with large language models. arXiv:2212.04088. Retrieved from https://arxiv.org/abs/2212.04088

[247] V. S. Dorbala, J. F. Mullen, Jr, and D. Manocha. 2023. Can an embodied agent find your "cat-shaped mug"? LLM-based zero-shot object navigation. arXiv:2303.03480. Retrieved from https://arxiv.org/abs/2303.03480

[248] C. Huang, O. Mees, A. Zeng, and W. Burgard. 2023. Visual language maps for robot navigation. In *Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 10608–10615.

[249] Y. Ding, X. Zhang, C. Paxton, and S. Zhang. 2023. Task and motion planning with large language models for object rearrangement. arXiv:2303.06247. Retrieved from https://arxiv.org/abs/2303.06247

[250] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang. 2021. GPT understands, too. arXiv:2103.10385. Retrieved from https://arxiv.org/abs/2103.10385

[251] G. Chen, F. Liu, Z. Meng, and S. Liang. 2022. Revisiting parameter-efficient tuning: Are we really there yet? arXiv:2202.07962. Retrieved from https://arxiv.org/abs/2202.07962

[252] Y. Wang, S. Mukherjee, X. Liu, J. Gao, A. H. Awadallah, and J. Gao. 2022. AdaMix: Mixture-of-adapter for parameter-efficient tuning of large language models. arXiv:2205.12410. Retrieved from https://arxiv.org/abs/2205.12410

[253] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. 2021. LoRA: Low-rank adaptation of large language models. arXiv:2106.09685. Retrieved from https://arxiv.org/abs/2106.09685

[254] X. Liu, K. Ji, Y. Fu, W. Tam, Z. Du, Z. Yang, and J. Tang. 2022. P-Tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 61–68.

[255] A. Razdaibiedina, Y. Mao, R. Hou, M. Khabsa, M. Lewis, and A. Almahairi. 2023. Progressive prompts: Continual learning for language models. arXiv:2301.12314. Retrieved from https://arxiv.org/abs/2301.12314

[256] Z.-R. Zhang, C. Tan, H. Xu, C. Wang, J. Huang, and S. Huang. 2023. Towards adaptive prefix tuning for parameter-efficient language model fine-tuning. arXiv:2305.15212. Retrieved from https://arxiv.org/abs/2305.15212

[257] E. B. Zaken, S. Ravfogel, and Y. Goldberg. 2021. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. arXiv:2106.10199. Retrieved from https://arxiv.org/abs/2106.10199

[258] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer. 2022. LLM.int8(): 8-bit matrix multiplication for transformers at scale. arXiv:2208.07339. Retrieved from https://arxiv.org/abs/2208.07339

[259] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh. 2022. GPTQ: Accurate post-training quantization for generative pre-trained transformers. arXiv:2210.17323. Retrieved from https://arxiv.org/abs/2210.17323

[260] X. Wei, Y. Zhang, Y. Li, X. Zhang, R. Gong, J. Guo, and X. Liu. 2023. Outlier Suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling. arXiv:2304.09145. Retrieved from https://arxiv.org/abs/2304.09145

[261] E. Frantar and D. Alistarh. 2022. Optimal brain compression: A framework for accurate post-training quantization and pruning. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 35, 4475–4488.

[262] C. Lee, J. Jin, T. Kim, H. Kim, and E. Park. 2023. OWQ: Lessons learned from activation outliers for weight quantization in large language models. arXiv:2306.02272. Retrieved from https://arxiv.org/abs/2306.02272

[263] S. J. Kwon, J. Kim, J. Bae, K. M. Yoo, J.-H. Kim, B. Park, B. Kim, J.-W. Ha, N. Sung, and D. Lee. 2022. AlphaTuning: Quantization-aware parameter-efficient adaptation of large-scale pre-trained language models. arXiv:2210.03858. Retrieved from https://arxiv.org/abs/2210.03858

[264] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. arXiv:2305.14314. Retrieved from https://arxiv.org/abs/2305.14314

[265] Z. Liu, B. Oguz, C. Zhao, E. Chang, P. Stock, Y. Mehdad, Y. Shi, R. Krishnamoorthi, and V. Chandra. 2023. LLM-QAT: Data-free quantization aware training for large language models. arXiv:2305.17888. Retrieved from https://arxiv.org/abs/2305.17888

[266] Y. Guo, A. Yao, H. Zhao, and Y. Chen. 2017. Network sketching: Exploiting binary structure in deep CNNs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5955–5963.

[267] J. Kim, J. H. Lee, S. Kim, J. Park, K. M. Yoo, S. J. Kwon, and D. Lee. 2023. Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. arXiv:2305.14152. Retrieved from https://arxiv.org/abs/2305.14152

[268] M. Sun, Z. Liu, A. Bair, and J. Z. Kolter. 2023. A simple and effective pruning approach for large language models. arXiv:2306.11695. Retrieved from https://arxiv.org/abs/2306.11695

[269] Z. Wang, J. Wohlwend, and T. Lei. 2019. Structured pruning of large language models. arXiv:1910.04732. Retrieved from https://arxiv.org/abs/1910.04732

[270] L. Yin, Y. Wu, Z. Zhang, C.-Y. Hsieh, Y. Wang, Y. Jia, M. Pechenizkiy, Y. Liang, Z. Wang, and S. Liu. 2023. Outlier weighed layerwise sparsity (OWL): A missing secret sauce for pruning LLMs to high sparsity. arXiv:2310.05175. Retrieved from https://arxiv.org/abs/2310.05175

[271] C. Tao, L. Hou, H. Bai, J. Wei, X. Jiang, Q. Liu, P. Luo, and N. Wong. 2023. Structured pruning for efficient generative pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, 10880–10895.

[272] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, et al. 2022. Flamingo: A visual language model for few-shot learning. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 35, 23716–23736.

[273] J. Li, D. Li, S. Savarese, and S. Hoi. 2023. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. arXiv:2301.12597. Retrieved from https://arxiv.org/abs/2301.12597

[274] H. Liu, C. Li, Q. Wu, and Y. J. Lee. 2023. Visual instruction tuning. arXiv:2304.08485. Retrieved from https://arxiv.org/abs/2304.08485

[275] K. Li, Y. He, Y. Wang, Y. Li, W. Wang, P. Luo, Y. Wang, L. Wang, and Y. Qiao. 2023. VideoChat: Chat-centric video understanding. arXiv:2305.06355. Retrieved from https://arxiv.org/abs/2305.06355

[276] M. Maaz, H. Rasheed, S. Khan, and F. S. Khan. 2023. Video-ChatGPT: Towards detailed video understanding via large vision and language models. arXiv:2306.05424. Retrieved from https://arxiv.org/abs/2306.05424

[277] H. Zhang, X. Li, and L. Bing. 2023. Video-LLaMA: An instruction-tuned audio-visual language model for video understanding. arXiv:2306.02858. Retrieved from https://arxiv.org/abs/2306.02858

[278] X. Mei, C. Meng, H. Liu, Q. Kong, T. Ko, C. Zhao, M. D. Plumbley, Y. Zou, and W. Wang. 2023. WavCaps: A ChatGPT-assisted weakly-labelled audio captioning dataset for audio-language multimodal research. arXiv:2303.17395. Retrieved from https://arxiv.org/abs/2303.17395

[279] C. Lyu, M. Wu, L. Wang, X. Huang, B. Liu, Z. Du, S. Shi, and Z. Tu. 2023. Macaw-LLM: Multi-modal language modeling with image, audio, video, and text integration. arXiv:2306.09093. Retrieved from https://arxiv.org/abs/2306.09093

[280] D. Zhu, J. Chen, X. Shen, X. Li, and M. Elhoseiny. 2023. MiniGPT-4: Enhancing vision-language understanding with advanced large language models. arXiv:2304.10592. Retrieved from https://arxiv.org/abs/2304.10592

[281] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. 2020. An image is worth 16×16 words: Transformers for image recognition at scale. arXiv:2010.11929. Retrieved from https://arxiv.org/abs/2010.11929

[282] W. Dai, J. Li, D. Li, A. M. H. Tiong, J. Zhao, W. Wang, B. Li, P. Fung, and S. Hoi. 2023. InstructBLIP: Towards general-purpose vision-language models with instruction tuning. arXiv:2305.06500. Retrieved from https://arxiv.org/abs/2305.06500

[283] Z. Xu, Y. Shen, and L. Huang. 2022. MultiInstruct: Improving multi-modal zero-shot learning via instruction tuning. arXiv:2212.10773. Retrieved from https://arxiv.org/abs/2212.10773

[284] Z. Zhao, L. Guo, T. Yue, S. Chen, S. Shao, X. Zhu, Z. Yuan, and J. Liu. 2023. ChatBridge: Bridging modalities with large language model as a language catalyst. arXiv:2305.16103. Retrieved from https://arxiv.org/abs/2305.16103

[285] L. Li, Y. Yin, S. Li, L. Chen, P. Wang, S. Ren, M. Li, Y. Yang, J. Xu, X. Sun, et al. 2023. M^3IT: A large-scale dataset towards multi-modal multilingual instruction tuning. arXiv:2306.04387. Retrieved from https://arxiv.org/abs/2306.04387

[286] R. Pi, J. Gao, S. Diao, R. Pan, H. Dong, J. Zhang, L. Yao, J. Han, H. Xu, and L. K. T. Zhang. 2023. DetGPT: Detect what you need via reasoning. arXiv:2305.14167. Retrieved from https://arxiv.org/abs/2305.14167

[287] G. Luo, Y. Zhou, T. Ren, S. Chen, X. Sun, and R. Ji. 2023. Cheap and quick: Efficient vision-language instruction tuning for large language models. arXiv:2305.15023. Retrieved from https://arxiv.org/abs/2305.15023

[288] R. Zhang, J. Han, A. Zhou, X. Hu, S. Yan, P. Lu, H. Li, P. Gao, and Y. Qiao. 2023. LLaMA-Adapter: Efficient fine-tuning of language models with zero-init attention. arXiv:2303.16199. Retrieved from https://arxiv.org/abs/2303.16199

[289] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *Proceedings of the International Conference on Machine Learning*. PMLR, 28492–28518.

[290] Z. Zhang, A. Zhang, M. Li, H. Zhao, G. Karypis, and A. Smola. 2023. Multimodal chain-of-thought reasoning in language models. arXiv:2302.00923. Retrieved from https://arxiv.org/abs/2302.00923

[291] J. Ge, H. Luo, S. Qian, Y. Gan, J. Fu, and S. Zhan. 2023. Chain of thought prompt tuning in vision language models. arXiv:2304.07919. Retrieved from https://arxiv.org/abs/2304.07919

[292] C. Wu, S. Yin, W. Qi, X. Wang, Z. Tang, and N. Duan. 2023. Visual ChatGPT: Talking, drawing and editing with visual foundation models. arXiv:2303.04671. Retrieved from https://arxiv.org/abs/2303.04671

[293] Z. Yang, L. Li, J. Wang, K. Lin, E. Azarnasab, F. Ahmed, Z. Liu, C. Liu, M. Zeng, and L. Wang. 2023. MM-REACT: Prompting ChatGPT for multimodal reasoning and action. arXiv:2303.11381. Retrieved from https://arxiv.org/abs/2303.11381

[294] T. Wang, J. Zhang, J. Fei, Y. Ge, H. Zheng, Y. Tang, Z. Li, M. Gao, S. Zhao, Y. Shan, et al. 2023. Caption anything: Interactive image description with diverse multimodal controls. arXiv:2305.02677. Retrieved from https://arxiv.org/abs/2305.02677

[295] X. Zhu, R. Zhang, B. He, Z. Zeng, S. Zhang, and P. Gao. 2022. PointCLIP V2: Adapting CLIP for powerful 3D open-world learning. arXiv:2211.11682. Retrieved from https://arxiv.org/abs/2211.11682
[296] T. Gupta and A. Kembhavi. 2023. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14953–14962.
[297] P. Gao, Z. Jiang, H. You, P. Lu, S. C. Hoi, X. Wang, and H. Li. 2019. Dynamic fusion with intra-and inter-modality attention flow for visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6639–6648.
[298] Z. Yu, J. Yu, Y. Cui, D. Tao, and Q. Tian. 2019. Deep modular co-attention networks for visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6281–6290.
[299] H. You, R. Sun, Z. Wang, L. Chen, G. Wang, H. A. Ayyubi, K.-W. Chang, and S.-F. Chang. 2023. IdealGPT: Iteratively decomposing vision and language reasoning via large language models. arXiv:2305.14985. Retrieved from https://arxiv.org/abs/2305.14985
[300] R. Zhang, X. Hu, B. Li, S. Huang, H. Deng, Y. Qiao, P. Gao, and H. Li. 2023. Prompt, generate, then cache: Cascade of foundation models makes strong few-shot learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15211–15222.
[301] T. Q. Nguyen and J. Salazar. 2019. Transformers without tears: Improving the normalization of self-attention. arXiv:1910.05895. Retrieved from https://arxiv.org/abs/1910.05895
[302] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. 2019. RoBERTa: A robustly optimized bert pretraining approach. arXiv:1907.11692. Retrieved from https://arxiv.org/abs/1907.11692
[303] X. Geng, A. Gudibande, H. Liu, E. Wallace, P. Abbeel, S. Levine, and D. Song. 2023. Koala: A Dialogue Model for Academic Research, Blog Post. Retrieved April 2023 from https://bair.berkeley.edu/blog/2023/04/03/koala/
[304] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, et al. 2020. The pile: An 800GB dataset of diverse text for language modeling. arXiv:2101.00027. Retrieved from https://arxiv.org/abs/2101.00027
[305] H. Laurençon, L. Saulnier, T. Wang, C. Akiki, A. Villanova del Moral, T. Le Scao, L. Von Werra, C. Mou, E. González Ponferrada, H. Nguyen, et al. 2022. The BigScience ROOTS corpus: A 1.6TB composite multilingual dataset. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 35, 31809–31826.
[306] Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Main_Page
[307] Together Computer. 2023. RedPajama: An Open Source Recipe to Reproduce LLaMA Training Dataset. 2023. Retrieved April 2023 from https://github.com/togethercomputer/RedPajama-Data
[308] O. Honovich, T. Scialom, O. Levy, and T. Schick. 2022. Unnatural instructions: Tuning language models with (almost) no human labor. arXiv:2212.09689. Retrieved from https://arxiv.org/abs/2212.09689
[309] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. arXiv:2204.05862. Retrieved from https://arxiv.org/abs/2204.05862
[310] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. 2020. Measuring massive multitask language understanding. arXiv:2009.03300. Retrieved from https://arxiv.org/abs/2009.03300
[311] A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. arXiv:2206.04615. Retrieved from https://arxiv.org/abs/2206.04615
[312] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. arXiv:1804.07461. Retrieved from https://arxiv.org/abs/1804.07461
[313] Y. Yao, Q. Dong, J. Guan, B. Cao, Z. Zhang, C. Xiao, X. Wang, F. Qi, J. Bao, J. Nie, et al. 2021. CUGE: A Chinese language understanding and generation evaluation benchmark. arXiv:2112.13610. Retrieved from https://arxiv.org/abs/2112.13610
[314] L. Xu, H. Hu, X. Zhang, L. Li, C. Cao, Y. Li, Y. Xu, K. Sun, D. Yu, C. Yu, et al. 2020. CLUE: A Chinese language understanding evaluation benchmark. arXiv:2004.05986. Retrieved from https://arxiv.org/abs/2004.05986
[315] L. Xu, X. Lu, C. Yuan, X. Zhang, H. Xu, H. Yuan, G. Wei, X. Pan, X. Tian, L. Qin, et al. 2021. FewCLUE: A Chinese few-shot learning evaluation benchmark. arXiv:2107.07498. Retrieved from https://arxiv.org/abs/2107.07498
[316] E. M. Smith, M. Williamson, K. Shuster, J. Weston, and Y.-L. Boureau. 2020. Can you put it all together: Evaluating conversational agents' ability to blend skills. arXiv:2004.08449. Retrieved from https://arxiv.org/abs/2004.08449
[317] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, et al. 2022. Holistic evaluation of language models. arXiv:2211.09110. Retrieved from https://arxiv.org/abs/2211.09110
[318] S. Park, J. Moon, S. Kim, W. I. Cho, J. Han, J. Park, C. Song, J. Kim, Y. Song, T. Oh, et al. 2021. KLUE: Korean language understanding evaluation. arXiv:2105.09680. Retrieved from https://arxiv.org/abs/2105.09680
[319] S. Reddy, D. Chen, and C. D. Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics* 7 (2019), 249–266.

[320] M. T. Pilehvar and J. Camacho-Collados. 2018. WiC: 10,000 example pairs for evaluating context-sensitive representations. arXiv:1808.09121. Retrieved from https://arxiv.org/abs/1808.09121
[321] S. Merity, C. Xiong, J. Bradbury, and R. Socher. 2016. Pointer sentinel mixture models. arXiv:1609.07843. Retrieved from https://arxiv.org/abs/1609.07843
[322] J. W. Rae, A. Potapenko, S. M. Jayakumar, and T. P. Lillicrap. 2019. Compressive transformers for long-range sequence modelling. arXiv:1911.05507. Retrieved from https://arxiv.org/abs/1911.05507
[323] X. Liu, Q. Chen, C. Deng, H. Zeng, J. Chen, D. Li, and B. Tang. 2018. LCQMC: A large-scale Chinese question matching corpus. In *Proceedings of the 27th International Conference on Computational Linguistics*, 1952–1962.
[324] S. Iyer, N. Dandekar, and K. Csernai. 2025. First Quora Dataset Release: Question Pairs. Retrieved from https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs
[325] R. Rudinger, J. Naradowsky, B. Leonard, and B. Van Durme. 2018. Gender bias in coreference resolution. arXiv:1804.09301. Retrieved from https://arxiv.org/abs/1804.09301
[326] M.-C. De Marneffe, M. Simons, and J. Tonhauser. 2019. The CommitmentBank: Investigating projection in naturally occurring discourse. In *Proceedings of Sinn und Bedeutung*, Vol. 23, 107–124.
[327] Z. Li, N. Ding, Z. Liu, H. Zheng, and Y. Shen. 2019. Chinese relation extraction with multi-grained information and external linguistic knowledge. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4377–4386.
[328] J. Xu, J. Wen, X. Sun, and Q. Su. 2017. A discourse-level named entity recognition and relation extraction dataset for Chinese literature text. arXiv:1711.07010. Retrieved from https://arxiv.org/abs/1711.07010
[329] J. Chen, Q. Chen, X. Liu, H. Yang, D. Lu, and B. Tang. 2018. The BQ corpus: A large-scale domain-specific Chinese corpus for sentence semantic equivalence identification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 4946–4951.
[330] B. Liu, D. Niu, H. Wei, J. Lin, Y. He, K. Lai, and Y. Xu. 2018. Matching article pairs with graphical decomposition and convolutions. arXiv:1802.07459. Retrieved from https://arxiv.org/abs/1802.07459
[331] P. Li, W. Li, Z. He, X. Wang, Y. Cao, J. Zhou, and W. Xu. 2016. Dataset and neural recurrent sequence labeling model for open-domain factoid question answering. arXiv:1607.06275. Retrieved from https://arxiv.org/abs/1607.06275
[332] N. Peng and M. Dredze. 2015. Named entity recognition for Chinese social media with jointly trained embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 548–554.
[333] W. Ling, D. Yogatama, C. Dyer, and P. Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. arXiv:1705.04146. Retrieved from https://arxiv.org/abs/1705.04146
[334] R. Weischedel, S. Pradhan, L. Ramshaw, M. Palmer, N. Xue, M. Marcus, A. Taylor, C. Greenberg, E. Hovy, R. Belvin, et al. 2011. *Ontonotes Release 4.0, LDC2011T03*. Linguistic Data Consortium, Philadelphia, PA.
[335] D. Vilares and C. Gómez-Rodríguez. 2019. HEAD-QA: A healthcare dataset for complex reasoning. arXiv:1906.04701. Retrieved from https://arxiv.org/abs/1906.04701
[336] S. L. Blodgett, L. Green, and B. O'Connor. 2016. Demographic dialectal variation in social media: A case study of African-American English. arXiv:1608.08868. Retrieved from https://arxiv.org/abs/1608.08868
[337] N. Mostafazadeh, N. Chambers, X. He, D. Parikh, D. Batra, L. Vanderwende, P. Kohli, and J. Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. arXiv:1604.01696. Retrieved from https://arxiv.org/abs/1604.01696
[338] D. Paperno, G. Kruszewski, A. Lazaridou, Q. N. Pham, R. Bernardi, S. Pezzelle, M. Baroni, G. Boleda, and R. Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. arXiv:1606.06031. Retrieved from https://arxiv.org/abs/1606.06031
[339] B. Hu, Q. Chen, and F. Zhu. 2015. LCSTS: A large scale Chinese short text summarization dataset. arXiv:1506.05865. Retrieved from https://arxiv.org/abs/1506.05865
[340] Z. Shao, M. Huang, J. Wen, W. Xu, and X. Zhu. 2019. Long and diverse text generation with planning-based hierarchical variational model. arXiv:1908.06605. Retrieved from https://arxiv.org/abs/1908.06605
[341] J. Novikova, O. Dušek, and V. Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. arXiv:1706.09254. Retrieved from https://arxiv.org/abs/1706.09254
[342] C. Zheng, M. Huang, and A. Sun. 2019. ChiD: A large-scale Chinese IDiom dataset for cloze test. arXiv:1906.01265. Retrieved from https://arxiv.org/abs/1906.01265
[343] Y. Bisk, R. Zellers, J. Gao, Y. Choi, et al. 2020. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34, 7432–7439.
[344] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. arXiv:1705.03551. Retrieved from https://arxiv.org/abs/1705.03551
[345] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. 2018. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. arXiv:1803.05457. Retrieved from https://arxiv.org/abs/1803.05457

[346] S. Aroca-Ouellette, C. Paik, A. Roncone, and K. Kann. 2021. PROST: Physical reasoning of objects through space and time. arXiv:2106.03634. Retrieved from https://arxiv.org/abs/2106.03634
[347] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal. 2018. Can a suit of armor conduct electricity? A new dataset for open book question answering. arXiv:1809.02789. Retrieved from https://arxiv.org/abs/1809.02789
[348] T. C. Ferreira, C. Gardent, N. Ilinykh, C. Van Der Lee, S. Mille, D. Moussallem, and A. Shimorina. 2020. The 2020 bilingual, bi-directional WebNLG+ shared task overview and evaluation results (WebNLG+). In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*.
[349] C. Xu, W. Zhou, T. Ge, K. Xu, J. McAuley, and F. Wei. 2021. Blow the dog whistle: A Chinese dataset for cant understanding with common sense and world knowledge. arXiv:2104.02704. Retrieved from https://arxiv.org/abs/2104.02704
[350] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy. 2017. RACE: Large-scale reading comprehension dataset from examinations. arXiv:1704.04683. Retrieved from https://arxiv.org/abs/1704.04683
[351] E. Choi, H. He, M. Iyyer, M. Yatskar, W-T Yih, Y. Choi, P. Liang, and L. Zettlemoyer. 2018. QuAC: Question answering in context. arXiv:1808.07036. Retrieved from https://arxiv.org/abs/1808.07036
[352] M. Geva, D. Khashabi, E. Segal, T. Khot, D. Roth, and J. Berant. 2021. Did aristotle use a laptop? A question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics* 9 (2021), 346–361.
[353] J. Boyd-Graber, B. Satinoff, H. He, and H. Daumé Iii. 2012. Besting the quiz master: Crowdsourcing incremental classification games. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 1290–1301.
[354] S. Zhang, X. Zhang, H. Wang, J. Cheng, P. Li, and Z. Ding. 2017. Chinese medical question answer matching using end-to-end character-level multi-scale CNNs. *Applied Sciences* 7, 8 (2017), 767.
[355] S. Zhang, X. Zhang, H. Wang, L. Guo, and S. Liu. 2018. Multi-scale attentive interaction networks for Chinese medical question answer selection. *IEEE Access* 6 (2018), 74061–74071.
[356] C. Xu, J. Pei, H. Wu, Y. Liu, and C. Li. 2020. MATINF: A jointly labeled large-scale dataset for classification, question answering and summarization. arXiv:2004.12302. Retrieved from https://arxiv.org/abs/2004.12302
[357] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi. 2021. WinoGrande: An adversarial Winograd schema challenge at scale. *Communications of the ACM* 64, 9 (2021), 99–106.
[358] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. 2019. HellaSwag: Can a machine really finish your sentence? arXiv:1905.07830. Retrieved from https://arxiv.org/abs/1905.07830
[359] M. Roemmele, C. A. Bejan, and A. S. Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *Proceedings of the AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, 90–95.
[360] H. Levesque, E. Davis, and L. Morgenstern. 2012. The Winograd schema challenge. In *Proceedings of the 13th International Conference on the Principles of Knowledge Representation and Reasoning*.
[361] A. Talmor, J. Herzig, N. Lourie, and J. Berant. 2018. CommonsenseQA: A question answering challenge targeting commonsense knowledge. arXiv:1811.00937. Retrieved from https://arxiv.org/abs/1811.00937
[362] M. Sap, H. Rashkin, D. Chen, R. LeBras, and Y. Choi. 2019. SocialIQA: Commonsense reasoning about social interactions. arXiv:1904.09728. Retrieved from https://arxiv.org/abs/1904.09728
[363] K. Sun, D. Yu, D. Yu, and C. Cardie. 2020. Investigating prior knowledge for challenging Chinese machine reading comprehension. *Transactions of the Association for Computational Linguistics* 8 (2020), 141–155.
[364] S. Zhang, X. Liu, J. Liu, J. Gao, K. Duh, and B. Van Durme. 2018. ReCoRD: Bridging the gap between human and machine commonsense reading comprehension. arXiv:1810.12885. Retrieved from https://arxiv.org/abs/1810.12885
[365] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. arXiv:1606.05250. Retrieved from https://arxiv.org/abs/1606.05250
[366] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. arXiv:1905.10044. Retrieved from https://arxiv.org/abs/1905.10044
[367] P. Rajpurkar, R. Jia, and P. Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. arXiv:1806.03822. Retrieved from https://arxiv.org/abs/1806.03822
[368] D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. arXiv:1903.00161. Retrieved from https://arxiv.org/abs/1903.00161
[369] I. Dagan, O. Glickman, and B. Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Proceedings of the Machine Learning Challenges Workshop*. Springer, 177–190.
[370] Y. Chang, M. Narang, H. Suzuki, G. Cao, J. Gao, and Y. Bisk. 2022. WebQA: Multihop and multimodal QA. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16495–16504.
[371] Y. Cui, T. Liu, Z. Chen, W. Ma, S. Wang, and G. Hu. 2017. Dataset for the first evaluation on Chinese machine reading comprehension. arXiv:1709.08299. Retrieved from https://arxiv.org/abs/1709.08299

[372] Y. Cui, T. Liu, W. Che, L. Xiao, Z. Chen, W. Ma, S. Wang, and G. Hu. 2018. A span-extraction dataset for Chinese machine reading comprehension. arXiv:1810.07366. Retrieved from https://arxiv.org/abs/1810.07366

[373] Y. Cui, T. Liu, Z. Yang, Z. Chen, W. Ma, W. Che, S. Wang, and G. Hu. 2020. A sentence cloze dataset for Chinese machine reading comprehension. arXiv:2004.03116. Retrieved from https://arxiv.org/abs/2004.03116

[374] Y. Li, T. Liu, D. Li, Q. Li, J. Shi, and Y. Wang. 2018. Character-based BiLSTM-CRF incorporating POS and dictionaries for Chinese opinion target extraction. In *Proceedings of the Asian Conference on Machine Learning*. PMLR, 518–533.

[375] D. Khashabi, S. Chaturvedi, M. Roth, S. Upadhyay, and D. Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Long Papers)*, Vol. 1, 252–262.

[376] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, et al. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics* 7 (2019), 453–466.

[377] C. C. Shao, T. Liu, Y. Lai, Y. Tseng, and S. Tsai. 2018. DRCD: A Chinese machine reading comprehension dataset. arXiv:1806.00920. Retrieved from https://arxiv.org/abs/1806.00920

[378] W. He, K. Liu, J. Liu, Y. Lyu, S. Zhao, X. Xiao, Y. Liu, Y. Wang, H. Wu, Q. She, et al. 2017. DuReader: A Chinese machine reading comprehension dataset from real-world applications. arXiv:1711.05073. Retrieved from https://arxiv.org/abs/1711.05073

[379] H. Tang, J. Liu, H. Li, Y. Hong, H. Wu, and H. Wang. 2020. $\text{DuReader}_{\text{robust}}$: A Chinese dataset towards evaluating the robustness of machine reading comprehension models. arXiv:2004.11142. Retrieved from https://arxiv.org/abs/2004.11142

[380] J. Welbl, N. F. Liu, and M. Gardner. 2017. Crowdsourcing multiple choice science questions. arXiv:1707.06209. Retrieved from https://arxiv.org/abs/1707.06209

[381] C. Xiong, Z. Dai, J. Callan, Z. Liu, and R. Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 55–64.

[382] A. Peñas, E. Hovy, P. Forner, Á. Rodrigo, R. Sutcliffe, and R. Morante. 2013. QA4MRE 2011-2013: Overview of question answering for machine reading evaluation. In *Proceedings of the 4th International Conference of the CLEF Initiative on Information Access Evaluation. Multilinguality, Multimodality, and Visualization (CLEF '13)*. Springer, 303–320.

[383] S. Lim, M. Kim, and J. Lee. 2019. KorQuAD1.0: Korean QA dataset for machine reading comprehension. arXiv:1909.07005. Retrieved from https://arxiv.org/abs/1909.07005

[384] C. Xiao, H. Zhong, Z. Guo, C. Tu, Z. Liu, M. Sun, Y. Feng, X. Han, Z. Hu, H. Wang, et al. 2018. CAIL2018: A large-scale legal dataset for judgment prediction. arXiv:1807.02478. Retrieved from https://arxiv.org/abs/1807.02478

[385] D. Hendrycks, S. Basart, S. Kadavath, M. Mazeika, A. Arora, E. Guo, C. Burns, S. Puranik, H. He, D. Song, et al. 2021. Measuring coding challenge competence with apps. arXiv:2105.09938. Retrieved from https://arxiv.org/abs/2105.09938

[386] Y. Wang, X. Liu, and S. Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 845–854.

[387] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. 2021. Training verifiers to solve math word problems. arXiv:2110.14168. Retrieved from https://arxiv.org/abs/2110.14168

[388] J. Austin, A. Odena, M. I. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. J. Cai, M. Terry, Q. V. Le, and C. Sutton. 2021. Program synthesis with large language models. arXiv:2108.07732. Retrieved from https://arxiv.org/abs/2108.07732

[389] F. Shi, M. Suzgun, M. Freitag, X. Wang, S. Srivats, S. Vosoughi, H. W. Chung, Y. Tay, S. Ruder, D. Zhou, et al. 2022. Language models are multilingual chain-of-thought reasoners. arXiv:2210.03057. Retrieved from https://arxiv.org/abs/2210.03057

[390] S. Roy and D. Roth. 2016. Solving general arithmetic word problems. arXiv:1608.01413. Retrieved from https://arxiv.org/abs/1608.01413

[391] S.-Y. Miao, C.-C. Liang, and K.-Y. Su. 2021. A diverse corpus for evaluating and developing English math word problem solvers. arXiv:2106.15772. Retrieved from https://arxiv.org/abs/2106.15772

[392] R. Koncel-Kedziorski, S. Roy, A. Amini, N. Kushman, and H. Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1152–1157.

[393] A. Patel, S. Bhattamishra, and N. Goyal. 2021. Are NLP models really able to solve simple math word problems? arXiv:2103.07191. Retrieved from https://arxiv.org/abs/2103.07191

[394] Y. Lai, C. Li, Y. Wang, T. Zhang, R. Zhong, L. Zettlemoyer, W-T Yih, D. Fried, S. Wang, and T. Yu. 2023. DS-1000: A natural and reliable benchmark for data science code generation. In *Proceedings of the International Conference on Machine Learning*, PMLR, 18319–18345.

[395] J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, et al. 2021. Program synthesis with large language models. arXiv:2108.07732. Retrieved from https://arxiv.org/abs/2108.07732

[396] Y. Nie, A. Williams, E. Dinan, M. Bansal, J. Weston, and D. Kiela. 2019. Adversarial NLI: A new benchmark for natural language understanding. arXiv:1910.14599. Retrieved from https://arxiv.org/abs/1910.14599
[397] A. Williams, N. Nangia, and S. R. Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. arXiv:1704.05426. Retrieved from https://arxiv.org/abs/1704.05426
[398] R. T. McCoy, E. Pavlick, and T. Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. arXiv:1902.01007. Retrieved from https://arxiv.org/abs/1902.01007
[399] J. Liu, L. Cui, H. Liu, D. Huang, Y. Wang, and Y. Zhang. 2020. LogiQA: A challenge dataset for machine reading comprehension with logical reasoning. arXiv:2007.08124. Retrieved from https://arxiv.org/abs/2007.08124
[400] P. Lewis, B. Oğuz, R. Rinott, S. Riedel, and H. Schwenk. 2019. MLQA: Evaluating cross-lingual extractive question answering. arXiv:1910.07475. Retrieved from https://arxiv.org/abs/1910.07475
[401] A. Conneau, G. Lample, R. Rinott, A. Williams, S. R. Bowman, H. Schwenk, and V. Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. arXiv:1809.05053. Retrieved from https://arxiv.org/abs/1809.05053
[402] Y. Yang, Y. Zhang, C. Tar, and J. Baldridge. 2019. PAWS-X: A cross-lingual adversarial dataset for paraphrase identification. arXiv:1908.11828. Retrieved from https://arxiv.org/abs/1908.11828
[403] S. Narayan, S. B. Cohen, and M. Lapata. 1808. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. arXiv:1808.08745. Retrieved from https://arxiv.org/abs/1808.08745
[404] E. M. Ponti, G. Glavaš, O. Majewska, Q. Liu, I. Vulić, and A. Korhonen. 2020. XCOPA: A multilingual dataset for causal commonsense reasoning. arXiv:2005.00333. Retrieved from https://arxiv.org/abs/2005.00333
[405] A. Tikhonov and M. Ryabinin. 2021. It's all in the heads: Using attention heads as a baseline for cross-lingual transfer in commonsense reasoning. arXiv:2106.12066. Retrieved from https://arxiv.org/abs/2106.12066
[406] J. H. Clark, E. Choi, M. Collins, D. Garrette, T. Kwiatkowski, V. Nikolaev, and J. Palomaki. 2020. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics* 8 (2020), 454–470.
[407] T. Scialom, P.-A. Dray, S. Lamprier, B. Piwowarski, and J. Staiano. 2020. MLSUM: The multilingual summarization corpus. arXiv:2004.14900. Retrieved from https://arxiv.org/abs/2004.14900
[408] S. Lin, J. Hilton, and O. Evans. 2021. TruthfulQA: Measuring how models mimic human falsehoods. arXiv:2109.07958. Retrieved from https://arxiv.org/abs/2109.07958
[409] I. Augenstein, C. Lioma, D. Wang, L. C. Lima, C. Hansen, C. Hansen, and J. G. Simonsen. 2019. MultiFC: A real-world multi-domain dataset for evidence-based fact checking of claims. arXiv:1909.03242. Retrieved from https://arxiv.org/abs/1909.03242
[410] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal. 2018. FEVER: A large-scale dataset for fact extraction and verification. arXiv:1803.05355. Retrieved from https://arxiv.org/abs/1803.05355
[411] I. Mollas, Z. Chrysopoulou, S. Karlos, and G. Tsoumakas. 2020. ETHOS: An online hate speech detection dataset. arXiv:2006.08328. Retrieved from https://arxiv.org/abs/2006.08328
[412] M. Nadeem, A. Bethke, and S. Reddy. 2020. StereoSet: Measuring stereotypical bias in pretrained language models. arXiv:2004.09456. Retrieved from https://arxiv.org/abs/2004.09456
[413] A. Parrish, A. Chen, N. Nangia, V. Padmakumar, J. Phang, J. Thompson, P. M. Htut, and S. R. Bowman. 2021. BBQ: A hand-built bias benchmark for question answering. arXiv:2110.08193. Retrieved from https://arxiv.org/abs/2110.08193
[414] J. Zhao, T. Wang, M. Yatskar, V. Ordonez, and K.-W. Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. arXiv:1804.06876. Retrieved from https://arxiv.org/abs/1804.06876
[415] N. Nangia, C. Vania, R. Bhalerao, and S. R. Bowman. 2020. Crows-pairs: A challenge dataset for measuring social biases in masked language models. arXiv:2010.00133. Retrieved from https://arxiv.org/abs/2010.00133
[416] S. Gehman, S. Gururangan, M. Sap, Y. Choi, and N. A. Smith. 2020. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. arXiv:2009.11462. Retrieved from https://arxiv.org/abs/2009.11462
[417] D. Borkan, L. Dixon, J. Sorensen, N. Thain, and L. Vasserman. 2019. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion Proceedings of the 2019 World Wide Web Conference*, 491–500.
[418] O. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, V. Logacheva, C. Monz, et al. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the 1st Conference on Machine Translation: Volume 2, Shared Task Papers*, 131–198.
[419] B. Loïc, B. Magdalena, B. Ondřej, F. Christian, G. Yvette, G. Roman, H. Barry, H. Matthias, J. Eric, K. Tom, et al. 2020. Findings of the 2020 conference on machine translation (WMT20). In *Proceedings of the 5th Conference on Machine Translation*. Association for Computational Linguistics, 1–55.
[420] W. Li, F. Qi, M. Sun, X. Yi, and J. Zhang. 2021. CCPM: A Chinese classical poetry matching dataset. arXiv:2106.01979. Retrieved from https://arxiv.org/abs/2106.01979
[421] E. Dinan, S. Roller, K. Shuster, A. Fan, M. Auli, and J. Weston. 2018. Wizard of Wikipedia: Knowledge-powered conversational agents. arXiv:1811.01241. Retrieved from https://arxiv.org/abs/1811.01241
[422] H. Rashkin, E. M. Smith, M. Li, and Y.-L. Boureau. 2018. Towards empathetic open-domain conversation models: A new benchmark and dataset. arXiv:1811.00207. Retrieved from https://arxiv.org/abs/1811.00207

[423] E. Dinan, V. Logacheva, V. Malykh, A. Miller, K. Shuster, J. Urbanek, D. Kiela, A. Szlam, I. Serban, R. Lowe, et al. 2020. The second conversational intelligence challenge (ConvAI2). In *The NeurIPS'18 Competition: From Machine Learning to Intelligent Conversations*. Springer, 187–208.

[424] H. Zhou, C. Zheng, K. Huang, M. Huang, and X. Zhu. 2020. KdConv: A Chinese multi-domain dialogue dataset towards multi-turn knowledge-driven conversation. arXiv:2004.04100. Retrieved from https://arxiv.org/abs/2004.04100

[425] L. Co. 2019. IFlytek: A multiple categories Chinese text classifier. Competition Official Website.

[426] J. Baumgartner, S. Zannettou, B. Keegan, M. Squire, and J. Blackburn. 2020. The Pushshift Reddit dataset. In *Proceedings of the International AAAI Conference on Web and Social Media*, Vol. 14, 830–839.

[427] A. Fan, Y. Jernite, E. Perez, D. Grangier, J. Weston, and M. Auli. 2019. ELI5: Long form question answering. arXiv:1907.09190. Retrieved from https://arxiv.org/abs/1907.09190

[428] Y. Wang, S. Mishra, P. Alipoormolabashi, Y. Kordi, A. Mirzaei, A. Arunkumar, A. Ashok, A. S. Dhanasekaran, A. Naik, D. Stap, et al. 2022. Benchmarking generalization via in-context instructions on 1,600+ language tasks. arXiv:2204.07705. Retrieved from https://arxiv.org/abs/2204.07705

[429] T. Xie, C. H. Wu, P. Shi, R. Zhong, T. Scholak, M. Yasunaga, C.-S. Wu, M. Zhong, P. Yin, S. I. Wang, et al. 2022. UnifiedSKG: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. arXiv:2201.05966. Retrieved from https://arxiv.org/abs/2201.05966

[430] Q. Ye, B. Y. Lin, and X. Ren. 2021. CrossFit: A few-shot learning challenge for cross-task generalization in NLP. arXiv:2104.08835. Retrieved from https://arxiv.org/abs/2104.08835

[431] V. Aribandi, Y. Tay, T. Schuster, J. Rao, H. S. Zheng, S. V. Mehta, H. Zhuang, V. Q. Tran D. Bahri, J. Ni et al. 2021. ExT5: Towards extreme multi-task scaling for transfer learning. arXiv:2111.10952. Retrieved from https://arxiv.org/abs/2111.10952

[432] A. Williams, N. Nangia, and S. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, 1112–1122. DOI: https://doi.org/10.18653/v1/N18-1101

[433] Y. Zhang, J. Baldridge, and L. He. 2019. PAWS: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 1298–1308. DOI: https://doi.org/10.18653/v1/N19-1131

[434] C. Qin, A. Zhang, Z. Zhang, J. Chen, M. Yasunaga, and D. Yang. 2023. Is ChatGPT a general-purpose natural language processing task solver? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Retrieved from https://openreview.net/forum?id=u03xn1COsO

[435] M. U. Hadi, R. Qureshi, A. Shah, M. Irfan, A. Zafar, M. B. Shaikh, N. Akhtar, J. Wu, S. Mirjalili, et al. 2023. Large language models: A comprehensive survey of its applications, challenges, limitations, and future prospects. *Authorea Preprints* 1 (2023), 1–26.

[436] X. L. Dong, S. Moon, Y. E. Xu, K. Malik, and Z. Yu. 2023. Towards next-generation intelligent assistants leveraging LLM techniques. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 5792–5793.

[437] K. Pandya and M. Holia. 2023. Automating customer service using LangChain: Building custom open-source GPT chatbot for organizations. arXiv:2310.05421. Retrieved from https://arxiv.org/abs/2310.05421

[438] J. Li, B. Hui, G. Qu, B. Li, J. Yang, B. Li, B. Wang, B. Qin, R. Cao, R. Geng, et al. 2023. Can LLM already serve as a database interface? A big bench for large-scale database grounded text-to-SQLs. arXiv:2305.03111. Retrieved from https://arxiv.org/abs/2305.03111

[439] A. Rao, J. Kim, M. Kamineni, M. Pang, W. Lie, and M. D. Succi. 2023. Evaluating ChatGPT as an adjunct for radiologic decision-making. *medRxiv*.

[440] M. Benary, X. D. Wang, M. Schmidt, D. Soll, G. Hilfenhaus, M. Nassir, C. Sigler, M. Knödler, U. Keller, D. Beule, et al. 2023. Leveraging large language models for decision support in personalized oncology. *JAMA Network Open* 6, 11 (2023), e2343689–e2343689.

[441] C. M. Chiesa-Estomba, J. R. Lechien, L. A. Vaira, A. Brunet, G. Cammaroto, M. Mayo-Yanez, A. Sanchez-Barrueco, and C. Saga-Gutierrez. 2024. Exploring the potential of Chat-GPT as a supportive tool for sialendoscopy clinical decision making and patient information support. *European Archives of Oto-Rhino-Laryngology* 281, 4 (2024), 2081–2086.

[442] S. Montagna, S. Ferretti, L. C. Klopfenstein, A. Florio, and M. F. Pengo. 2023. Data decentralisation of LLM-based chatbot systems in chronic disease self-management. In *Proceedings of the 2023 ACM Conference on Information Technology for Social Good*, 205–212.

[443] D. Bill and T. Eriksson. 2023. Fine-tuning a LLM using reinforcement learning from human feedback for a therapy chatbot application.

[444] M. Abbasian, I. Azimi, A. M. Rahmani, and R. Jain. 2023. Conversational health agents: A personalized LLM-powered agent framework. arXiv:2310.02374. Retrieved from https://arxiv.org/abs/2310.02374

[445] K. V. Lemley. 2024. Does ChatGPT help us understand the medical literature? *Journal of the American Society of Nephrology* 35, 2 (2024), 232–233.
[446] S. Pal, M. Bhattacharya, S.-S. Lee, and C. Chakraborty. 2024. A domain-specific next-generation large language model (LLM) or ChatGPT is required for biomedical engineering and research. *Annals of Biomedical Engineering* 52, 3 (2024), 451–454.
[447] Y. Du, S. Zhao, Y. Chen, R. Bai, J. Liu, H. Wu, H. Wang, and B. Qin. 2023. The CALLA dataset: Probing LLMs' interactive knowledge acquisition from Chinese medical literature. arXiv:2309.04198. Retrieved from https://arxiv.org/abs/2309.04198
[448] A. Abd-Alrazaq, R. AlSaad, D. Alhuwail, A. Ahmed, P. M. Healy, S. Latifi, S. Aziz, R. Damseh, S. A. Alrazak, and J. Sheikh. 2023. Large language models in medical education: Opportunities, challenges, and future directions. *JMIR Medical Education* 9, 1 (2023), e48291.
[449] A. B. Mbakwe, I. Lourentzou, L. A. Celi, O. J. Mechanic, and A. Dagan. 2023. ChatGPT passing USMLE shines a spotlight on the flaws of medical education.
[450] S. Ahn. 2023. The impending impacts of large language models on medical education. *Korean Journal of Medical Education* 35, 1 (2023), 103–107.
[451] E. Waisberg, J. Ong, M. Masalkhi, and A. G. Lee. 2024. Large language model (LLM)-driven chatbots for neuro-ophthalmic medical education. *Eye* 38, 4 (2024), 639–641.
[452] G. Deiana, M. Dettori, A. Arghittu, A. Azara, G. Gabutti, and P. Castiglia. 2023. Artificial intelligence and public health: Evaluating ChatGPT responses to vaccination myths and misconceptions. *Vaccines* 11, 7 (2023), 1217.
[453] L. De Angelis, F. Baglivo, G. Arzilli, G. P. Privitera, P. Ferragina, A. E. Tozzi, and C. Rizzo. 2023. ChatGPT and the rise of large language models: The new AI-driven infodemic threat in public health. *Frontiers in Public Health* 11 (2023), 1166120
[454] N. L. Rane, A. Tawde, S. P. Choudhary, and J. Rane. 2023. Contribution and performance of ChatGPT and other large language models (LLM) for scientific and research advancements: A double-edged sword. *International Research Journal of Modernization in Engineering Technology and Science* 5, 10 (2023), 875–899.
[455] W. Dai, J. Lin, H. Jin, T. Li, Y.-S. Tsai, D. Gašević, and G. Chen. 2023. Can large language models provide feedback to students? A case study on ChatGPT. In *Proceedings of the IEEE International Conference on Advanced Learning Technologies (ICALT)*. IEEE, 323–325.
[456] Enkelejda Kasneci, Kathrin Sessler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. 2023. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences* 103 (2023), 102274.
[457] N. Rane. 2023. Enhancing the quality of teaching and learning through ChatGPT and similar large language models: Challenges, future prospects, and ethical considerations in education, future prospects, and ethical considerations in education. *Future Prospects, and Ethical Considerations in Education* (15 September 2023).
[458] J. C. Young and M. Shishido. 2023. Investigating OpenAI's ChatGPT potentials in generating Chatbot's dialogue for English as a foreign language learning. *International Journal of Advanced Computer Science and Applications* 14, 6 (2023).
[459] J. Irons, C. Mason, P. Cooper, S. Sidra, A. Reeson, and C. Paris. 2023. Exploring the Impacts of ChatGPT on Future Scientific Work. *SocArXiv*.
[460] P. G. Schmidt and A. J. Meir. 2023. Using generative AI for literature searches and scholarly writing: Is the integrity of the scientific discourse in jeopardy? arXiv:2311.06981. Retrieved from https://arxiv.org/abs/2311.06981
[461] Y. Zheng, H. Y. Koh, J. Ju, A. T. Nguyen, L. T. May, G. I. Webb, and S. Pan. 2023. Large language models for scientific synthesis, inference and explanation. arXiv:2310.07984. Retrieved from https://arxiv.org/abs/2310.07984
[462] B. Aczel and E.-J. Wagenmakers. 2023. Transparency guidance for ChatGPT usage in scientific writing. *PsyArXiv*.
[463] S. Altmäe, A. Sola-Leyva, and A. Salumets. 2023. Artificial intelligence in scientific writing: A friend or a foe? *Reproductive Biomedicine Online* 47, 1 (2023), 3–9.
[464] S. Imani, L. Du, and H. Shrivastava. 2023. MathPrompter: Mathematical reasoning using large language models. arXiv:2303.05398. Retrieved from https://arxiv.org/abs/2303.05398
[465] Z. Yuan, H. Yuan, C. Li, G. Dong, C. Tan, and C. Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. arXiv:2308.01825. Retrieved from https://arxiv.org/abs/2308.01825
[466] K. Yang, A. M. Swope, A. Gu, R. Chalamala, P. Song, S. Yu, S. Godil, R. Prenger, and A. Anandkumar. 2023. LeanDojo: Theorem proving with retrieval-augmented language models. arXiv:2306.15626. Retrieved from https://arxiv.org/abs/2306.15626
[467] K. M. Collins, A. Q. Jiang, S. Frieder, L. Wong, M. Zilka, U. Bhatt, T. Lukasiewicz, Y. Wu, J. B. Tenenbaum, W. Hart, et al. 2023. Evaluating language models for mathematics through interactions. arXiv:2306.01694. Retrieved from https://arxiv.org/abs/2306.01694
[468] Y. Liu, T. Han, S. Ma, J. Zhang, Y. Yang, J. Tian, H. He, A. Li, M. He, Z. Liu, et al. 2023. Summary of ChatGPT-related research and perspective towards the future of large language models. *Meta-Radiology* 1, 2 (2023), 100017.

[469] J. Drápal, H. Westermann, and J. Savelka. 2023. Using large language models to support thematic analysis in empirical legal studies. arXiv:2310.18729. Retrieved from https://arxiv.org/abs/2310.18729
[470] J. Savelka, K. D. Ashley, M. A. Gray, H. Westermann, and H. Xu. 2023. Explaining legal concepts with augmented large language models (GPT-4). arXiv:2306.09525. Retrieved from https://arxiv.org/abs/2306.09525
[471] N. Guha, J. Nyarko, D. E. Ho, C. Ré, A. Chilton, A. Narayana, A. Chohlas-Wood, A. Peters, B. Waldon, D. N. Rockmore, et al. 2023. LegalBench: A collaboratively built benchmark for measuring legal reasoning in large language models. arXiv:2308.11462. Retrieved from https://arxiv.org/abs/2308.11462
[472] J. Cui, Z. Li, Y. Yan, B. Chen, and L. Yuan. 2023. Chatlaw: Open-source legal large language model with integrated external knowledge bases. arXiv:2306.16092. Retrieved from https://arxiv.org/abs/2306.16092
[473] H. Yang, X.-Y. Liu, and C. D. Wang. 2023. FinGPT: Open-source financial large language models. arXiv:2306.06031. Retrieved from https://arxiv.org/abs/2306.06031
[474] Y. Li, S. Wang, H. Ding, and H. Chen. 2023. Large language models in finance: A survey. In *Proceedings of the 4th ACM International Conference on AI in Finance*, 374–382.
[475] A. Lykov and D. Tsetserukou. 2023. LLM-BRAIn: AI-driven fast generation of robot behaviour tree based on large language model. arXiv:2305.19352. Retrieved from https://arxiv.org/abs/2305.19352
[476] E. Billing, J. Rosén, and M. Lamb. 2023. Language models for human-robot interaction. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 905–906.
[477] Y. Ye, H. You, and J. Du. 2023. Improved trust in human-robot collaboration with ChatGPT. *IEEE Access* 11 (2023), 55748–55754.
[478] Y. Ding, X. Zhang, C. Paxton, and S. Zhang. 2023. Leveraging commonsense knowledge from large language models for task and motion planning. In *Proceedings of the RSS 2023 Workshop on Learning for Task and Motion Planning*
[479] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser. 2023. TidyBot: Personalized robot assistance with large language models. arXiv:2305.05658. Retrieved from https://arxiv.org/abs/2305.05658
[480] Z. Liu, C. Zhao, F. Iandola, C. Lai, Y. Tian, I. Fedorov, Y. Xiong, E. Chang, Y. Shi, R. Krishnamoorthi, et al. 2024. MobileLLM: Optimizing sub-billion parameter language models for on-device use cases. In *Proceedings of the 41st International Conference on Machine Learning*.
[481] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, 611–626.
[482] A. Agrawal, N. Kedia, A. Panwar, J. Mohan, N. Kwatra, B. Gulavani, A. Tumanov, and R. Ramjee. 2024. Taming throughput-latency tradeoff in LLM inference with Sarathi-Serve. In *Proceedings of the 18th USENIX symposium on Operating Systems Design and Implementation (ODSI '24)*, 117–134.
[483] G.-I. Yu, J. S. Jeong, G.-W. Kim, S. Kim, and B.-G. Chun. 2022. ORCA: A distributed serving system for transformer-based generative models. In *Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI '22)*, 521–538.
[484] Y. Jiang, F. Fu, X. Yao, G. He, X. Miao, A. Klimovic, B. Cui, B. Yuan, and E. Yoneki. 2025. Demystifying cost-efficiency in LLM serving over heterogeneous GPUs. arXiv:2502.00722. Retrieved from https://arxiv.org/abs/2502.00722
[485] E. Ren. 2024. Task scheduling for decentralized LLM serving in heterogeneous networks.
[486] Y. Jiang, F. Fu, X. Yao, T. Wang, B. Cui, A. Klimovic, and E. Yoneki. 2025. ThunderServe: High-performance and cost-efficient LLM serving in cloud environments. arXiv:2502.09334. Retrieved from https://arxiv.org/abs/2502.09334
[487] R. Qin, D. Liu, C. Xu, Z. Yan, Z. Tan, Z. Jia, A. Nassereldine, J. Li, M. Jiang, A. Abbasi, et al. 2024. Empirical guidelines for deploying LLMs onto resource-constrained edge devices. arXiv:2406.03777. Retrieved from https://arxiv.org/abs/2406.03777
[488] M. Zhang, X. Shen, J. Cao, Z. Cui, and S. Jiang. 2025. EdgeShard: Efficient LLM inference via collaborative edge computing. *IEEE Internet of Things Journal* 12, 10 (2025), 13119–13131.
[489] J. Haris, R. Saha, W. Hu, and J. Cano. 2024. Designing efficient LLM accelerators for edge devices. arXiv:2408.00462. Retrieved from https://arxiv.org/abs/2408.00462
[490] E. Strubell, A. Ganesh, and A. McCallum. 2019. Energy and policy considerations for deep learning in NLP. arXiv:1906.02243. Retrieved from https://arxiv.org/abs/1906.02243
[491] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 610–623.
[492] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. 2021. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM* 64, 3 (2021), 107–115.
[493] M. Tänzer, S. Ruder, and M. Rei. 2021. Memorisation versus generalisation in pre-trained language models. arXiv:2105.00828. Retrieved from https://arxiv.org/abs/2105.00828

[494] S. M. West, M. Whittaker, and K. Crawford. 2019. Discriminating systems. *AI Now* (2019), 1–33.
[495] K. Valmeekam, A. Olmo, S. Sreedharan, and S. Kambhampati. 2022. Large language models still can't plan (a benchmark for LLMs on planning and reasoning about change). arXiv:2206.10498. Retrieved from https://arxiv.org/abs/2206.10498
[496] Y. Zhang, Y. Li, L. Cui, D. Cai, L. Liu, T. Fu, X. Huang, E. Zhao, Y. Zhang, Y. Chen, et al. 2023. Siren's song in the AI ocean: A survey on hallucination in large language models. arXiv:2309.01219. Retrieved from https://arxiv.org/abs/2309.01219
[497] A. Webson and E. Pavlick. 2021. Do prompt-based models really understand the meaning of their prompts? arXiv:2109.01247. Retrieved from https://arxiv.org/abs/2109.01247
[498] S. Chen, A. Zharmagambetov, S. Mahloujifar, K. Chaudhuri, and C. Guo. 2024. Aligning LLMs to be robust against prompt injection. arXiv:2410.05451. Retrieved from https://arxiv.org/abs/2410.05451
[499] X. Liu, H. Cheng, P. He, W. Chen, Y. Wang, H. Poon, and J. Gao. 2020. Adversarial Training for Large Neural Language Models. Retrieved April 2020 from https://www.microsoft.com/en-us/research/publication/adversarial-training-for-large-neural-language-models/
[500] X. Xu, K. Kong, N. Liu, L. Cui, D. Wang, J. Zhang, and M. Kankanhalli. 2023. An LLM can fool itself: A prompt-based adversarial attack. arXiv:2310.13345. Retrieved from https://arxiv.org/abs/2310.13345
[501] O. Shaikh, H. Zhang, W. Held, M. Bernstein, and D. Yang. 2022. On second thought, let's not think step by step! Bias and toxicity in zero-shot reasoning. arXiv:2212.08061. Retrieved from https://arxiv.org/abs/2212.08061
[502] B. C. Das, M. H. Amini, and Y. Wu. 2024. Security and privacy challenges of large language models: A survey. arXiv:2402.00888. Retrieved from https://arxiv.org/abs/2402.00888
[503] E. Shayegani, M. A. A. Mamun, Y. Fu, P. Zaree, Y. Dong, and N. Abu-Ghazaleh. 2023. Survey of vulnerabilities in large language models revealed by adversarial attacks. arXiv:2310.10844. Retrieved from https://arxiv.org/abs/2310.10844
[504] H. Zhao, H. Chen, F. Yang, N. Liu, H. Deng, H. Cai, S. Wang, D. Yin, and M. Du. 2023. Explainability for large language models: A survey. arXiv:2309.01029. Retrieved from https://arxiv.org/abs/2309.01029
[505] S. Huang, S. Mamidanna, S. Jangam, Y. Zhou, and L. H. Gilpin. 2023. Can large language models explain themselves? A study of LLM-generated self-explanations. arXiv:2310.11207. Retrieved from https://arxiv.org/abs/2310.11207
[506] H. Brown, K. Lee, F. Mireshghallah, R. Shokri, and F. Tramèr. 2022. What does it mean for a language model to preserve privacy? In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, 2280–2292.
[507] R. Plant, V. Giuffrida, and D. Gkatzia. 2022. You are what you write: Preserving privacy in the era of large language models. arXiv:2204.09391. Retrieved from https://arxiv.org/abs/2204.09391
[508] W. Niu, Z. Kong, G. Yuan, W. Jiang, J. Guan, C. Ding, P. Zhao, S. Liu, B. Ren, and Y. Wang. 2020. Real-time execution of large-scale language models on mobile. arXiv:2009.06823. Retrieved from https://arxiv.org/abs/2009.06823
[509] C. Guo, J. Tang, W. Hu, J. Leng, C. Zhang, F. Yang, Y. Liu, M. Guo, and Y. Zhu. 2023. Olive: Accelerating large language models via hardware-friendly outlier-victim pair quantization In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 1–15.
[510] B. Meskó and E. J. Topol. 2023. The imperative for regulatory oversight of large language models (or generative AI) in healthcare. *NPJ Digital Medicine* 6, 1 (2023), 120.
[511] J. Zhang, X. Ji, Z. Zhao, X. Hei, and K.-K. R. Choo. 2023. Ethical considerations and policy implications for large language models: Guiding responsible development and deployment. arXiv:2308.02678. Retrieved from https://arxiv.org/abs/2308.02678
[512] J. Mökander, J. Schuett, H. R. Kirk, and L. Floridi. 2024. Auditing large language models: A three-layered approach. *AI and Ethics* 4, 4 (2024), 1085–1115.
[513] J. FitzGerald, S. Ananthakrishnan, K. Arkoudas, D. Bernardi, A. Bhagia, C. Delli Bovi, J. Cao, R. Chada, A. Chauhan, L. Chen, et al. 2022. Alexa teacher model: Pretraining and distilling multi-billion-parameter encoders for natural language understanding systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2893–2902.

Received 30 October 2024; revised 3 April 2025; accepted 26 May 2025