A Survey of Large Language Models

Wayne Xin Zhao, Kun Zhou*, Junyi Li*, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie and Ji-Rong Wen

Abstract—Ever since the Turing Test was proposed in the 1950s, humans have explored the mastering of language intelligence by machine. Language is essentially a complex, intricate system of human expressions governed by grammatical rules. It poses a significant challenge to develop capable artificial intelligence (AI) algorithms for comprehending and grasping a language. As a major approach, *language modeling* has been widely studied for language understanding and generation in the past two decades, evolving from statistical language models to neural language models. Recently, pre-trained language models (PLMs) have been proposed by pre-training Transformer models over large-scale corpora, showing strong capabilities in solving various natural language processing (NLP) tasks. Since the researchers have found that model scaling can lead to an improved model capacity, they further investigate the scaling effect by increasing the parameter scale to an even larger size. Interestingly, when the parameter scale exceeds a certain level, these enlarged language models not only achieve a significant performance improvement, but also exhibit some special abilities (*e.g.,* in-context learning) that are not present in small-scale language models (*e.g.,* BERT). To discriminate the language models in different parameter scales, the research community has coined the term *large language models (LLM)* for the PLMs of significant size (*e.g.,* containing tens or hundreds of billions of parameters). Recently, the research on LLMs has been largely advanced by both academia and industry, and a remarkable progress is the launch of ChatGPT (a powerful AI chatbot developed based on LLMs), which has attracted widespread attention from society. The technical evolution of LLMs has been making an important impact on the entire AI community, which would revolutionize the way how we develop and use AI algorithms. Considering this rapid technical progress, in this survey, we review the recent advances of LLMs by introducing the background, key findings, and mainstream techniques. In particular, we focus on four major aspects of LLMs, namely pre-training, adaptation tuning, utilization, and capacity evaluation. Furthermore, we also summarize the available resources for developing LLMs and discuss the remaining issues for future directions. This survey provides an up-to-date review of the literature on LLMs, which can be a useful resource for both researchers and engineers.

Index Terms—Large Language Models; Emergent Abilities; Adaptation Tuning; Utilization; Alignment; Capacity Evaluation

1 Introduction

"The limits of my language mean the limits of my world."
—Ludwig Wittgenstein

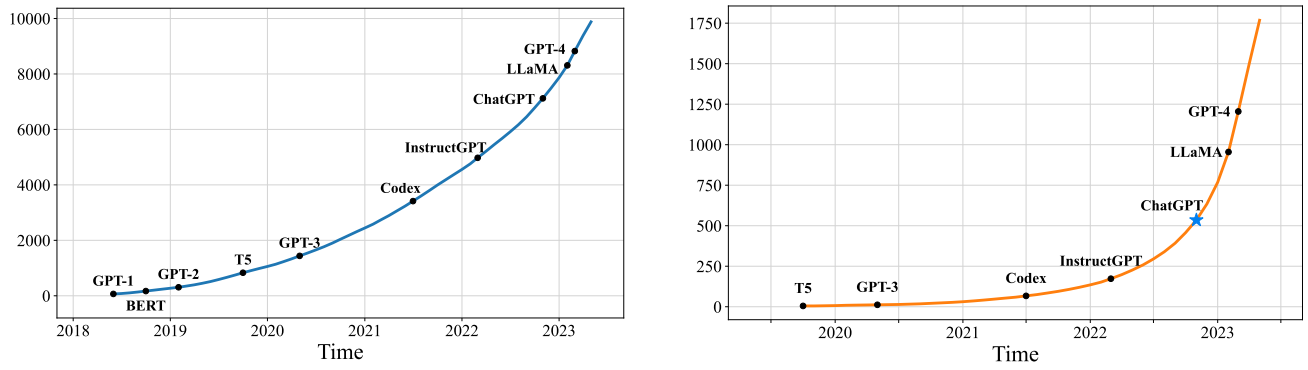
Language is a prominent ability in human beings to express and communicate, which develops in early childhood and evolves over a lifetime [1, 2]. Machines, however, cannot naturally grasp the abilities of understanding and communicating in the form of human language, unless equipped with powerful artificial intelligence (AI) algorithms. It has been a longstanding research challenge to achieve this goal, to enable machines to read, write, and communicate like humans [3].

Technically, *language modeling (LM)* is one of the major approaches to advancing language intelligence of machines. In general, LM aims to model the generative likelihood of word sequences, so as to predict the probabilities of future (or missing) tokens. The research of LM has received extensive attention in the literature, which can be divided into four major development stages:

• *Statistical language models (SLM)*. SLMs [4–7] are developed based on *statistical learning* methods that rose in the 1990s. The basic idea is to build the word prediction model based on the Markov assumption, *e.g.,* predicting the next word based on the most recent context. The SLMs with a fixed context length n are also called n-gram language models, *e.g.,* bigram and trigram language models. SLMs have been widely applied to enhance task performance in information retrieval (IR) [8, 9] and natural language processing (NLP) [10–12]. However, they often suffer from the curse of dimensionality: it is difficult to accurately estimate high-order language models since an exponential number of transition probabilities need to be estimated. Thus, specially designed smoothing strategies such as back-off estimation [13] and Good–Turing estimation [14] have been introduced to alleviate the data sparsity problem.

• *Neural language models (NLM)*. NLMs [15–17] characterize the probability of word sequences by neural networks, *e.g.,* recurrent neural networks (RNNs). As a remarkable contribution, the work in [15] introduced the concept of *distributed representation* of words and built the word prediction function conditioned on the aggregated context features (*i.e.,* the distributed word vectors). By extending the idea of learning effective features for words or sentences, a general neural network approach was developed to build

• *Version: v12 (major update on September 10, 2023).*
• *GitHub link: https://github.com/RUCAIBox/LLMSurvey*
• *Chinese version link: https://github.com/RUCAIBox/LLMSurvey/blob/main/assets/LLM_Survey_Chinese.pdf*
• ** K. Zhou and J. Li contribute equally to this work.*
• *The authors are mainly with Gaoling School of Artificial Intelligence and School of Information, Renmin University of China, Beijing, China; Jian-Yun Nie is with DIRO, Université de Montréal, Canada.*
Contact e-mail: batmanfly@gmail.com



(a) Query="Language Model"

(b) Query="Large Language Model"

Fig. 1: The trends of the cumulative numbers of arXiv papers that contain the keyphrases "*language model*" (since June 2018) and "*large language model*" (since October 2019), respectively. The statistics are calculated using exact match by querying the keyphrases in title or abstract by months. We set different x-axis ranges for the two keyphrases, because "language models" have been explored at an earlier time. We label the points corresponding to important landmarks in the research progress of LLMs. A sharp increase occurs after the release of ChatGPT: the average number of published arXiv papers that contain "*large language model*" in title or abstract goes from 0.40 per day to 8.58 per day (Figure 1(b)).

a unified solution for various NLP tasks [18]. Further, word2vec [19, 20] was proposed to build a simplified shallow neural network for learning distributed word representations, which were demonstrated to be very effective across a variety of NLP tasks. These studies have initiated the use of language models for representation learning (beyond word sequence modeling), having an important impact on the field of NLP.

- *Pre-trained language models (PLM)*. As an early attempt, ELMo [21] was proposed to capture context-aware word representations by first pre-training a bidirectional LSTM (biLSTM) network (instead of learning fixed word representations) and then fine-tuning the biLSTM network according to specific downstream tasks. Further, based on the highly parallelizable Transformer architecture [22] with self-attention mechanisms, BERT [23] was proposed by pre-training bidirectional language models with specially designed pre-training tasks on large-scale unlabeled corpora. These pre-trained context-aware word representations are very effective as general-purpose semantic features, which have largely raised the performance bar of NLP tasks. This study has inspired a large number of follow-up work, which sets the "*pre-training* and *fine-tuning*" learning paradigm. Following this paradigm, a great number of studies on PLMs have been developed, introducing either different architectures [24, 25] (*e.g.*, GPT-2 [26] and BART [24]) or improved pre-training strategies [27–29]. In this paradigm, it often requires fine-tuning the PLM for adapting to different downstream tasks.

- *Large language models (LLM)*. Researchers find that scaling PLM (*e.g.*, scaling model size or data size) often leads to an improved model capacity on downstream tasks (*i.e.*, following the scaling law [30]). A number of studies have explored the performance limit by training an ever larger PLM (*e.g.*, the 175B-parameter GPT-3 and the 540B-parameter PaLM). Although scaling is mainly conducted in model size (with similar architectures and pre-training tasks), these large-sized PLMs display different behaviors from smaller PLMs (*e.g.*, 330M-parameter BERT and 1.5B-parameter GPT-2) and show surprising abilities (called *emergent abilities* [31]) in solving a series of complex tasks. For example, GPT-3 can solve few-shot tasks through *in-context learning*, whereas GPT-2 cannot do well. Thus, the research community coins the term "*large language models (LLM)*"[1] for these large-sized PLMs [32–35], which attract increasing research attention (See Figure 1). A remarkable application of LLMs is *ChatGPT*[2] that adapts the LLMs from the GPT series for dialogue, which presents an amazing conversation ability with humans. We can observe a sharp increase of the arXiv papers that are related to LLMs after the release of ChatGPT in Figure 1.

In the existing literature, PLMs have been widely discussed and surveyed [36–39], while LLMs are seldom reviewed in a systematic way. To motivate our survey, we first highlight three major differences between LLMs and PLMs. First, LLMs display some surprising emergent abilities that may not be observed in previous smaller PLMs. These abilities are key to the performance of language models on complex tasks, making AI algorithms unprecedently powerful and effective. Second, LLMs would revolutionize the way that humans develop and use AI algorithms. Unlike small PLMs, the major approach to accessing LLMs is through the prompting interface (*e.g.*, GPT-4 API). Humans have to understand how LLMs work and format their tasks in a way that LLMs can follow. Third, the development of LLMs no longer draws a clear distinction between research and engineering. The training of LLMs requires extensive practical experiences in large-scale data processing and distributed parallel training. To develop capable LLMs, researchers have to solve complicated engineering issues, working with engineers or being engineers.

Nowadays, LLMs are posing a significant impact on the AI community, and the advent of ChatGPT and GPT-4

1. Note that a LLM is not necessarily more capable than a small PLM, and emergent abilities may not occur in some LLMs.
2. https://openai.com/blog/chatgpt/

leads to the rethinking of the possibilities of artificial general intelligence (AGI). OpenAI has published a technical article entitled "*Planning for AGI and beyond*", which discusses the short-term and long-term plans to approach AGI [40], and a more recent paper has argued that GPT-4 might be considered as an early version of an AGI system [41]. The research areas of AI are being revolutionized by the rapid progress of LLMs. In the field of NLP, LLMs can serve as a general-purpose language task solver (to some extent), and the research paradigm has been shifting towards the use of LLMs. In the field of IR, traditional search engines are challenged by the new information seeking way through AI chatbots (*i.e.,* ChatGPT), and *New Bing*[3] presents an initial attempt that enhances the search results based on LLMs. In the field of CV, the researchers try to develop ChatGPT-like vision-language models that can better serve multimodal dialogues [42–45], and GPT-4 [46] has supported multimodal input by integrating the visual information. This new wave of technology would potentially lead to a prosperous ecosystem of real-world applications based on LLMs. For instance, Microsoft 365 is being empowered by LLMs (*i.e.,* Copilot) to automate the office work, and OpenAI supports the use of plugins in ChatGPT for implementing special functions.

Despite the progress and impact, the underlying principles of LLMs are still not well explored. Firstly, it is mysterious why emergent abilities occur in LLMs, instead of smaller PLMs. As a more general issue, there lacks a deep, detailed investigation of the key factors that contribute to the superior abilities of LLMs. It is important to study when and how LLMs obtain such abilities [47]. Although there are some meaningful discussions about this problem [31, 47], more principled investigations are needed to uncover the "*secrets*" of LLMs. Secondly, it is difficult for the research community to train capable LLMs. Due to the huge demand of computation resources, it is very costly to carry out repetitive, ablating studies for investigating the effect of various strategies for training LLMs. Indeed, LLMs are mainly trained by industry, where many important training details (*e.g.,* data collection and cleaning) are not revealed to the public. Thirdly, it is challenging to align LLMs with human values or preferences. Despite the capacities, LLMs are also likely to produce toxic, fictitious, or harmful contents. It requires effective and efficient control approaches to eliminating the potential risk of the use of LLMs [46].

Faced with both opportunities and challenges, it needs more attention on the research and development of LLMs. In order to provide a basic understanding of LLMs, this survey conducts a literature review of the recent advances in LLMs from four major aspects, including *pre-training* (how to pre-train a capable LLM), *adaptation* (how to effectively adapt pre-trained LLMs for better use), *utilization* (how to use LLMs for solving various downstream tasks) and *capability evaluation* (how to evaluate the abilities of LLMs and existing empirical findings). We thoroughly comb the literature and summarize the key findings, techniques, and methods of LLMs. For this survey, we also create a GitHub project website by collecting the supporting resources for LLMs, at the link https://github.com/RUCAIBox/LLMSurvey. We are also aware of several related review articles on PLMs or LLMs [32, 36, 38, 39, 43, 48–54]. These papers either discuss PLMs or some specific (or general) aspects of LLMs. Compared with them, we focus on the techniques and methods to develop and use LLMs and provide a relatively comprehensive reference to important aspects of LLMs.

The remainder of this survey is organized as follows: Section 2 introduces the background for LLMs and the evolution of GPT-series models, followed by the summarization of available resources for developing LLMs in Section 3. Sections 4, 5, 6, and 7 review and summarize the recent progress from the four aspects of pre-training, adaptation, utilization, and capacity evaluation, respectively. Then, Section 8 discusses the practical guide for prompt design, and Section 9 reviews the applications of LLMs in several representative domains. Finally, we conclude the survey in Section 10 by summarizing the major findings and discuss the remaining issues for future work.

2 Overview

In this section, we present an overview about the background of LLMs and then summarize the technical evolution of the GPT-series models.

2.1 Background for LLMs

Typically, *large language models* (LLMs) refer to Transformer language models that contain hundreds of billions (or more) of parameters[4], which are trained on massive text data [32], such as GPT-3 [55], PaLM [56], Galactica [35], and LLaMA [57]. LLMs exhibit strong capacities to understand natural language and solve complex tasks (via text generation). To have a quick understanding of how LLMs work, this part introduces the basic background for LLMs, including scaling laws, emergent abilities and key techniques.

Scaling Laws for LLMs. Currently, LLMs are mainly built upon the Transformer architecture [22], where multi-head attention layers are stacked in a very deep neural network. Existing LLMs adopt similar Transformer architectures and pre-training objectives (*e.g.,* language modeling) as small language models. However, LLMs significantly extend the model size, data size, and total compute (orders of magnification). Extensive research has shown that scaling can largely improve the model capacity of LLMs [26, 55, 56]. Thus, it is useful to establish a quantitative approach to characterizing the scaling effect. Next, we introduce two representative scaling laws for Transformer language models [30, 34].

• *KM scaling law*[5]. In 2020, Kaplan et al. [30] (the OpenAI team) firstly proposed to model the power-law relationship of model performance with respective to three major factors, namely model size (N), dataset size (D), and the amount of

3. https://www.bing.com/new

4. In existing literature, there is no formal consensus on the minimum parameter scale for LLMs, since the model capacity is also related to data size and total compute. In this survey, we take a slightly loose definition of LLMs, and mainly focus on discussing language models with a model size larger than 10B.

5. Since there was not a model trained following this law in the original paper, we took the last names of the two co-first authors to name this scaling law.

training compute (C), for neural language models. Given a compute budget c, they empirically presented three basic formulas for the scaling law[6]:

$$\begin{aligned} L(N) &= \left(\frac{N_c}{N}\right)^{\alpha_N}, \quad \alpha_N \sim 0.076, N_c \sim 8.8 \times 10^{13} \quad (1) \\ L(D) &= \left(\frac{D_c}{D}\right)^{\alpha_D}, \quad \alpha_D \sim 0.095, D_c \sim 5.4 \times 10^{13} \\ L(C) &= \left(\frac{C_c}{C}\right)^{\alpha_C}, \quad \alpha_C \sim 0.050, C_c \sim 3.1 \times 10^{8} \end{aligned}$$

where $L(\cdot)$ denotes the cross entropy loss in nats. The three laws were derived by fitting the model performance with varied data sizes (22M to 23B tokens), model sizes (768M to 1.5B non-embedding parameters) and training compute, under some assumptions (*e.g.,* the analysis of one factor should be not bottlenecked by the other two factors). They showed that the model performance has a strong dependence relation on the three factors.

• *Chinchilla scaling law*. As another representative study, Hoffmann et al. [34] (the Google DeepMind team) proposed an alternative form for scaling laws to instruct the compute-optimal training for LLMs. They conducted rigorous experiments by varying a larger range of model sizes (70M to 16B) and data sizes (5B to 500B tokens), and fitted a similar scaling law yet with different coefficients as below [34]:

$$L(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}, \quad (2)$$

where $E = 1.69, A = 406.4, B = 410.7$, $\alpha = 0.34$ and $\beta = 0.28$. By optimizing the loss $L(N, D)$ under the constraint $C \approx 6ND$, they showed that the optimal allocation of compute budget to model size and data size can be derived as follows:

$$N_{opt}(C) = G\left(\frac{C}{6}\right)^{a}, \quad D_{opt}(C) = G^{-1}\left(\frac{C}{6}\right)^{b}, \quad (3)$$

where $a = \frac{\alpha}{\alpha+\beta}$, $b = \frac{\beta}{\alpha+\beta}$ and G is a scaling coefficient that can be computed by A, B, α and β. As analyzed in [34], given an increase in compute budget, the KM scaling law favors a larger budget allocation in model size than the data size, while the Chinchilla scaling law argues that the two sizes should be increased in equal scales, *i.e.,* having similar values for a and b in Equation (3).

Though with some restricted assumptions, these scaling laws provide an intuitive understanding of the scaling effect, making it feasible to predict the performance of LLMs during training [46]. However, some abilities (*e.g.,* in-context learning [55]) are unpredictable according to the scaling law, which can be observed only when the model size exceeds a certain level (as discussed below).

Emergent Abilities of LLMs. In the literature [31], *emergent abilities* of LLMs are formally defined as "the abilities that are not present in small models but arise in large models", which is one of the most prominent features that distinguish LLMs from previous PLMs. It further introduces a notable characteristic when emergent abilities occur [31]: performance rises significantly above random when the scale reaches a certain level. By analogy, such an emergent pattern has close connections with the phenomenon of *phase transition* in physics [31, 58]. In principle, emergent abilities can be defined in relation to some complex tasks [31, 59], while we are more concerned with general abilities that can be applied to solve a variety of tasks. Here, we briefly introduce three typical emergent abilities for LLMs and representative models that possess such an ability[7].

• *In-context learning*. The in-context learning (ICL) ability is formally introduced by GPT-3 [55]: assuming that the language model has been provided with a natural language instruction and/or several task demonstrations, it can generate the expected output for the test instances by completing the word sequence of input text, without requiring additional training or gradient update[8]. Among the GPT-series models, the 175B GPT-3 model exhibited a strong ICL ability in general, but not the GPT-1 and GPT-2 models. Such an ability also depends on the specific downstream task. For example, the ICL ability can emerge on the arithmetic tasks (*e.g.,* the 3-digit addition and subtraction) for the 13B GPT-3, but 175B GPT-3 even cannot work well on the Persian QA task [31].

• *Instruction following*. By fine-tuning with a mixture of multi-task datasets formatted via natural language descriptions (called *instruction tuning*), LLMs are shown to perform well on unseen tasks that are also described in the form of instructions [28, 61, 62]. With instruction tuning, LLMs are enabled to follow the task instructions for new tasks without using explicit examples, thus having an improved generalization ability. According to the experiments in [62], instruction-tuned LaMDA-PT [63] started to significantly outperform the untuned one on unseen tasks when the model size reached 68B, but not for 8B or smaller model sizes. A recent study [64] found that a model size of 62B is at least required for PaLM to perform well on various tasks in four evaluation benchmarks (*i.e.,* MMLU, BBH, TyDiQA and MGSM), though a much smaller size might suffice for some specific tasks (*e.g.,* MMLU).

• *Step-by-step reasoning*. For small language models, it is usually difficult to solve complex tasks that involve multiple reasoning steps, *e.g.,* mathematical word problems. In contrast, with the chain-of-thought (CoT) prompting strategy [33], LLMs can solve such tasks by utilizing the prompting mechanism that involves intermediate reasoning steps for deriving the final answer. This ability is speculated to be potentially obtained by training on code [33, 47]. An empirical study [33] has shown that CoT prompting can bring performance gains (on arithmetic reasoning benchmarks) when applied to PaLM and LaMDA variants with a model size larger than 60B, while its advantage over

6. Here, N_c, D_c and C_c are measured in the number of non-embedding parameters, the number of training tokens and the number of FP-days, respectively. According to the original paper [30], C_c and C should be denoted by C_c^{min} and C_{min}, corresponding to the optimal use of compute. We use the simplified notations for ease of discussions.

7. It is difficult to accurately examine the critical size for emergent abilities of LLMs (*i.e.,* the minimum size to possess an ability), since it might vary for different models or tasks. Also, existing studies often test emergent abilities on very limited model sizes for a specific LLM. For example, PaLM is often tested with three sizes of 8B, 62B and 540B. It is unclear about the model performance of the untested sizes.

8. In a recent study [60], it also shows that in-context learning implicitly performs meta-optimization through the attention mechanism.

the standard prompting becomes more evident when the model size exceeds 100B. Furthermore, the performance improvement with CoT prompting seems to be also varied for different tasks, *e.g.*, GSM8K > MAWPS > SWAMP for PaLM [33].

Key Techniques for LLMs. It has been a long way that LLMs evolve into the current state: *general* and *capable* learners. In the development process, a number of important techniques are proposed, which largely improve the capacity of LLMs. Here, we briefly list several important techniques that (potentially) lead to the success of LLMs, as follows.

• *Scaling*. As discussed in previous parts, there exists an evident scaling effect in Transformer language models: larger model/data sizes and more training compute typically lead to an improved model capacity [30, 34]. As two representative models, GPT-3 and PaLM explored the scaling limits by increasing the model size to 175B and 540B, respectively. Since compute budget is usually limited, scaling laws can be further employed to conduct a more compute-efficient allocation of the compute resources. For example, Chinchilla (with more training tokens) outperforms its counterpart model Gopher (with a larger model size) by increasing the data scale with the same compute budget [34]. In addition, data scaling should be with careful cleaning process, since the quality of pre-training data plays a key role in the model capacity.

• *Training*. Due to the huge model size, it is very challenging to successfully train a capable LLM. Distributed training algorithms are needed to learn the network parameters of LLMs, in which various parallel strategies are often jointly utilized. To support distributed training, several optimization frameworks have been released to facilitate the implementation and deployment of parallel algorithms, such as DeepSpeed [65] and Megatron-LM [66–68]. Also, optimization tricks are also important for training stability and model performance, *e.g.*, restart to overcome training loss spike [56] and mixed precision training [69]. More recently, GPT-4 [46] proposes to develop special infrastructure and optimization methods that reliably predict the performance of large models with much smaller models.

• *Ability eliciting*. After being pre-trained on large-scale corpora, LLMs are endowed with potential abilities as general-purpose task solvers. These abilities might not be explicitly exhibited when LLMs perform some specific tasks. As the technical approach, it is useful to design suitable task instructions or specific in-context learning strategies to elicit such abilities. For instance, chain-of-thought prompting has been shown to be useful to solve complex reasoning tasks by including intermediate reasoning steps. Furthermore, we can perform instruction tuning on LLMs with task descriptions expressed in natural language, for improving the generalizability of LLMs on unseen tasks. These eliciting techniques mainly correspond to the emergent abilities of LLMs, which may not show the same effect on small language models.

• *Alignment tuning*. Since LLMs are trained to capture the data characteristics of pre-training corpora (including both high-quality and low-quality data), they are likely to generate toxic, biased, or even harmful content for humans. It is necessary to align LLMs with human values, *e.g.*, *helpful*, *honest*, and *harmless*. For this purpose, InstructGPT [61] designs an effective tuning approach that enables LLMs to follow the expected instructions, which utilizes the technique of *reinforcement learning with human feedback* [61, 70]. It incorporates human in the training loop with elaborately designed labeling strategies. ChatGPT is indeed developed on a similar technique to InstructGPT, which shows a strong alignment capacity in producing high-quality, harmless responses, *e.g.*, rejecting to answer insulting questions.

• *Tools manipulation*. In essence, LLMs are trained as text generators over massive plain text corpora, thus performing less well on the tasks that are not best expressed in the form of text (*e.g.*, numerical computation). In addition, their capacities are also limited to the pre-training data, *e.g.*, the inability to capture up-to-date information. To tackle these issues, a recently proposed technique is to employ external tools to compensate for the deficiencies of LLMs [71, 72]. For example, LLMs can utilize the calculator for accurate computation [71] and employ search engines to retrieve unknown information [72]. More recently, ChatGPT has enabled the mechanism of using external plugins (existing or newly created apps)[9], which are by analogy with the "*eyes and ears*" of LLMs. Such a mechanism can broadly expand the scope of capacities for LLMs.

In addition, many other factors (*e.g.*, the upgrade of hardware) also contribute to the success of LLMs. Currently, we limit our discussion to the major technical approaches and key findings for developing LLMs.

2.2 Technical Evolution of GPT-series Models

Due to the excellent capacity in communicating with humans, ChatGPT has ignited the excitement of the AI community since its release. ChatGPT is developed based on the powerful GPT model with specially optimized conversation capacities. Considering the ever-growing interest in ChatGPT and GPT models, we add a special discussion about the technical evolution of the GPT-series models, to briefly summarize the progress how they have been developed in the past years. Meanwhile, we drew a schematic diagram depicting the technological evolution of the GPT-series models in Figure 3. The basic principle underlying GPT models is to compress the world knowledge into the decoder-only Transformer model by language modeling, such that it can recover (or memorize) the semantics of world knowledge and serve as a general-purpose task solver. Two key points to the success are (I) training decoder-only Transformer language models that can *accurately predict the next word* and (II) *scaling up the size of language models*. Overall, the research of OpenAI on LLMs can be roughly divided into the following stages[10].

Early Explorations. According to one interview with Ilya Sutskever[11] (a co-founder and chief scientist of OpenAI),

9. https://openai.com/blog/chatgpt-plugins

10. Note that the discussion of this part can be somewhat subjective. The overall viewpoints and summaries are made based on the understanding of the survey authors by reading the papers, blog articles, interview reports and APIs released by OpenAI.

11. https://hackernoon.com/an-interview-with-ilya-sutskever-co-founder-of-openai

TABLE 1: Statistics of large language models (having a size larger than 10B in this survey) in recent years, including the capacity evaluation, pre-training data scale (either in the number of tokens or storage size) and hardware resource costs. In this table, we only include LLMs with a public paper about the technical details. Here, "Release Time" indicates the date when the corresponding paper was officially released. "Publicly Available" means that the model checkpoints can be publicly accessible while "Closed Source" means the opposite. "Adaptation" indicates whether the model has been with subsequent fine-tuning: IT denotes instruction tuning and RLHF denotes reinforcement learning with human feedback. "Evaluation" indicates whether the model has been evaluated with corresponding abilities in their original paper: ICL denotes in-context learning and CoT denotes chain-of-thought. "*" denotes the largest publicly available version.

	Model	Release Time	Size (B)	Base Model	Adaptation		Pre-train Data Scale	Latest Data Timestamp	Hardware (GPUs / TPUs)	Training Time	Evaluation	
					IT	RLHF					ICL	CoT
Publicly Available	T5 [73]	Oct-2019	11	-	-	-	1T tokens	Apr-2019	1024 TPU v3	-	✓	-
	mT5 [74]	Oct-2020	13	-	-	-	1T tokens	-	-	-	✓	-
	PanGu-α [75]	Apr-2021	13*	-	-	-	1.1TB	-	2048 Ascend 910	-	✓	-
	CPM-2 [76]	Jun-2021	198	-	-	-	2.6TB	-	-	-	-	-
	T0 [28]	Oct-2021	11	T5	✓	-	-	-	512 TPU v3	27 h	✓	-
	CodeGen [77]	Mar-2022	16	-	-	-	577B tokens	-	-	-	✓	-
	GPT-NeoX-20B [78]	Apr-2022	20	-	-	-	825GB	-	96 40G A100	-	✓	-
	Tk-Instruct [79]	Apr-2022	11	T5	✓	-	-	-	256 TPU v3	4 h	✓	-
	UL2 [80]	May-2022	20	-	-	-	1T tokens	Apr-2019	512 TPU v4	-	✓	✓
	OPT [81]	May-2022	175	-	-	-	180B tokens	-	992 80G A100	-	✓	-
	NLLB [82]	Jul-2022	54.5	-	-	-	-	-	-	-	✓	-
	CodeGeeX [83]	Sep-2022	13	-	-	-	850B tokens	-	1536 Ascend 910	60 d	✓	-
	GLM [84]	Oct-2022	130	-	-	-	400B tokens	-	768 40G A100	60 d	✓	-
	Flan-T5 [64]	Oct-2022	11	T5	✓	-	-	-	-	-	✓	✓
	BLOOM [69]	Nov-2022	176	-	-	-	366B tokens	-	384 80G A100	105 d	✓	-
	mT0 [85]	Nov-2022	13	mT5	✓	-	-	-	-	-	✓	-
	Galactica [35]	Nov-2022	120	-	-	-	106B tokens	-	-	-	✓	✓
	BLOOMZ [85]	Nov-2022	176	BLOOM	✓	-	-	-	-	-	✓	-
	OPT-IML [86]	Dec-2022	175	OPT	✓	-	-	-	128 40G A100	-	✓	✓
	LLaMA [57]	Feb-2023	65	-	-	-	1.4T tokens	-	2048 80G A100	21 d	✓	-
	Pythia [87]	Apr-2023	12	-	-	-	300B tokens	-	256 40G A100	-	✓	-
	CodeGen2 [88]	May-2023	16	-	-	-	400B tokens	-	-	-	✓	-
	StarCoder [89]	May-2023	15.5	-	-	-	1T tokens	-	512 40G A100	-	✓	✓
	LLaMA2 [90]	Jul-2023	70	-	✓	✓	2T tokens	-	2000 80G A100	-	✓	-
Closed Source	GPT-3 [55]	May-2020	175	-	-	-	300B tokens	-	-	-	✓	-
	GShard [91]	Jun-2020	600	-	-	-	1T tokens	-	2048 TPU v3	4 d	-	-
	Codex [92]	Jul-2021	12	GPT-3	-	-	100B tokens	May-2020	-	-	✓	-
	ERNIE 3.0 [93]	Jul-2021	10	-	-	-	375B tokens	-	384 V100	-	✓	-
	Jurassic-1 [94]	Aug-2021	178	-	-	-	300B tokens	-	800 GPU	-	✓	-
	HyperCLOVA [95]	Sep-2021	82	-	-	-	300B tokens	-	1024 A100	13.4 d	✓	-
	FLAN [62]	Sep-2021	137	LaMDA-PT	✓	-	-	-	128 TPU v3	60 h	✓	-
	Yuan 1.0 [96]	Oct-2021	245	-	-	-	180B tokens	-	2128 GPU	-	✓	-
	Anthropic [97]	Dec-2021	52	-	-	-	400B tokens	-	-	-	✓	-
	WebGPT [72]	Dec-2021	175	GPT-3	-	✓	-	-	-	-	✓	-
	Gopher [59]	Dec-2021	280	-	-	-	300B tokens	-	4096 TPU v3	920 h	✓	-
	ERNIE 3.0 Titan [98]	Dec-2021	260	-	-	-	-	-	-	-	✓	-
	GLaM [99]	Dec-2021	1200	-	-	-	280B tokens	-	1024 TPU v4	574 h	✓	-
	LaMDA [63]	Jan-2022	137	-	-	-	768B tokens	-	1024 TPU v3	57.7 d	-	-
	MT-NLG [100]	Jan-2022	530	-	-	-	270B tokens	-	4480 80G A100	-	✓	-
	AlphaCode [101]	Feb-2022	41	-	-	-	967B tokens	Jul-2021	-	-	-	-
	InstructGPT [61]	Mar-2022	175	GPT-3	✓	✓	-	-	-	-	✓	-
	Chinchilla [34]	Mar-2022	70	-	-	-	1.4T tokens	-	-	-	✓	-
	PaLM [56]	Apr-2022	540	-	-	-	780B tokens	-	6144 TPU v4	-	✓	✓
	AlexaTM [102]	Aug-2022	20	-	-	-	1.3T tokens	-	128 A100	120 d	✓	✓
	Sparrow [103]	Sep-2022	70	-	-	✓	-	-	64 TPU v3	-	✓	-
	WeLM [104]	Sep-2022	10	-	-	-	300B tokens	-	128 A100 40G	24 d	✓	-
	U-PaLM [105]	Oct-2022	540	PaLM	-	-	-	-	512 TPU v4	5 d	✓	✓
	Flan-PaLM [64]	Oct-2022	540	PaLM	✓	-	-	-	512 TPU v4	37 h	✓	✓
	Flan-U-PaLM [64]	Oct-2022	540	U-PaLM	✓	-	-	-	-	-	✓	✓
	GPT-4 [46]	Mar-2023	-	-	✓	✓	-	-	-	-	✓	✓
	PanGu-Σ [106]	Mar-2023	1085	PanGu-α	-	-	329B tokens	-	512 Ascend 910	100 d	✓	-
	PaLM2 [107]	May-2023	16	-	✓	-	100B tokens	-	-	-	✓	✓

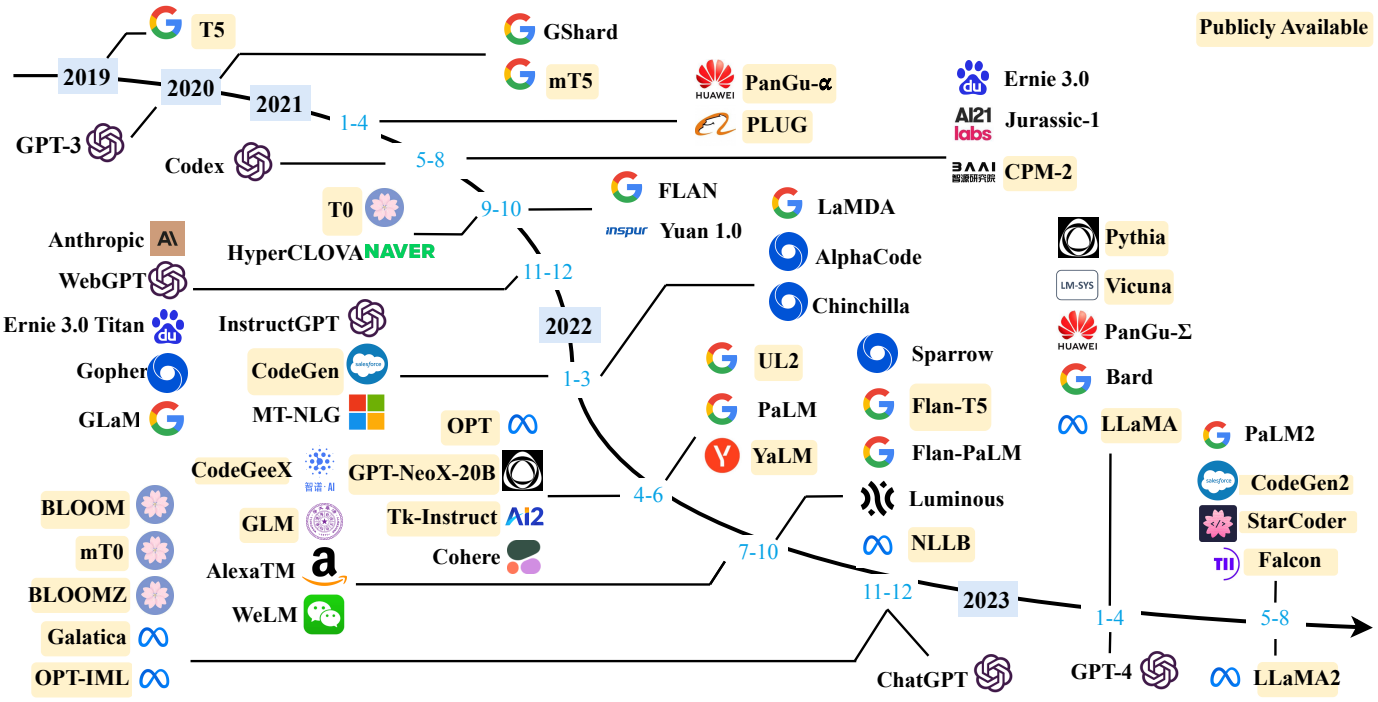


Fig. 2: A timeline of existing large language models (having a size larger than 10B) in recent years. The timeline was established mainly according to the release date (*e.g.*, the submission date to arXiv) of the technical paper for a model. If there was not a corresponding paper, we set the date of a model as the earliest time of its public release or announcement. We mark the LLMs with publicly available model checkpoints in yellow color. Due to the space limit of the figure, we only include the LLMs with publicly reported evaluation results.

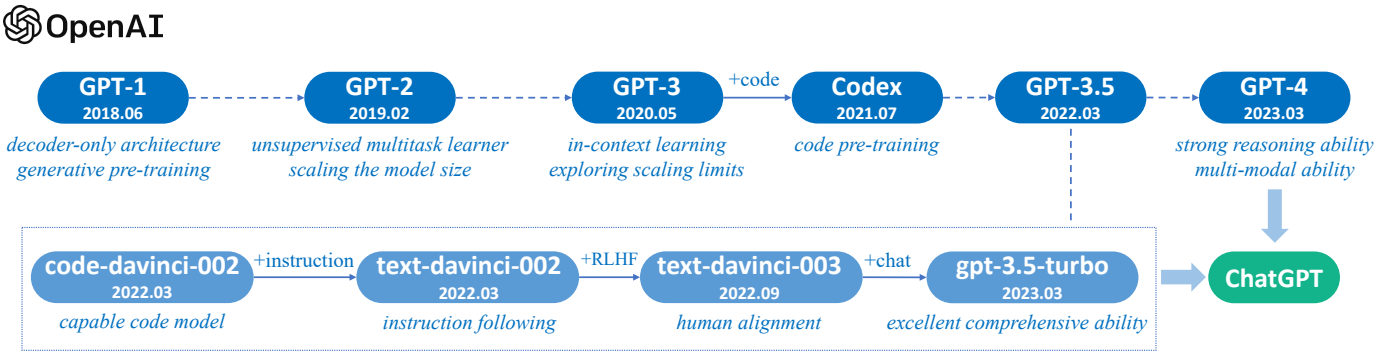


Fig. 3: A brief illustration for the technical evolution of GPT-series models. We plot this figure mainly based on the papers, blog articles and official APIs from OpenAI. Here, *solid lines* denote that there exists an explicit evidence (*e.g.*, the official statement that a new model is developed based on a base model) on the evolution path between two models, while *dashed lines* denote a relatively weaker evolution relation.

the idea of approaching intelligent systems with language models was already explored in the early days of OpenAI, while it was attempted with recurrent neural networks (RNN) [108]. With the advent of Transformer, OpenAI developed two initial GPT models, namely GPT-1 [109] and GPT-2 [26], which can be considered as the foundation to more powerful models subsequently *i.e.*, GPT-3 and GPT-4.

• *GPT-1*. In 2017, the Transformer model [22] was introduced by Google, and the OpenAI team quickly adapted their language modeling work to this new neural network architecture. They released the first GPT model in 2018, *i.e.*, GPT-1 [109], and coined the abbreviation term *GPT* as the model name, standing for *Generative Pre-Training*. GPT-1 was developed based on a generative, decoder-only Transformer architecture, and adopted a hybrid approach of unsupervised pretraining and supervised fine-tuning. GPT-1 has set up the core architecture for the GPT-series models and established the underlying principle to model natural language text, *i.e.*, predicting the next word.

• *GPT-2*. Following a similar architecture of GPT-1, GPT-2 [26] increased the parameter scale to 1.5B, which was trained with a large webpage dataset WebText. As claimed in the paper of GPT-2, it sought to perform tasks via unsupervised language modeling, without explicit

fine-tuning using labeled data. To motivate the approach, they introduced a probabilistic form for multi-task solving, *i.e.*, $p(output|input, task)$ (similar approaches have been adopted in [110]), which predicts the output conditioned on the input and task information. To model this conditional probability, language text can be naturally employed as a unified way to format input, output and task information. In this way, the process of solving a task can be cast as a word prediction problem for generating the solution text. Further, they introduced a more formal claim for this idea: "Since the (task-specific) supervised objective is the same as the unsupervised (language modeling) objective but only evaluated on a subset of the sequence, the global minimum of the unsupervised objective is also the global minimum of the supervised objective (for various tasks)" [26][12]. A basic understanding of this claim is that each (NLP) task can be considered as the word prediction problem based on a subset of the world text. Thus, unsupervised language modeling could be capable in solving various tasks, if it was trained to have sufficient capacity in recovering the world text. These early discussion in GPT-2's paper echoed in the interview of Ilya Sutskever by Jensen Huang: "What the neural network learns is some representation of the process that produced the text. This text is actually a projection of the world...the more accurate you are in predicting the next word, the higher the fidelity, the more resolution you get in this process..."[13].

Capacity Leap. Although GPT-2 is intended to be an "unsupervised multitask learner", it overall has an inferior performance compared with supervised fine-tuning state-of-the-art methods. Because it has a relatively small model size, it has been widely fine-tuned in downstream tasks, especially the dialog tasks [111, 112]. Based on GPT-2, GPT-3 demonstrates a key capacity leap by scaling of the (nearly same) generative pre-training architecture.

- *GPT-3*. GPT-3 [55] was released in 2020, which scaled the model parameters to an ever larger size of 175B. In the GPT-3's paper, it formally introduced the concept of in-context learning (ICL)[14], which utilizes LLMs in a few-shot or zero-shot way. ICL can teach (or instruct) LLMs to understand the tasks in the form of natural language text. With ICL, the pre-training and utilization of LLMs converge to the same language modeling paradigm: pre-training predicts the following text sequence conditioned on the context, while ICL predicts the correct task solution, which can be also formatted as a text sequence, given the task description and demonstrations. GPT-3 not only demonstrates very excellent performance in a variety of NLP tasks, but also on a number of specially designed tasks that require the abilities of reasoning or domain adaptation. Although the GPT-3's paper does not explicitly discuss the emergent abilities of LLMs, we can observe large performance leap that might transcend the basic scaling law [30], *e.g.*, larger models have significantly stronger ICL ability (illustrated in the original Figure 1.2 of the GPT-3's paper [55]). Overall, GPT-3 can be viewed as a remarkable landmark in the journey evolving from PLMs to LLMs. It has empirically proved that scaling the neural networks to a significant size can lead to a huge increase in model capacity.

Capacity Enhancement. Due to the strong capacities, GPT-3 has been the base model to develop even more capable LLMs for OpenAI. Overall, OpenAI has explored two major approaches to further improving the GPT-3 model, *i.e.*, training on code data and alignment with human preference, which are detailed as follows.

- *Training on code data*. A major limitation of the original GPT-3 model (pre-trained on plain text) lies in the lack of the reasoning ability on complex tasks, *e.g.*, completing the code and solving math problems. To enhance this ability, Codex [92] was introduced by OpenAI in July 2021, which was a GPT model fine-tuned on a large corpus of GitHub code. It demonstrated that Codex can solve very difficult programming problems, and also lead to a significant performance improvement in solving math problems [113]. Further, a contrastive approach [114] to training text and code embedding was reported in January 2022, which was shown to improve a series of related tasks (*i.e.*, linear-probe classification, text search and code search). Actually, the GPT-3.5 models are developed based on a code-based GPT model (*i.e.*, `code-davinci-002`), which indicates that training on code data is a very useful practice to improve the model capacity of GPT models, especially the reasoning ability. Furthermore, there is also a speculation that training on code data can greatly increase the chain-of-thought prompting abilities of LLMs [47], while it is still worth further investigation with more thorough verification.
- *Human alignment*. The related research of human alignment can be dated back to the year 2017 (or earlier) for OpenAI: a blog article entitled "learning from human preferences"[15] was posted on the OpenAI blog describing a work that applied reinforcement learning (RL) to learn from the *preference comparisons* annotated by humans [70] (similar to the *reward training* step in the aligning algorithm of InstructGPT in Figure 10). Shortly after the release of this RL paper [70], the paper of the Proximal Policy Optimization (PPO) [115] was published in July 2017, which now has been the foundational RL algorithm for learning from human preferences [61]. Later in January 2020, GPT-2 was fine-tuned using the aforementioned RL algorithms [70, 115], which leveraged human preferences to improve the capacities of GPT-2 on NLP tasks. In the same year, another work [116] trained a summarization model for optimizing human preferences in a similar way. Based on these prior work, InstructGPT [61] was proposed in January 2022 to improve the GPT-3 model for human alignment, which formally established a three-stage *reinforcement learning from human feedback (RLHF)* algorithm. Note that it seems that the wording of "*instruction tuning*" has seldom been used in OpenAI's paper and documentation, which is substituted by *supervised fine-tuning on human demonstrations* (*i.e.*, the first step of the RLHF algorithm [61]). In addition to improving the instruction following capacity, the RLHF algorithm is particularly useful to mitigate the issues of generating harm

12. To better understand this sentence, we put some explanation words in parentheses.

13. https://lifearchitect.ai/ilya/

14. GPT-2 essentially used ICL for unsupervised task learning, though it wasn't called ICL at that time.

15. https://openai.com/research/learning-from-human-preferences

or toxic content for LLMs, which is key to the safe deployment of LLMs in practice. OpenAI describes their approach to alignment research in a technical article [117], which has summarized three promising directions: "training AI systems to use human feedback, to assist human evaluation and to do alignment research".

These enhancement techniques lead to the improved GPT-3 models with stronger capacities, which are called GPT-3.5 models by OpenAI (see the discussion about the OpenAI API in Section 3.1).

The Milestones of Language Models. Based on all the exploration efforts, two major milestones have been achieved by OpenAI, namely ChatGPT [118] and GPT-4 [46], which have largely raised the capacity bar of existing AI systems.

• *ChatGPT*. In November 2022, OpenAI released the conversation model ChatGPT, based on the GPT models (GPT-3.5 and GPT-4). As the official blog article introduced [118], ChatGPT was trained in a similar way as InstructGPT (called "a sibling model to InstructGPT" in the original post), while specially optimized for dialogue. They reported a difference between the training of ChatGPT and InstructGPT in the data collection setup: human-generated conversations (playing both the roles of user and AI) are combined with the InstructGPT dataset in a dialogue format for training ChatGPT. ChatGPT exhibited superior capacities in communicating with humans: possessing a vast store of knowledge, skill at reasoning on mathematical problems, tracing the context accurately in multi-turn dialogues, and aligning well with human values for safe use. Later on, the plugin mechanism has been supported in ChatGPT, which further extends the capacities of ChatGPT with existing tools or apps. So far, it seems to be the ever most powerful chatbot in the AI history. The launch of ChatGPT has a significant impact on the AI research in the future, which sheds light on the exploration of human-like AI systems.

• *GPT-4*. As another remarkable progress, GPT-4 [46] was released in March 2023, which extended the text input to multimodal signals. Overall, GPT-4 has stronger capacities in solving complex tasks than GPT-3.5, showing a large performance improvement on many evaluation tasks. A recent study [41] investigated the capacities of GPT-4 by conducting qualitative tests with human-generated problems, spanning a diverse range of difficult tasks, and showed that GPT-4 can achieve more superior performance than prior GPT models such as ChatGPT. Furthermore, GPT-4 responds more safely to malicious or provocative queries, due to a six-month iterative alignment (with an additional safety reward signal in the RLHF training). In the technical report, OpenAI has emphasized how to safely develop GPT-4 and applied a number of intervention strategies to mitigate the possible issues of LLMs, such as hallucinations, privacy and overreliance. For example, they introduced the mechanism called *red teaming* [119] to reduce the harm or toxic content generation. As another important aspect, GPT-4 has been developed on a well-established deep learning infrastructure with improved optimization methods. They introduced a new mechanism called *predictable scaling* that can accurately predict the final performance with a small proportion of compute during model training.

Despite the huge progress, there are still limitations with these superior LLMs, *e.g.*, generating hallucinations with factual errors or potentially risky response within some specific context [46]. More limitations or issues of LLMs will be discussed in Section 7. It poses long-standing research challenges to develop more capable, safer LLMs. From the perspective of engineering, OpenAI has adopted an iterative deployment strategy [120] to develop the models and products by following a five-stage development and deployment life-cycle, which aims to effectively reduce the potential risks of using the models. In the following, we will dive into the technical details in order to have a specific understanding of how they have been developed.

3 Resources of LLMs

It is by no means an easy job to develop or reproduce LLMs, considering the challenging technical issues and huge demands of computation resources. A feasible way is to learn experiences from existing LLMs and reuse publicly available resources for incremental development or experimental study. In this section, we briefly summarize the publicly available resources for developing LLMs, including model checkpoints (or APIs), corpora and libraries.

3.1 Publicly Available Model Checkpoints or APIs

Given the huge cost of model pre-training, well-trained model checkpoints are critical to the study and development of LLMs for the research community. Since the parameter scale is a key factor to consider for using LLMs, we categorize these public models into two scale levels (*i.e., tens of billions of parameters* and *hundreds of billions of parameters*), which is useful for users to identify the suitable resources according to their resource budget. In addition, for inference, we can directly employ public APIs to perform our tasks, without running the model locally. Next, we introduce the publicly available model checkpoints and APIs.

Models with Tens of Billions of Parameters. Most of the models in this category have a parameter scale ranging from 10B to 20B, except LLaMA [57] and LLaMA2 [90] (containing 70B parameters in the largest version), NLLB [82] (containing 54.5B parameters in the largest version), and Falcon [121] (containing 40B parameters in the largest version). Other models within this range include mT5 [74], PanGu-α [75], T0 [28], GPT-NeoX-20B [78], CodeGen [77], UL2 [80], Flan-T5 [64], and mT0 [85]. Among them, Flan-T5 (11B version) can serve as a premier model for research on instruction tuning, since it explores the instruction tuning from three aspects [64]: increasing the number of tasks, scaling the model size, and fine-tuning with chain-of-thought prompting data. Besides, CodeGen (11B version), as an autoregressive language model designed for generating code, can be considered as a good candidate for exploring the code generation ability. It also introduces a new benchmark MTPB [77] specially for multi-turn program synthesis, which is composed by 115 expert-generated problems. To solve these problems, it requires LLMs to acquire sufficient programming knowledge (*e.g.*, math, array operations, and algorithms). More recently, CodeGen2 [88] has been released to explore the impact of choices in model architecture, learning algorithms, and data distributions on the model. As

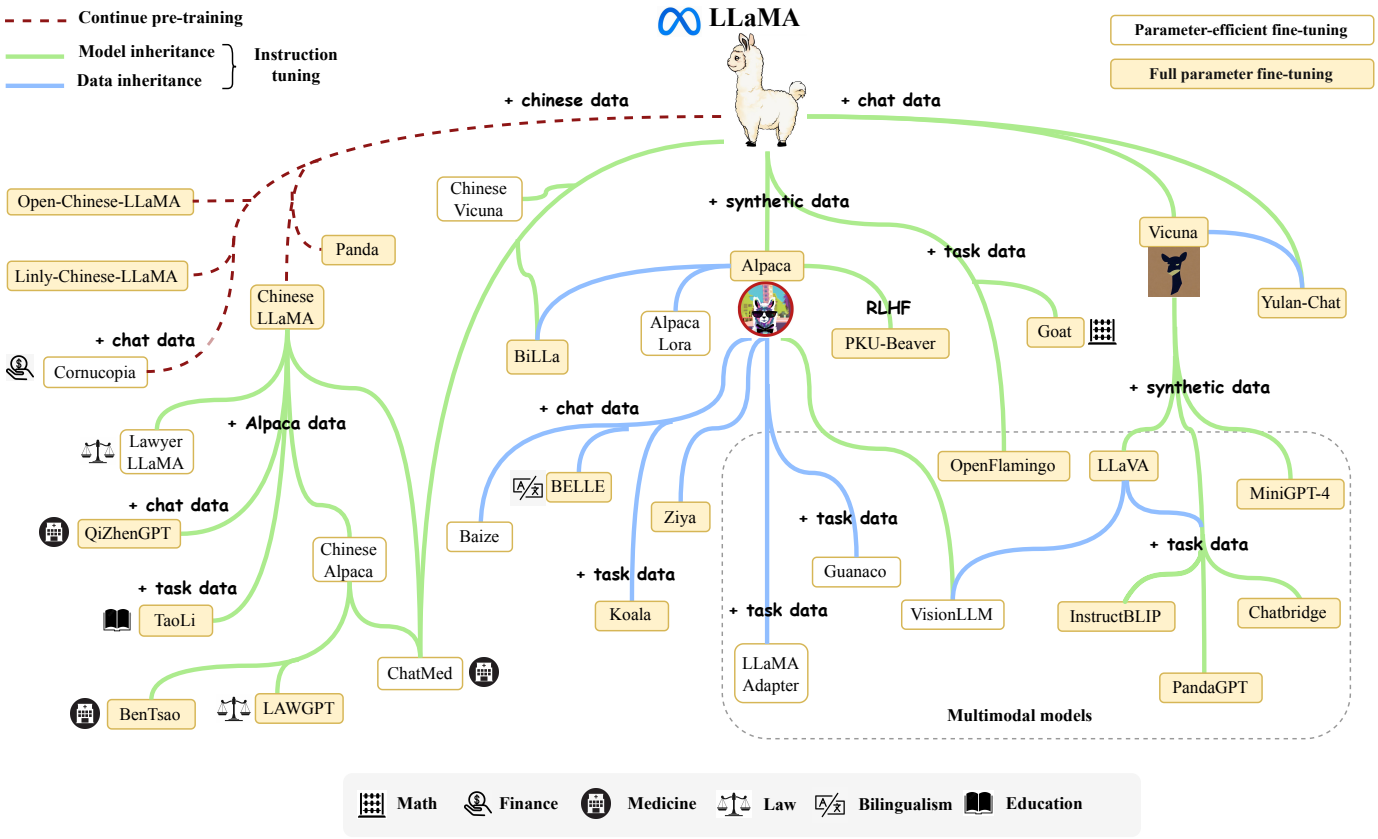


Fig. 4: An evolutionary graph of the research work conducted on LLaMA. Due to the huge number, we cannot include all the LLaMA variants in this figure, even much excellent work. To support incremental update, we share the source file of this figure, and welcome the readers to include the desired models by submitting the pull requests on our GitHub page.

another LLM specialized in coding abilities, StarCoder [89] has also achieved excellent results. As for multilingual tasks, mT0 (13B version) might be a good candidate model, which has been fine-tuned on multilingual tasks with multilingual prompts. Furthermore, PanGu-α [75] shows good performance in Chinese downstream tasks in zero-shot or few-shot settings, which is developed based on the deep learning framework MindSpore [122]. Note that PanGu-α [75] holds multiple versions of models (up to 200B parameters), while the largest public version has 13B parameters. As a popular LLM, LLaMA (65B version) [57], which contains approximately five times as many parameters as other models, has exhibited superior performance in tasks related to instruction following. Compared to LLaMA, LLaMA2 [90] has made more explorations in reinforcement learning from human feedback (RLHF) and developed a chat-oriented version called *LLaMA-chat*, which generally outperforms existing open-source models across a range of helpfulness and safety benchmarks. Due to the openness and effectiveness, LLaMA has attracted significant attention from the research community, and many efforts [123–126] have been devoted to fine-tuning or continually pre-training its different model versions for implementing new models or tools. More recently, Falcon [121], as another open-source LLM, has also achieved very excellent performance on open benchmarks. It is featured by a more careful data cleaning process to prepare the pre-training data (with a publicly shared dataset *RefinedWeb [127]*). Typically, pre-training models at this scale require hundreds or even thousands of GPUs or TPUs. For instance, GPT-NeoX-20B uses 12 supermicro servers, each equipped with 8 NVIDIA A100-SXM4-40GB GPUs, while LLaMA utilizes 2,048 A100-80G GPUs as reported in their original publications. To accurately estimate the computation resources needed, it is suggested to use the metrics measuring the number of involved computations such as *FLOPS* (*i.e.*, FLoating point number Operations Per Second) [30].

Models with Hundreds of Billions of Parameters. For models in this category, only a handful of models have been publicly released. For example, OPT [81], OPT-IML [86], BLOOM [69], and BLOOMZ [85] have nearly the same number of parameters as GPT-3 (175B version), while GLM [84] and Galactica [35] have 130B and 120B parameters, respectively. Among them, OPT (175B version), with the instruction-tuned version OPT-IML, has been specially motivated for open sharing, which aims to enable researchers to carry out reproducible research at scale. For research in cross-lingual generalization, BLOOM (176B version) and BLOOMZ (176B version) can be used as base models, due to the competence in multilingual language modeling tasks. As a bilingual LLM, GLM has also provided a popular small-sized Chinese chat model ChatGLM2-6B (a updated version for ChatGLM-6B), which is featured with many improvements in efficiency and capacity (*e.g.*, quantization,

32K-length context, fast inference rate). Models of this scale typically require thousands of GPUs or TPUs to train. For instance, OPT (175B version) used 992 A100-80GB GPUs, while GLM (130B version) used a cluster of 96 NVIDIA DGX-A100 (8x40G) GPU nodes.

LLaMA Model Family. The collection of LLaMA models [57] were introduced by Meta AI in February, 2023, consisting of four sizes (7B, 13B, 30B and 65B). Since released, LLaMA has attracted extensive attention from both research and industry communities. LLaMA models have achieved very excellent performance on various open benchmarks, which have become the most popular open language models thus far. A large number of researchers have extended LLaMA models by either instruction tuning or continual pretraining. In particular, instruction tuning LLaMA has become a major approach to developing customized or specialized models, due to the relatively low computational costs. To effectively adapt LLaMA models in non-English languages, it often needs to extend the original vocabulary (trained mainly on English corpus) or fine-tune it with instructions or data in the target language. Among these extended models, Stanford Alpaca [128] is the first open instruct-following model fine-tuned based on LLaMA (7B). It is trained by 52K instruction-following demonstrations generated via self-instruct [129] using `text-davinci-003`. The instruction data, named *Alpaca-52K*, and training code have been extensively adopted in subsequent work, such as Alpaca-LoRA [130] (a reproduction of Stanford Alpaca using LoRA [131]), Koala [132], and BELLE [133]. In addition, Vicuna [124] is another popular LLaMA variant, trained upon user-shared conversations collected from ShareGPT [16]. Due to the excellent performance and availability of the LLaMA model family, many multimodal models incorporate them as the base language models, to achieve strong language understanding and generation abilities. Compared with other variants, Vicuna is more preferred in multimodal language models, which have led to the emergence of a variety of popular models, including LLaVA [134], MiniGPT-4 [135], InstructBLIP [136], and PandaGPT [137]. The release of LLaMA has greatly advanced the research progress of LLMs. To summarize the research work conducted on LLaMA, we present a brief evolutionary graph in Figure 4.

Public API of LLMs. Instead of directly using the model copies, APIs provide a more convenient way for common users to use LLMs, without the need of running the model locally. As a representative interface for using LLMs, the APIs for the GPT-series models [46, 55, 61, 92] have been widely used for both academia and industry[17]. OpenAI has provided seven major interfaces to the models in GPT-3 series: `ada`, `babbage`, `curie`, `davinci` (the most powerful version in GPT-3 series), `text-ada-001`, `text-babbage-001`, and `text-curie-001`. Among them, the first four interfaces can be further fine-tuned on the host server of OpenAI. In particular, `babbage`, `curie`, and `davinci` correspond to the GPT-3 (1B), GPT-3 (6.7B), and GPT-3 (175B) models, respectively [55]. In addition, there are also two APIs related to Codex [92], called `code-cushman-001` (a powerful and multilingual version of the Codex (12B) [92]) and `code-davinci-002`. Further, GPT-3.5 series include one base model `code-davinci-002` and three enhanced versions, namely `text-davinci-002`, `text-davinci-003`, and `gpt-3.5-turbo-0301`. It is worth noting that `gpt-3.5-turbo-0301` is the interface to invoke ChatGPT. More recently, OpenAI has also released the corresponding APIs for GPT-4, including `gpt-4`, `gpt-4-0314`, `gpt-4-32k`, and `gpt-4-32k-0314`. Overall, the choice of API interfaces depends on the specific application scenarios and response requirements. The detailed usage can be found on their project websites[18].

TABLE 2: Statistics of commonly-used data sources.

Corpora	Size	Source	Latest Update Time
BookCorpus [138]	5GB	Books	Dec-2015
Gutenberg [139]	-	Books	Dec-2021
C4 [73]	800GB	CommonCrawl	Apr-2019
CC-Stories-R [140]	31GB	CommonCrawl	Sep-2019
CC-NEWS [27]	78GB	CommonCrawl	Feb-2019
REALNEWs [141]	120GB	CommonCrawl	Apr-2019
OpenWebText [142]	38GB	Reddit links	Mar-2023
Pushift.io [143]	2TB	Reddit links	Mar-2023
Wikipedia [144]	21GB	Wikipedia	Mar-2023
BigQuery [145]	-	Codes	Mar-2023
the Pile [146]	800GB	Other	Dec-2020
ROOTS [147]	1.6TB	Other	Jun-2022

3.2 Commonly Used Corpora

In contrast to earlier PLMs, LLMs which consist of a significantly larger number of parameters require a higher volume of training data that covers a broad range of content. For this need, there are increasingly more accessible training datasets that have been released for research. In this section, we will briefly summarize several widely used corpora for training LLMs. Based on their content types, we categorize these corpora into six groups: Books, CommonCrawl, Reddit links, Wikipedia, Code, and others.

Books. BookCorpus [138] is a commonly used dataset in previous small-scale models (*e.g.*, GPT [109] and GPT-2 [26]), consisting of over 11,000 books covering a wide range of topics and genres (*e.g.*, novels and biographies). Another large-scale book corpus is Project Gutenberg [139], consisting of over 70,000 literary books including novels, essays, poetry, drama, history, science, philosophy, and other types of works in the public domain. It is currently one of the largest open-source book collections, which is used in training of MT-NLG [100] and LLaMA [57]. As for Books1 [55] and Books2 [55] used in GPT-3 [55], they are much larger than BookCorpus but have not been publicly released so far.

CommonCrawl. CommonCrawl [148] is one of the largest open-source web crawling databases, containing a petabyte-scale data volume, which has been widely used as training data for existing LLMs. As the whole dataset is very large, existing studies mainly extract subsets of web pages from

16. https://sharegpt.com/
17. https://platform.openai.com/docs/api-reference/introduction
18. https://platform.openai.com/docs/models/overview

it within a specific period. However, due to the widespread existence of noisy and low-quality information in web data, it is necessary to perform data preprocessing before usage. Based on CommonCrawl, there are four filtered datasets that are commonly used in existing work: C4 [73], CC-Stories [140], CC-News [27], and RealNews [141]. The Colossal Clean Crawled Corpus (C4) includes five variants[19], namely en (806G), en.noclean (6T), realnewslike (36G), webtextlike (17G), and multilingual (38T). The *en* version has been utilized for pre-training T5 [73], LaMDA [63], Gopher [59], and UL2 [80]. The multilingual C4, also called mC4, has been used in mT5 [74]. CC-Stories (31G) is composed of a subset of CommonCrawl data, in which the contents are made in a story-like way. Because the original source of CC-Stories is not available now, we include a reproduction version, *CC-Stories-R* [149], in Table 2. Moreover, two news corpora extracted from CommonCrawl, *i.e.,* REALNEWS (120G) and CC-News (76G), are also commonly used as the pre-training data.

Reddit Links. Reddit is a social media platform that enables users to submit links and text posts, which can be voted on by others through "upvotes" or "downvotes". Highly upvoted posts are often considered useful, and can be utilized to create high-quality datasets. WebText [26] is a well-known corpus composed of highly upvoted links from Reddit, but it is not publicly available. As a surrogate, there is a readily accessible open-source alternative called OpenWebText [142]. Another corpus extracted from Reddit is PushShift.io [143], a real-time updated dataset that consists of historical data from Reddit since its creation day. Pushshift provides not only monthly data dumps but also useful utility tools to support users in searching, summarizing, and conducting preliminary investigations on the entire dataset. This makes it easy for users to collect and process Reddit data.

Wikipedia. Wikipedia [144] is an online encyclopedia containing a large volume of high-quality articles on diverse topics. Most of these articles are composed in an expository style of writing (with supporting references), covering a wide range of languages and fields. Typically, the English-only filtered versions of Wikipedia are widely used in most LLMs (*e.g.,* GPT-3 [55], LaMDA [63], and LLaMA [57]). Wikipedia is available in multiple languages, so it can be used in multilingual settings.

Code. To collect code data, existing work mainly crawls open-source licensed codes from the Internet. Two major sources are public code repositories under open-source licenses (*e.g.,* GitHub) and code-related question-answering platforms (*e.g.,* StackOverflow). Google has publicly released the BigQuery dataset [145], which includes a substantial number of open-source licensed code snippets in various programming languages, serving as a representative code dataset. CodeGen has utilized BIGQUERY [77], a subset of the BigQuery dataset, for training the multilingual version of CodeGen (CodeGen-Multi).

Others. The Pile [146] is a large-scale, diverse, and open-source text dataset consisting of over 800GB of data from multiple sources, including books, websites, codes, scientific papers, and social media platforms. It is constructed from 22 diverse high-quality subsets. The Pile dataset is widely used in models with different parameter scales, such as GPT-J (6B) [150], CodeGen (16B) [77], and Megatron-Turing NLG (530B) [100]. ROOTS [147] is composed of various smaller datasets (totally 1.61 TB of text) and covers 59 different languages (containing natural languages and programming languages), which have been used for training BLOOM [69].

In practice, it commonly requires a mixture of different data sources for pre-training LLMs (see Figure 5), instead of a single corpus. Therefore, existing studies commonly mix several ready-made datasets (*e.g.,* C4, OpenWebText, and the Pile), and then perform further processing to obtain the pre-training corpus. Furthermore, to train the LLMs that are adaptive to specific applications, it is also important to extract data from relevant sources (*e.g.,* Wikipedia and BigQuery) for enriching the corresponding information in pre-training data. To have a quick reference of the data sources used in existing LLMs, we present the pre-training corpora of three representative LLMs:

- *GPT-3* (175B) [55] was trained on a mixed dataset of 300B tokens, including CommonCrawl [148], WebText2 [55], Books1 [55], Books2 [55], and Wikipedia [144].
- *PaLM* (540B) [56] uses a pre-training dataset of 780B tokens, which is sourced from social media conversations, filtered webpages, books, Github, multilingual Wikipedia, and news.
- *LLaMA* [57] extracts training data from various sources, including CommonCrawl, C4 [73], Github, Wikipedia, books, ArXiv, and StackExchange. The training data size for LLaMA (6B) and LLaMA (13B) is 1.0T tokens, while 1.4T tokens are used for LLaMA (32B) and LLaMA (65B).

3.3 Library Resource

In this part, we briefly introduce a series of available libraries for developing LLMs.

- **Transformers** [151] is an open-source Python library for building models using the Transformer architecture, which is developed and maintained by Hugging Face. It has a simple and user-friendly API, making it easy to use and customize various pre-trained models. It is a powerful library with a large and active community of users and developers who regularly update and improve the models and algorithms.
- **DeepSpeed** [65] is a deep learning optimization library (compatible with PyTorch) developed by Microsoft, which has been used to train a number of LLMs, such as MT-NLG [100] and BLOOM [69]. It provides the support of various optimization techniques for distributed training, such as memory optimization (ZeRO technique, gradient checkpointing), and pipeline parallelism.
- **Megatron-LM** [66–68] is a deep learning library developed by NVIDIA for training large-scale language models. It also provides rich optimization techniques for distributed training, including model and data parallelism, mixed-precision training, and FlashAttention. These optimization techniques can largely improve the training efficiency and speed, enabling efficient distributed training across GPUs.

19. https://www.tensorflow.org/datasets/catalog/c4

• **JAX** [152] is a Python library for high-performance machine learning algorithms developed by Google, allowing users to easily perform computations on arrays with hardware acceleration (*e.g.,* GPU or TPU). It enables efficient computation on various devices and also supports several featured functions, such as automatic differentiation and just-in-time compilation.

• **Colossal-AI** [153] is a deep learning library developed by HPC-AI Tech for training large-scale AI models. It is implemented based on PyTorch and supports a rich collection of parallel training strategies. Furthermore, it can also optimize heterogeneous memory management with methods proposed by PatrickStar [154]. Recently, a ChatGPT-like model called ColossalChat [126] has been publicly released with two versions (7B and 13B), which are developed using Colossal-AI based on LLaMA [57].

• **BMTrain** [155] is an efficient library developed by OpenBMB for training models with large-scale parameters in a distributed manner, which emphasizes code simplicity, low resource, and high availability. BMTrain has already incorporated several common LLMs (*e.g.,* Flan-T5 [64] and GLM [84]) into its ModelCenter, where developers can use these models directly.

• **FastMoE** [156] is a specialized training library for MoE (*i.e.,* mixture-of-experts) models. It is developed based on PyTorch, prioritizing both efficiency and user-friendliness in its design. FastMoE simplifies the process of transferring Transformer models to MoE models and supports both data parallelism and model parallelism during training.

In addition to the above library resources, existing deep learning frameworks (*e.g.,* PyTorch [157], TensorFlow [158], MXNet [159], PaddlePaddle [160], MindSpore [122] and OneFlow [161]) have also provided the support for parallel algorithms, which are commonly used for training large-scale models.

4 Pre-training

Pre-training establishes the basis of the abilities of LLMs. By pre-training on large-scale corpora, LLMs can acquire essential language understanding and generation skills [55, 56]. In this process, the scale and quality of the pre-training corpus are critical for LLMs to attain powerful capabilities. Furthermore, to effectively pre-train LLMs, model architectures, acceleration methods, and optimization techniques need to be well designed. In what follows, we first discuss the data collection and processing in Section 4.1, then introduce the commonly used model architectures in Section 4.2, and finally present the training techniques to stably and efficiently optimize LLMs in Section 4.3.

4.1 Data Collection

Compared with small-scale language models, LLMs have a stronger demand for high-quality data for model pre-training, and their model capacities largely rely on the pre-training corpus and how it has been preprocessed. In this part, we discuss the collection and processing of pre-training data, including data sources, preprocessing methods, and important analysis of how pre-training data affects the performance of LLMs.

4.1.1 Data Source

To develop a capable LLM, it is key to collect a large amount of natural language corpus from various data sources. Existing LLMs mainly leverage a mixture of diverse public textual datasets as the pre-training corpus. Figure 5 shows the distribution of the sources of pre-training data for a number of representative LLMs.

The source of pre-training corpus can be broadly categorized into two types: general data and specialized data. General data, such as webpages, books, and conversational text, is utilized by most LLMs [55, 56, 81] due to its large, diverse, and accessible nature, which can enhance the language modeling and generalization abilities of LLMs. In light of the impressive generalization capabilities exhibited by LLMs, there are also studies that extend their pre-training corpus to more specialized datasets, such as multilingual data, scientific data, and code, endowing LLMs with specific task-solving capabilities [35, 56, 77]. In what follows, we describe these two types of pre-training data sources and their effects on LLMs. For a detailed introduction to the commonly used corpus, one can refer to Section 3.2.

General Text Data. As we can see in Figure 5, the vast majority of LLMs adopt general-purpose pre-training data, such as webpages, books, and conversational text, which provides rich text sources on a variety of topics. Next, we briefly summarize three important kinds of general data.

• *Webpages*. Owing to the proliferation of the Internet, various types of data have been created, which enables LLMs to gain diverse linguistic knowledge and enhance their generalization capabilities [26, 73]. For convenient use of these data resources, a large amount of data is crawled from the web in previous work, such as CommonCrawl [148]. However, the crawled web data tends to contain both high-quality text, such as Wikipedia and low-quality text, like spam mail, thus it is important to filter and process webpages for improving the data quality.

• *Conversation text*. Conversation data can enhance the conversational competence of LLMs [81] and potentially improve their performance on a range of question-answering tasks [56]. Researchers can utilize subsets of public conversation corpus (*e.g.,* PushShift.io Reddit corpus) [143, 162] or collect conversation data from online social media. Since online conversational data often involves discussions among multiple participants, an effective processing way is to transform a conversation into a tree structure, where the utterance is linked to the one it responds to. In this way, the multi-party conversation tree can be divided into multiple sub-conversations, which can be collected in the pre-training corpus. Furthermore, a potential risk is that the excessive integration of dialogue data into LLMs may result in a side effect [81]: declarative instructions and direct interrogatives are erroneously perceived as the beginning of conversations, thus leading to a decline in the efficacy of the instructions.

• *Books*. Compared to other corpus, books provide an important source of formal long texts, which are potentially beneficial for LLMs to learn linguistic knowledge, model long-term dependency, and generate narrative and coherent texts. To obtain open-source book data, existing studies usually adopt the Books3 and Bookcorpus2 datasets, which are available in the Pile dataset [146].

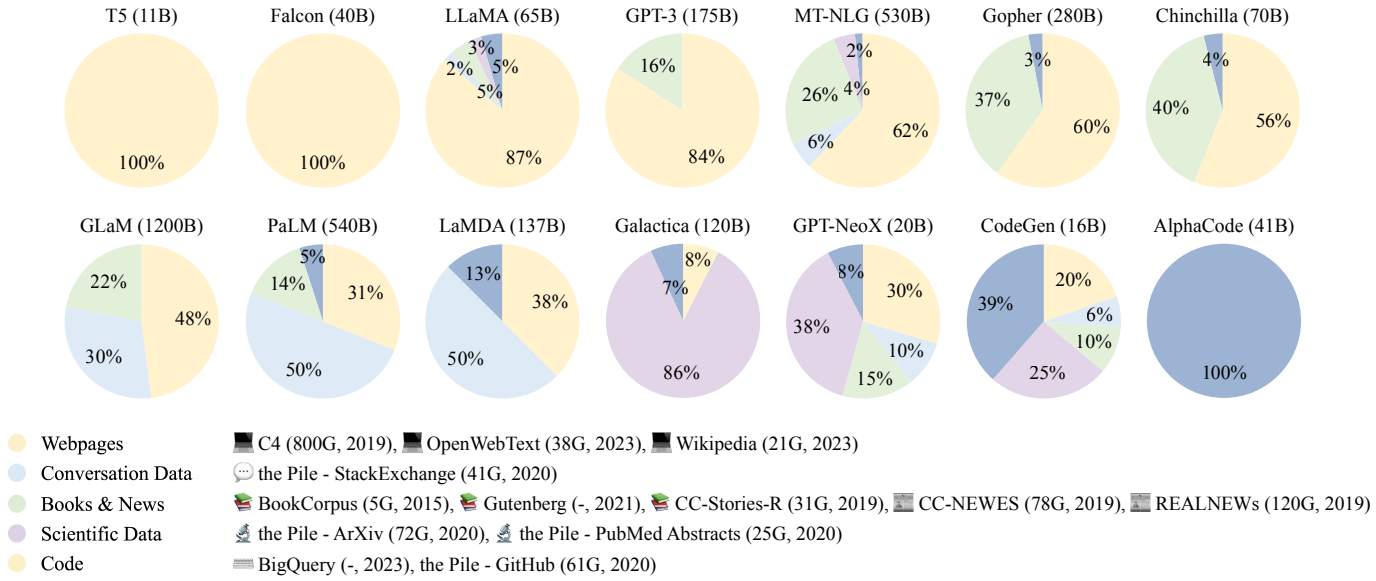


Fig. 5: Ratios of various data sources in the pre-training data for existing LLMs.

Specialized Text Data. Specialized datasets are useful to improve the specific capabilities of LLMs on downstream tasks. Next, we introduce three kinds of specialized data.

• *Multilingual text*. In addition to the text in the target language, integrating a multilingual corpus can enhance the multilingual abilities of language understanding and generation. For example, BLOOM [69] and PaLM [56] have curated multilingual data covering 46 and 122 languages, respectively, within their pre-training corpora. These models demonstrate impressive performance in multilingual tasks, such as translation, multilingual summarization, and multilingual question answering, and achieve comparable or superior performance to the state-of-the-art models that are fine-tuned on the corpus in the target language(s).

• *Scientific text*. The exploration of science by humans has been witnessed by the increasing growth of scientific publications. In order to enhance the understanding of scientific knowledge for LLMs [35, 163], it is useful to incorporate a scientific corpus for model pre-training [35, 163]. By pre-training on a vast amount of scientific text, LLMs can achieve impressive performance in scientific and reasoning tasks [164]. To construct the scientific corpus, existing efforts mainly collect arXiv papers, scientific textbooks, math webpages, and other related scientific resources. Due to the complex nature of data in scientific fields, such as mathematical symbols and protein sequences, specific tokenization and preprocessing techniques are usually required to transform these different formats of data into a unified form that can be processed by language models.

• *Code*. Program synthesis has been widely studied in the research community [92, 165–168], especially the use of PLMs trained on code [150, 169]. However, it remains challenging for these PLMs (*e.g.*, GPT-J [150]) to generate high-quality and accurate programs. Recent studies [92, 168] have found that training LLMs on a vast code corpus can lead to a substantial improvement in the quality of the synthesized programs. The generated programs can successfully pass expert-designed unit-test cases [92] or solve competitive programming questions [101]. In general, two types of code corpora are commonly used for pre-training LLMs. The first source is from programming question answering communities like Stack Exchange [170]. The second source is from public software repositories such as GitHub [77, 92, 168], where code data (including comments and docstrings) are collected for utilization. Compared to natural language text, code is in the format of a programming language, corresponding to long-range dependencies and accurate execution logic [171]. A recent study [47] also speculates that training on code might be a source of complex reasoning abilities (*e.g.*, chain-of-thought ability [33]). Furthermore, it has been shown that formatting reasoning tasks into code can help LLMs generate more accurate results [171].

4.1.2 Data Preprocessing

After collecting a large amount of text data, it is essential to preprocess the data for constructing the pre-training corpus, especially removing noisy, redundant, irrelevant, and potentially toxic data [56, 59, 172], which may largely affect the capacity and performance of LLMs. To facilitate the data processing, a recent study [173] proposes a useful data processing system for LLMs, named Data-Juicer, which provides over 50 processing operators and tools. In this part, we review the detailed data preprocessing strategies to improve the quality of the collected data [59, 69, 99]. A typical pipeline of preprocessing the pre-training data for LLMs has been illustrated in Figure 6.

Quality Filtering. To remove low-quality data from the collected corpus, existing work generally adopts two approaches: (1) classifier-based, and (2) heuristic-based. The former approach trains a selection classifier based on high-quality texts and leverages it to identify and filter out low-quality data. Typically, these methods [55, 56, 99] train a binary classifier with well-curated data (*e.g.*, Wikipedia pages) as positive instances and sample candidate data as negative instances, and predict the score that measures the quality of each data example. However, several stud-

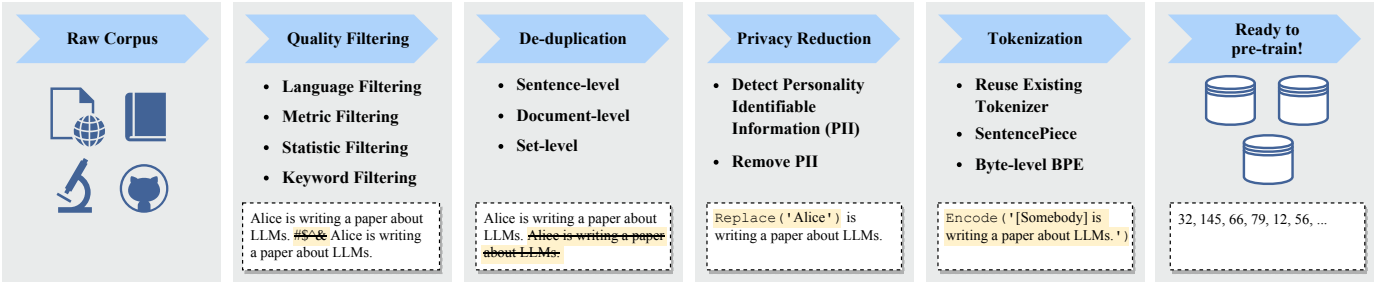


Fig. 6: An illustration of a typical data preprocessing pipeline for pre-training large language models.

ies [59, 99] find that a classifier-based approach may result in the unintentional removal of high-quality texts in dialectal, colloquial, and sociolectal languages, which potentially leads to bias in the pre-training corpus and diminishes the corpus diversity. As the second approach, several studies, such as BLOOM [69] and Gopher [59], employ heuristic-based approaches to eliminate low-quality texts through a set of well-designed rules, which can be summarized as follows:

- *Language based filtering*. If a LLM would be mainly used in the tasks of certain languages, the text in other languages can be filtered.
- *Metric based filtering*. Evaluation metrics about the generated texts, *e.g.*, perplexity, can be employed to detect and remove unnatural sentences.
- *Statistic based filtering*. Statistical features of a corpus, *e.g.*, the punctuation distribution, symbol-to-word ratio, and sentence length, can be utilized to measure the text quality and filter the low-quality data.
- *Keyword based filtering*. Based on specific keyword set, the noisy or unuseful elements in the text, such as HTML tags, hyperlinks, boilerplates, and offensive words, can be identified and removed.

De-duplication. Existing work [174] has found that duplicate data in a corpus would reduce the diversity of language models, which may cause the training process to become unstable and thus affect the model performance. Therefore, it is necessary to de-duplicate the pre-training corpus. Specially, de-duplication can be performed at different granularities, including sentence-level, document-level, and dataset-level de-duplication. First, low-quality sentences that contain repeated words and phrases should be removed, as they may introduce repetitive patterns in language modeling [175]. At the document level, existing studies mostly rely on the overlap ratio of surface features (*e.g.*, words and n-grams overlap) between documents to detect and remove duplicate documents containing similar contents [57, 59, 69, 176]. Furthermore, to avoid the dataset contamination problem, it is also crucial to prevent the overlap between the training and evaluation sets [56], by removing the possible duplicate texts from the training set. It has been shown that the three levels of de-duplication are useful to improve the training of LLMs [56, 177], which should be jointly used in practice.

Privacy Reduction. The majority of pre-training text data is obtained from web sources, including user-generated content involving sensitive or personal information, which may increase the risk of privacy breaches [178]. Thus, it is necessary to remove the *personally identifiable information (PII)* from the pre-training corpus. One direct and effective approach is to employ rule-based methods, such as keyword spotting, to detect and remove PII such as names, addresses, and phone numbers [147]. Furthermore, researchers also find that the vulnerability of LLMs under privacy attacks can be attributed to the presence of duplicate PII data in the pre-training corpus [179]. Therefore, de-duplication can also reduce privacy risks to some extent.

Tokenization. Tokenization is also a crucial step for data preprocessing. It aims to segment raw text into sequences of individual tokens, which are subsequently used as the inputs of LLMs. In traditional NLP research (*e.g.*, sequence labeling with conditional random fields [180]), word-based tokenization is the predominant approach, which is more aligned with human's language cognition. However, word-based tokenization can yield different segmentation results for the same input in some languages (*e.g.*, Chinese word segmentation), generate a huge word vocabulary containing many low-frequency words, and also suffer from the "*out-of-vocabulary*" issue. Thus, several neural network models employ *character* as the minimum unit to derive the word representation (*e.g.*, a CNN word encoder in ELMo [21]). Recently, *subword tokenizers* have been widely used in Transformer based language models, typically including Byte-Pair Encoding tokenization, WordPiece tokenization and Unigram tokenization. HuggingFace has maintained an excellent online NLP course on tokenizer[20] with running examples, and we refer to the beginners to this course. Next, we briefly describe the three representative tokenization methods.

- *Byte-Pair Encoding (BPE) tokenization*. BPE was originally proposed as a general data compression algorithm in 1994 [181], and then adapted to NLP for tokenization [182]. It starts with a set of basic symbols (*e.g.*, the alphabets and boundary characters), and iteratively combine frequent pairs of two consecutive tokens in the corpus as new tokens (called *merge*). For each merge, the selection criterion is based on the co-occurrence frequency of two contiguous tokens: the top frequent pair would be selected. The merge process continues until it reaches the predefined size. Further, Byte-level BPE has been used to improve the tokenization quality for multilingual corpus (*e.g.*, the text containing non-ASCII characters) by considering *bytes* as the

20. https://huggingface.co/learn/nlp-course/chapter6

basic symbols for merge. Representative language models with this tokenization approach include GPT-2, BART, and LLaMA.

• *WordPiece tokenization*. WordPiece was a Google internal subword tokenization algorithm. It was originally proposed by Google in developing voice search systems [183]. Then, it was used in the neural machine translation system in 2016 [184], and was adopted as the word tokenizer for BERT in 2018 [23]. WordPiece has a very similar idea with BPE by iteratively merging consecutive tokens, whereas taking a slightly different selection criterion for the merge. To conduct the merge, it first trains a language model and employs it to score all possible pairs. Then, at each merge, it selects the pair that leads to the most increase in the likelihood of training data. Since Google has't released the official implementation of the WordPiece algorithm, HuggingFace gives a more intuitive selection measure in its online NLP course: a pair is scored by dividing the co-occurrence count by the product of the occurrence counts of two tokens in the pair based on training corpus.

• *Unigram tokenization*. Unlike BPE and WordPiece, Unigram tokenization [185] starts with a sufficiently large set of possible substrings or subtokens for a corpus, and iteratively removes the tokens in the current vocabulary until the expected vocabulary size is reached. As the selection criterion, it calculates the yielded increase in the likelihood of training corpus by assuming that some token was removed from current vocabulary. This step is conducted based on a trained unigram language model. To estimate the unigram language model, it adopts an expectation–maximization (EM) algorithm: at each iteration, we first find the currently optimal tokenization of words based on the old language model, and then re-estimate the probabilities of unigrams to update the language model. During this procedure, dynamic programming algorithms (*i.e.,* the Viterbi algorithm) are used to efficiently find the optimal decomposition way of a word given the language model. Representative models that adopt this tokenization approach include T5 and mBART.

Although it is expedient to leverage an existing tokenizer (*e.g.,* OPT [81] and GPT-3 [55] utilize the tokenizer of GPT-2 [26]), using a tokenizer specially designed for the pre-training corpus can be highly beneficial [69], especially for the corpus that consists of diverse domains, languages, and formats. Therefore, recent LLMs often train the customized tokenizers specially for the pre-training corpus with the SentencePiece library [186], which includes Byte-level BPE and Unigram tokenization. A note is that normalization techniques in BPE, such as NFKC [187], may degrade the tokenization performance [34, 59, 69]. When extending existing LLMs (*i.e.,* continual pre-training or instruction tuning), we should be also aware of the potential side effect with customized tokenizers. For example, LLaMA trains the BPE tokenizer based on a pre-training corpus mainly consisting of English texts, and the derived vocabulary might be less capable in processing non-English data, *e.g.,* taking longer inference latency to generate Chinese texts.

4.1.3 Effect of Pre-training Data on LLMs

Unlike small-scale PLMs, it is usually infeasible to iterate the pre-training of LLMs multiple times, due to the huge demand for computational resources. Thus, it is particularly important to construct a well-prepared pre-training corpus before training a LLM. In this part, we discuss how the quality and distribution of the pre-training corpus potentially influence the performance of LLMs.

Mixture of Sources. As discussed before, pre-training data from different domains or scenarios has distinct linguistic characteristics or semantic knowledge. By pre-training on a mixture of text data from diverse sources, LLMs can acquire a broad scope of knowledge and may exhibit a strong generalization capacity. Thus, when mixing different sources, it is suggested to include as many high-quality data sources as possible, and carefully set the distribution of pre-training data, since it is also likely to affect the performance of LLMs on downstream tasks [59]. Gopher [59] conducts the ablation experiment on data distribution to examine the impact of mixed sources on downstream tasks. Experimental results on the LAMBADA dataset [188] show that increasing the proportion of books data can improve the capacity of the model in capturing long-term dependencies from text, and increasing the proportion of the C4 dataset [73] leads to performance improvement on the C4 validation dataset [59]. However, as a side effect, training on excessive data about a certain domain would affect the generalization capability of LLMs on other domains [35, 59]. Therefore, it is suggested that researchers should carefully determine the proportion of data from different domains in the pre-training corpus, in order to develop LLMs that better meet their specific needs. The readers can refer to Figure 5 for a comparison of the data sources for different LLMs.

Amount of Pre-training Data. For pre-training an effective LLM, it is important to collect sufficient high-quality data that satisfies the data quantity demand of the LLM. Existing studies have found that with the increasing parameter scale in the LLM, more data is also required to train the model [34, 57]: a similar scaling law as model size is also observed in data size, with respect to model performance. A recent study has shown that a number of existing LLMs suffer from sub-optimal training due to inadequate pre-training data [34]. By conducting extensive experiments, it further demonstrates increasing the model size and data size in equal scales can lead to a more compute-efficient model (*i.e.,* the Chinchilla model), for a given compute budget. More recently, LLaMA [57] shows that with more data and longer training, smaller models can also achieve good performance. Overall, it is suggested that researchers should pay more attention to the amount of high-quality data for adequately training the model, especially when scaling the model parameters.

Quality of Pre-training Data. Existing work has shown that pre-training on the low-quality corpus, such as noisy, toxic, and duplicate data, may hurt the performance of models [59, 174, 176, 179]. For developing a well-performing LLM, it is crucial to consider both the quantity and the quality of the collected training data. Recent studies, such as T5 [73], GLaM [99], and Gopher [59], have investigated the influence of data quality on the performance of downstream tasks. By comparing the performance of models trained on the filtered and unfiltered corpus, they reach

the same conclusion that pre-training LLMs on cleaned data can improve the performance. More specifically, the duplication of data may result in "*double descent*" (referring to the phenomenon of performance initially deteriorating and subsequently improving) [174, 189], or even overwhelm the training process [174]. In addition, it has been shown that duplicate data degrades the ability of LLMs to copy from the context, which might further affect the generalization capacity of LLMs using in-context learning [174]. Therefore, as suggested in [56, 59, 69], it is essential to incorporate preprocessing methods on the pre-training corpus carefully (as illustrated in Section 4.1.2), to improve stability of the training process and avoid affecting the model performance.

4.2 Architecture

In this section, we review the architecture design of LLMs, *i.e.,* mainstream architecture, pre-training objective, and detailed configuration. Table 3 presents the model cards of several representative LLMs with public details.

4.2.1 Typical Architectures

Due to the excellent parallelizability and capacity, the Transformer architecture [22] has become the de facto backbone to develop various LLMs, making it possible to scale language models to hundreds or thousands of billions of parameters. In general, the mainstream architectures of existing LLMs can be roughly categorized into three major types, namely encoder-decoder, causal decoder, and prefix decoder, as shown in Figure 7.

Encoder-decoder Architecture. The vanilla Transformer model is built on the encoder-decoder architecture [22], which consists of two stacks of Transformer blocks as the encoder and decoder, respectively. The encoder adopts stacked multi-head self-attention layers to encode the input sequence for generating its latent representations, while the decoder performs cross-attention on these representations and autoregressively generates the target sequence. Encoder-decoder PLMs (*e.g.,* T5 [73] and BART [24]) have shown effectiveness on a variety of NLP tasks. So far, there are only a small number of LLMs that are built based on the encoder-decoder architecture, *e.g.,* Flan-T5 [64]. We leave a detailed discussion about the architecture selection in Section 4.2.5.

Causal Decoder Architecture. The causal decoder architecture incorporates the unidirectional attention mask, to guarantee that each input token can only attend to the past tokens and itself. The input and output tokens are processed in the same fashion through the decoder. As representative language models of this architecture, the GPT-series models [26, 55, 109] are developed based on the causal-decoder architecture. In particular, GPT-3 [55] has successfully demonstrated the effectiveness of this architecture, also showing an amazing in-context learning capability of LLMs. Interestingly, GPT-1 [109] and GPT-2 [26] do not exhibit such superior abilities as those in GPT-3, and it seems that scaling plays an important role in increasing the model capacity of this model architecture. So far, the causal decoders have been widely adopted as the architecture of LLMs by various existing LLMs, such as OPT [81], BLOOM [69], and Gopher [59]. Note that both the causal decoder and prefix decoder discussed next belong to decoder-only architectures. When mentioning "decoder-only architecture", it mainly refers to the causal decoder architecture in existing literature, unless specified.

Prefix Decoder Architecture. The prefix decoder architecture (*a.k.a.,* non-causal decoder [190]) revises the masking mechanism of causal decoders, to enable performing bidirectional attention over the prefix tokens [191] and unidirectional attention only on generated tokens. In this way, like the encoder-decoder architecture, the prefix decoders can bidirectionally encode the prefix sequence and autoregressively predict the output tokens one by one, where the same parameters are shared during encoding and decoding. Instead of pre-training from scratch, a practical suggestion is to continually train causal decoders and then convert them into prefix decoders for accelerating convergence [29], *e.g.,* U-PaLM [105] is derived from PaLM [56]. Existing representative LLMs based on prefix decoders include GLM-130B [84] and U-PaLM [105].

For the three types of architectures, we can also consider extending them via the mixture-of-experts (MoE) scaling, in which a subset of neural network weights for each input are sparsely activated, *e.g.,* Switch Transformer [25] and GLaM [99]. It has been shown that substantial performance improvement can be observed by increasing either the number of experts or the total parameter size [192].

Emergent Architectures. The conventional Transformer architectures typically suffer from quadratic computational complexity. Because of this, efficiency has become an important issue when training and making inference with long inputs. To improve efficiency, some studies aim to devise new architectures for language modeling, including parameterized state space models (*e.g.,* S4 [193], GSS [194], and H3 [195]), long convolutions like Hyena [196], and Transformer-like architectures that incorporate recursive update mechanisms (*e.g.,* RWKV [197] and RetNet [198]). The key merits of these new architectures are twofold. First, these models can generate outputs recursively like RNNs, meaning that they only need to refer to the single previous state during decoding. It makes the decoding process more efficient as it eliminates the need to revisit all previous states as in conventional Transformers. Second, these models have the capacity to encode an entire sentence in parallel like Transformers. This contrasts with conventional RNNs which has to encode sentences on a token-by-token basis. Thus, they can benefit from the parallelism of GPUs with techniques such as Parallel Scan [199, 200], FFT [196, 197], and Chunkwise Recurrent [198]. These techniques enable models with these new architectures to be trained in a highly parallel and efficient manner.

4.2.2 Detailed Configuration

Since the launch of Transformer [22], various improvements have been proposed to enhance its training stability, performance, and computational efficiency. In this part, we will discuss the corresponding configurations for four major parts of the Transformer, including normalization, position embeddings, activation functions, and attention and bias.

TABLE 3: Model cards of several selected LLMs with public configuration details. Here, PE denotes position embedding, #L denotes the number of layers, #H denotes the number of attention heads, d_{model} denotes the size of hidden states, and MCL denotes the maximum context length during training.

Model	Category	Size	Normalization	PE	Activation	Bias	#L	#H	d_{model}	MCL
GPT3 [55]	Causal decoder	175B	Pre LayerNorm	Learned	GeLU	✓	96	96	12288	2048
PanGU- α [75]	Causal decoder	207B	Pre LayerNorm	Learned	GeLU	✓	64	128	16384	1024
OPT [81]	Causal decoder	175B	Pre LayerNorm	Learned	ReLU	✓	96	96	12288	2048
PaLM [56]	Causal decoder	540B	Pre LayerNorm	RoPE	SwiGLU	×	118	48	18432	2048
BLOOM [69]	Causal decoder	176B	Pre LayerNorm	ALiBi	GeLU	✓	70	112	14336	2048
MT-NLG [100]	Causal decoder	530B	-	-	-	-	105	128	20480	2048
Gopher [59]	Causal decoder	280B	Pre RMSNorm	Relative	-	-	80	128	16384	2048
Chinchilla [34]	Causal decoder	70B	Pre RMSNorm	Relative	-	-	80	64	8192	-
Galactica [35]	Causal decoder	120B	Pre LayerNorm	Learned	GeLU	×	96	80	10240	2048
LaMDA [63]	Causal decoder	137B	-	Relative	GeGLU	-	64	128	8192	-
Jurassic-1 [94]	Causal decoder	178B	Pre LayerNorm	Learned	GeLU	✓	76	96	13824	2048
LLaMA [57]	Causal decoder	65B	Pre RMSNorm	RoPE	SwiGLU	×	80	64	8192	2048
LLaMA 2 [90]	Causal decoder	70B	Pre RMSNorm	RePE	SwiGLU	×	80	64	8192	4096
Falcon [127]	Causal decoder	40B	Pre LayerNorm	RoPE	GeLU	×	60	64	8192	2048
GLM-130B [84]	Prefix decoder	130B	Post DeepNorm	RoPE	GeGLU	✓	70	96	12288	2048
T5 [73]	Encoder-decoder	11B	Pre RMSNorm	Relative	ReLU	×	24	128	1024	512

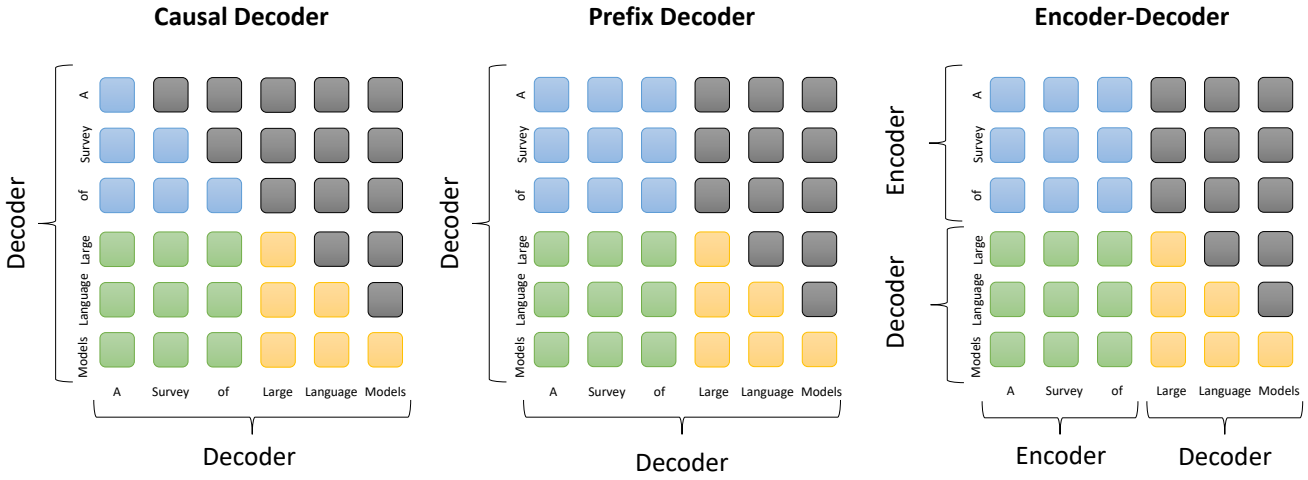


Fig. 7: A comparison of the attention patterns in three mainstream architectures. Here, the blue, green, yellow and grey rounded rectangles indicate the attention between prefix tokens, attention between prefix and target tokens, attention between target tokens, and masked attention respectively.

To make this survey more self-contained, we present the detailed formulations for these configurations in Table 4.

Normalization Methods. Training instability is a challenging issue for pre-training LLMs. To alleviate this issue, normalization is a widely adopted strategy to stabilize the training of neural networks. In the vanilla Transformer [22], LayerNorm [202] is employed. Recently, several advanced normalization techniques have been proposed as alternatives to LayerNorm, *e.g.*, RMSNorm, and DeepNorm.

• *LayerNorm.* In the early research, BatchNorm [211] is a commonly used normalization method. However, it is difficult to deal with sequence data of variable lengths and small-batch data. Thus, LayerNorm [202] is introduced to conduct layerwise normalization. Specifically, the mean and variance over all activations per layer are calculated to re-center and re-scale the activations.

• *RMSNorm.* To improve the training speed of LayerNorm (LN), RMSNorm [203] is proposed by re-scaling the activations with only the root mean square (RMS) of the summed activations, instead of the mean and variance. Related research has demonstrated its superiority in training speed and performance on Transformer [212]. Representative models that adopt RMSNorm include Gopher [59] and Chinchilla [34].

• *DeepNorm.* DeepNorm is proposed by Microsoft [204] to stabilize the training of deep Transformers. With DeepNorm as residual connections, Transformers can be scaled up to 1,000 layers [204], which has shown the advantages of stability and good performance. It has been adopted by GLM-130B [84].

Normalization Position. In addition to the normalization method, normalization position also plays a crucial role in the LLMs. There are generally three choices for the normalization position, *i.e.*, post-LN, pre-LN, and sandwich-LN.

• *Post-LN.* Post-LN is used in the vanilla Transformer [22], which is placed between residual blocks. However, existing work has found that the training of Transformers with post-LN tends to be instable due to the large

TABLE 4: Detailed formulations for the network configurations. Here, Sublayer denotes a FFN or a self-attention module in a Transformer layer, d denotes the size of hidden states, $\mathbf{p}_i$ denotes position embedding at position i, A_{ij} denotes the attention score between a query and a key, r_{i-j} denotes a learnable scalar based on the offset between the query and the key, and $\mathbf{R}_{\theta,t}$ denotes a rotary matrix with rotation degree $t \cdot \theta$.

Configuration	Method	Equation
Normalization position	Post Norm [22]	$\text{Norm}(\mathbf{x}+\text{Sublayer}(\mathbf{x}))$
	Pre Norm [26]	$\mathbf{x}+\text{Sublayer}(\text{Norm}(\mathbf{x}))$
	Sandwich Norm [201]	$\mathbf{x}+\text{Norm}(\text{Sublayer}(\text{Norm}(\mathbf{x})))$
Normalization method	LayerNorm [202]	$\frac{\mathbf{x}-\mu}{\sqrt{\sigma}} \cdot \gamma + \beta, \quad \mu = \frac{1}{d}\sum_{i=1}^{d} x_i, \quad \sigma = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(x_i-\mu))^2}$
	RMSNorm [203]	$\frac{\mathbf{x}}{\text{RMS}(\mathbf{x})} \cdot \gamma, \quad \text{RMS}(\mathbf{x}) = \sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}$
	DeepNorm [204]	$\text{LayerNorm}(\alpha \cdot \mathbf{x} + \text{Sublayer}(\mathbf{x}))$
Activation function	ReLU [205]	$\text{ReLU}(\mathbf{x}) = \max(\mathbf{x}, \mathbf{0})$
	GeLU [206]	$\text{GeLU}(\mathbf{x}) = 0.5\mathbf{x} \otimes [1 + \text{erf}(\mathbf{x}/\sqrt{2})], \quad \text{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2} dt$
	Swish [207]	$\text{Swish}(\mathbf{x}) = \mathbf{x} \otimes \text{sigmoid}(\mathbf{x})$
	SwiGLU [208]	$\text{SwiGLU}(\mathbf{x_1}, \mathbf{x}_2) = \text{Swish}(\mathbf{x_1}) \otimes \mathbf{x_2}$
	GeGLU [208]	$\text{GeGLU}(\mathbf{x_1}, \mathbf{x}_2) = \text{GeLU}(\mathbf{x_1}) \otimes \mathbf{x_2}$
Position embedding	Absolute [22]	$\mathbf{x}_i = \mathbf{x}_i + \mathbf{p}_i$
	Relative [73]	$A_{ij} = \mathbf{W}_q \mathbf{x}_i \mathbf{x}_j^T \mathbf{W}_k^T + r_{i-j}$
	RoPE [209]	$A_{ij} = \mathbf{W}_q \mathbf{x}_i \mathbf{R}_{\theta,i-j} \mathbf{x}_j^T \mathbf{W}_k^T$
	Alibi [210]	$A_{ij} = \mathbf{W}_q \mathbf{x}_i \mathbf{R}_{\theta,i-j} \mathbf{x}_j^T \mathbf{W}_k^T A_{ij} = \mathbf{W}_q \mathbf{x}_i \mathbf{x}_j^T \mathbf{W}_k^T - m(i-j)$

gradients near the output layer [213]. Thus, post-LN is rarely employed in existing LLMs except combined with other strategies (*e.g.*, combining post-LN with pre-LN in GLM-130B [84]).

• *Pre-LN*. Different from post-LN, pre-LN [214] is applied before each sub-layer, and an additional LN is placed before the final prediction. Compared with post-LN, the Transformers with pre-LN are more stable in training. However, it performs worse than the variants with post-LN [215]. Despite the decreasing performance, most LLMs still adopt pre-LN due to the training stability. However, one exception is that pre-LN has been found unstable in GLM when training models more than 100B parameters [84].

• *Sandwich-LN*. Based on pre-LN, Sandwich-LN [201] adds extra LN before the residual connections to avoid the value explosion issues in Transformer layer outputs. However, it has been found that Sandwich-LN sometimes fails to stabilize the training of LLMs and may lead to the collapse of training [84].

Activation Functions. To obtain good performance, activation functions also need to be properly set in feed-forward networks. In existing LLMs, GeLU activations [216] are widely used. Specially, in the latest LLMs (*e.g.*, PaLM and LaMDA), variants of GLU activation [208, 217] have also been utilized, especially the SwiGLU and GeGLU variants, which often achieve better performance in practice [212]. However, compared with GeLU, they require extra parameters (about 50%) in the feed-forward networks [218].

Position Embeddings. Since the self-attention modules in Transformer are permutation equivariant, position embeddings (PE) are employed to inject absolute or relative position information for modeling sequences.

• *Absolute position embedding*. In the vanilla Transformer [22], absolute position embeddings are employed. At the bottoms of the encoder and the decoder, the absolute positional embeddings are added to the input embeddings. There are two variants of absolute position embeddings proposed in the vanilla Transformer [22], *i.e.*, sinusoidal and learned position embeddings, where the latter is commonly used in existing pre-trained language models.

• *Relative position embedding*. Unlike absolute position embeddings, relative positional embeddings are generated according to the offsets between keys and queries [219]. A popular variant of relative PE was introduced in Transformer-XL [220, 221]. The calculation of attention scores between keys and queries has been modified to introduce learnable embeddings corresponding to relative positions. T5 [73] further simplified relative positional embeddings, which was subsequently adopted by Gopher [59]. Specifically, it adds learnable scalars to the attention scores, where the scalars are calculated based on the distances between the positions of the query and the key. Compared with the absolute PE, Transformers with relative position embedding can generalize to sequences longer than those sequences for training, *i.e.*, extrapolation [210].

• *Rotary Position Embedding*. Rotary position embedding (RoPE) [209] sets specific rotatory matrices based on the absolute position of each token. The scores between keys and queries can be computed with relative position information. Due to the excellent performance and the long-term decay property, RoPE is widely adopted in the latest LLMs, *e.g.*, PaLM [56] and LLaMA [57]. Based on RoPE, xPos [222] further improves the translation invariance and length extrapolation of Transformer. At each dimension of the rotation degree vector, xPos adds a special exponential decay that is smaller when the rotation degree is larger. It can alleviate the unstable phenomenon during training as the distance increases.

• *ALiBi*. ALiBi [210] is proposed to improve the extrapolation of Transformer. Similar to relative position embedding, it biases attention scores with a penalty based on the distances between keys and queries. Different from the relative positional embedding methods like T5 [73], the penalty scores in ALiBi are pre-defined without any trainable parameters. Empirical results in [210] have shown that ALiBi

has a better extrapolation performance on sequences that are longer than those for training than several popular position embedding methods such as sinusoidal PE [22], RoPE [209], and T5 bias [73]. In addition, it has been shown that ALiBi can also improve training stability in BLOOM [69].

Attention. Attention mechanism is a critical component of Transformer. It allows the tokens across the sequence to interact with each other and compute the representations of the input and output sequence.

• *Full attention*. In the vanilla Transformer [22], the attention mechanism is conducted in a pairwise way, considering the relations between all token pairs in a sequence. It adopts scaled dot-product attention, in which the hidden states are mapped into queries, keys, and values. Additionally, Transformer uses multi-head attention instead of single attention, projecting the queries, keys, and values with different projections in different heads. The concatenation of the output of each head is taken as the final output.

• *Sparse attention*. A crucial challenge of full attention is the quadratic computational complexity, which becomes a burden when dealing with long sequences. Therefore, various efficient Transformer variants are proposed to reduce the computational complexity of the attention mechanism [223, 224]. For instance, locally banded sparse attention (*i.e.,* Factorized Attention [225] has been adopted in GPT-3 [55]. Instead of the whole sequence, each query can only attend to a subset of tokens based on the positions.

• *Multi-query/grouped-query attention*. Multi-query attention refers to the attention variant where different heads share the same linear transformation matrices on the keys and values [226]. It can significantly save computation costs with only a minor sacrifice in model quality. Representative models with multi-query attention include PaLM [56] and StarCoder [89]. To make a trade-off between multi-query attention and multi-head attention, grouped-query attention (GQA) [227] has been explored. In GQA, heads are assigned into different groups, and those heads that belong to the same group will share the same transformation matrices. Specially, GQA has been adopted and empirically tested in the recently released LLaMA 2 model [90].

• *FlashAttention*. Different from most existing approximate attention methods that trade-off model quality to improve the computing efficiency, FlashAttention [228] proposes to optimize the speed and memory consumption of attention modules on GPUs from an IO-aware perspective. There exist different levels of memory on modern GPUs, *e.g.,* SRAM with a fast IO and HBM with a relatively slow IO. FlashAttention organizes the input into blocks and introduces necessary recomputation, both to make better use of the fast memory SRAM. Implemented as a fused kernel in CUDA, FlashAttention has been integrated into PyTorch [157], DeepSpeed [65], and Megatron-LM [66]. The updated version FlashAttention-2 [229] further optimizes the work partitioning of GPU thread blocks and warps, leading to around 2× speedup when compared to the original FlashAttention.

• *PagedAttention*. It has been observed when LLM are deployed on servers, GPU memory is largely occupied by cached attention key and value tensors (called *KV cache*). The major reason is that the input lengths are often varied, leading to fragmentation and over-reservation issues. Inspired by the classic paging technique in operating systems, PagedAttention has been proposed to improve the memory efficiency and throughput of deployed LLMs [230]. In detail, PagedAttention partitions each sequence into subsequences, and the corresponding KV caches of these subsequences are allocated into non-contiguous physical blocks. The paging technique increases the GPU utilization and enables efficient memory sharing in parallel sampling.

To put all these discussions together, we summarize the suggestions from existing literature for detailed configuration. For stronger generalization and training stability, it is suggested to choose the pre RMSNorm for layer normalization, and SwiGLU or GeGLU as the activation function. In addition, LN may not be used immediately after embedding layers, which is likely to incur performance degradation. As for position embeddings, RoPE or ALiBi is a better choice since it performs better on long sequences.

4.2.3 Pre-training Tasks

Pre-training plays a key role that encodes general knowledge from large-scale corpus into the massive model parameters. For training LLMs, there are two commonly used pre-training tasks, namely language modeling and denoising autoencoding.

Language Modeling. The language modeling task (LM) is the most commonly used objective to pre-train decoder-only LLMs, *e.g.,* GPT3 [55] and PaLM [56]. Given a sequence of tokens $\mathbf{x} = \{x_1, \ldots, x_n\}$, the LM task aims to autoregressively predict the target tokens x_i based on the preceding tokens $x_{<i}$ in a sequence. A general training objective is to maximize the following likelihood:

$$\mathcal{L}_{LM}(\mathbf{x}) = \sum_{i=1}^{n} \log P(x_i|\mathbf{x}_{<i}). \tag{4}$$

Since most language tasks can be cast as the prediction problem based on the input, these decoder-only LLMs might be potentially advantageous to implicitly learn how to accomplish these tasks in a unified LM way. Some studies have also revealed that decoder-only LLMs can be naturally transferred to certain tasks by autoregressively predicting the next tokens [26, 55], without fine-tuning. An important variant of LM is the *prefix language modeling* task, which is designed for pre-training models with the prefix decoder architecture. The tokens within a randomly selected prefix would not be used in computing the loss of prefix language modeling. With the same amount of tokens seen during pre-training, prefix language modeling performs slightly worse than language modeling, since fewer tokens in the sequence are involved for model pre-training [29].

Denoising Autoencoding. In addition to conventional LM, the denoising autoencoding task (DAE) has also been widely used to pre-train language models [24, 73]. The inputs $\mathbf{x}_{\setminus\tilde{\mathbf{x}}}$ for DAE task are corrupted text with randomly replaced spans. Then, the language models are trained to recover the replaced tokens $\tilde{\mathbf{x}}$. Formally, the training objective of DAE is denoted as follows:

$$\mathcal{L}_{DAE}(\mathbf{x}) = \log P(\tilde{\mathbf{x}}|\mathbf{x}_{\setminus\tilde{\mathbf{x}}}). \tag{5}$$

I am sleepy. I start a pot of _____

coffee	0.661	strong	0.008	soup	0.005
water	0.119	black	0.008	...	...
tea	0.057	hot	0.007	happy	4.3e-6
rice	0.017	oat	0.006	Boh	4.3e-6
chai	0.012	beans	0.006	...	...

Fig. 8: The probability distribution over the vocabulary in descending order for the next token of the context *"I am sleepy. I start a pot of"*. For ease of discussion, this example is given in word units instead of subword units.

However, the DAE task seems to be more complicated in implementation than LM task. As a result, it has not been widely used to pre-train large language models. Existing LLMs that take DAE as pre-training objectives include T5 [73] and GLM-130B [84]. These models are mainly trained to recover the replaced spans in an autoregressive way.

Mixture-of-Denoisers. Mixture-of-Denoisers (MoD) [80], also known as UL2 loss, was introduced as a unified objective for pre-training language models. MoD regards both LM and DAE objectives as different types of denoising tasks, namely S-denoiser (LM), R-denoiser (DAE, short span and low corruption), and X-denoiser (DAE, long span or high corruption). Among the three denoising tasks, S-denoiser is similar to the conventional LM objective (Equation (4)), while R-denoiser and X-denoiser are similar to DAE objectives (Equation (5)) but differ from each other in the lengths of spans and ratio of corrupted text. For input sentences started with different special tokens (*i.e.*, {`[R]`, `[S]`, `[X]`}), the model will be optimized using the corresponding denoisers. MoD has been applied in the latest PaLM 2 model [107].

4.2.4 Decoding Strategy

After the LLMs have been pre-trained, it is essential to employ a specific decoding strategy to generate the appropriate output from the LLMs.

Background. We start the discussion with the prevalent decoder-only architecture, and introduce the auto-regressive decoding mechanism. Since such LLMs are pre-trained based on the language modeling task (Equation 4), a basic decoding method is *greedy search* that predicts the most likely token at each step based on the previously generated tokens, formally modeled as:

$$x_i = \arg\max_x P(x|\mathbf{x}_{<i}), \tag{6}$$

where x_i is the token with with the highest probability at i-th step of generation conditioned on the context $\mathbf{x}_{<i}$. For instance in Figure 8, when predicting the next token of the sentence *"I am sleepy. I start a pot of"*, greedy search selects the token "coffee" which has the highest probability at the current step. Greedy search can achieve satisfactory results in text generation tasks (*e.g.*, machine translation and text summarization), in which the output is highly dependent on the input [231]. However, in terms of open-ended generation tasks (*e.g.*, story generation and dialog), greedy search sometimes tends to generate awkward and repetitive sentences [232].

As another alternative decoding strategy, sampling-based methods are proposed to randomly select the next token based on the probability distribution to enhance the randomness and diversity during generation:

$$x_i \sim P(x|\mathbf{x}_{<i}). \tag{7}$$

For the example in Figure 8, sampling-based methods will sample the word "coffee" with higher probability while also retaining the possibilities of selecting the rest words, "water", "tea", "rice", *etc.*

Not limited to the decoder-only architecture, these two decoding methods can be generally applied to encoder-decoder models and prefix decoder models in a similar way.

Improvement for Greedy Search. Selecting the token with the highest probability at each step may result in overlooking a sentence with a higher overall probability but a lower local estimation. Next, we introduce several improvement strategies to alleviate this issue.

• *Beam search.* Beam search [233] retains the sentences with the n (beam size) highest probabilities at each step during the decoding process, and finally selects the generated response with the top probability. Typically, the beam size is configured within the range of 3 to 6. However, opting for a larger beam size might result in a decline in performance [234].

• *Length penalty.* Since beam search favours shorter sentences, imposing length penalty (*a.k.a.*, length normalization) is a commonly used technique [235] to overcome this issue, which normalizes the sentence probability according to the sentence length (divided by an exponential power α of the length).

Besides, some researchers [236] propose to penalize the generation of previously generated tokens or n-grams to alleviate the issue of repetitive generation. In addition, diverse beam search [237] can be leveraged to produce a set of diverse outputs based on the same input.

Improvement for Random Sampling. Sampling-based methods sample the token over the whole vocabulary, which may select wrong or irrelevant tokens (*e.g.*, "happy" and "Boh" in Figure 8) based on the context. To improve the generation quality, several strategies have been proposed for mitigating or preventing the selection of words with exceedingly low probabilities.

• *Temperature sampling.* To modulate the randomness of sampling, a practical method is to adjust the temperature coefficient of the softmax function for computing the probability of the j-th token over the vocabulary:

$$P(x_j|\mathbf{x}_{<i}) = \frac{\exp\left(l_j/t\right)}{\sum_{j'} \exp\left(l_{j'}/t\right)}, \tag{8}$$

where $l_{j'}$ is the logits of each word and t is the temperature coefficient. Reducing the temperature t increases the chance of selecting words with high probabilities while decreases the chances of selecting words with low probabilities. When t is set to 1, it becomes the default random sampling; when t is approaching 0, it is equivalent to greedy search. In addition, when t goes to infinity, it degenerates to uniform sampling.

• *Top-k sampling.* Different from temperature sampling, top-k sampling directly truncates the tokens with lower probability and only samples from the tokens with the top k highest probabilities [238]. For example in Figure 8, top-5 sampling will sample from the words "coffee", "water", "tea", "rice", and "chai" from their re-scaled probabilities.

• *Top-p sampling.* Since top-k sampling does not consider the overall possibility distribution, a constant value of k may be not be suitable for different contexts. Therefore, top-p sampling (*a.k.a.,* nucleus sampling) is proposed by sampling from the smallest set having a cumulative probability above (or equal to) p [232]. In practice, the smallest set can be constructed by gradually adding tokens from the vocabulary sorted in descending order of generative probability, until their cumulative value exceeds p.

Recently, researchers have also explored other sampling strategies for LLMs. For instance, η-sampling further improves top-p sampling by introducing a dynamic threshold based on the probability distribution. Furthermore, contrastive search [239] and typical sampling [240] can be utilized to improve the generation coherence during decoding.

Efficient Decoding Strategies. Considering the essence of auto-regressive generation in LLMs, the generation process takes progressively more time as the length of sequences increases. As a result, several research work investigates the methods to accelerate the decoding process. Speculative decoding [241, 242] first leverages a compact but efficient model (*e.g.,* a n-gram model or a small PLM) to generate short segments and then utilizes the LLM to verify and correct these drafts. This method leads to a notable $2\times$ to $3\times$ speedup without comprising the generation quality. SpecInfer [243] further proposes a learning-based method to combine several small models to increase the possibility of coverage. In addition, token-level early-exit techniques have been proposed enabling the generation of a token at lower Transformer layers, rather than passing through all the layers [244]. It can attain greater speedup, but at the cost of sacrificing generation quality.

Practical Settings. In practice, existing libraries (*e.g.,* Transformers [151]) and public APIs of LLMs (*e.g.,* OpenAI) have supported various decoding strategies to serve different scenarios of text generation. Next, we present the decoding settings of several representative LLMs:

• *T5* [73] utilizes greedy search as the default setting and applies beam search (beam size of 4) with a length penalty of 0.6 for translation and summarization tasks.

• *GPT-3* [55] employs beam search with a beam size of 4 and a length penalty of 0.6 for all generation tasks.

• *Alpaca* [128] utilizes sampling-based strategies with top-k ($k = 50$), top-p ($p = 0.9$), and temperature of 0.7 for open-ended generation.

• *LLaMA* [57] applies diverse decoding strategies tailored to specific tasks. For instance, it employs the greedy search for question answering tasks while utilizes a sampling strategy with the temperature settings of 0.1 (pass@1) and 0.8 (pass@100) for code generation.

• *OpenAI API* supports several basic decoding strategies, including greedy search (by setting `temperature` to 0), beam search (with the setting `best_of`), temperature sampling (with the setting `temperature`), nucleus sampling (with the setting `top_p`). It also introduce parameters `presence_penalty` and `frequency_penalty` to control the repetition degree of generation. According to the OpenAI's document, their APIs would produce different outputs even if the input and the hyper-parameters are the same. Setting temperature to 0 can yield more deterministic outputs, albeit with a slight chance of variability.

4.2.5 Summary and Discussion

The choice of architecture and pre-training tasks may incur different inductive biases for LLMs, which would lead to different model capacities. In this part, we discuss on several two open issues about LLM architecture.

Architecture Choice. In earlier literature of pre-trained language models, there are lots of discussions on the effects of different architectures [29, 80]. However, most LLMs are developed based on the causal decoder architecture, and there still lacks a theoretical analysis on its advantage over the other alternatives. Next, we briefly summarize existing discussions on this issue.

• By pre-training with the LM objective, it seems that causal decoder architecture can achieve a superior zero-shot and few-shot generalization capacity. Existing research has shown that without multi-task fine-tuning, the causal decoder has better zero-shot performance than other architectures [29]. The success of GPT-3 [55] has demonstrates that the large causal decoder model can be a good few-shot learner. In addition, instruction tuning and alignment tuning discussed in Section 5 have been proven to further enhance the capability of large causal decoder models [61, 62, 64].

• Scaling law has been widely observed in causal decoders. By scaling the model size, the dataset size, and the total computation, the performance of causal decoders can be substantially improved [30, 55]. Thus, it has become an important strategy to increase the model capacity of the causal decoder via scaling. However, more detailed investigation on encoder-decoder models is still lacking, and more efforts are needed to investigate the performance of encoder-decoder models at a large scale.

More research efforts about the discussions on architectures and pre-training objectives are in need to analyze how the choices of the architecture and pre-training tasks affect the capacity of LLMs, especially for encoder-decoder architectures. Besides the major architecture, the detailed configuration of LLM is also worth attention, which has been discussed in Section 4.2.2.

Long Context. One of the drawbacks of Transformer-based LMs is the limited context length due to the quadratic computational costs in both time and memory. Meanwhile, there is an increasing demand for long context windows in applications such as PDF processing and story writing [245]. A variant of ChatGPT with 16K tokens as the context window has recently been released, which is much longer than the initial 4K tokens. Additionally, the context window of GPT-4 has been extended to 32K tokens [46]. Next, we discuss two important factors related to long context modeling.

• *Extrapolation.* In real-world applications, it is possible for LLMs to process long input texts that exceed the maximum length of the training corpus. The ability of

encoding longer texts is often referred to as *extrapolation capability* [210]. Several position embedding methods, such as RoPE [209] and T5 bias [73], have been empirically validated to possess certain extrapolation capabilities [210]. Specifically, LMs equipped with ALiBi [210] have been shown to maintain relatively stable perplexity on sequences even ten times longer than those for training. There are also efforts like xPos [222] to enhance the extrapolation ability of RoPE by improving the design of rotation matrix.

• *Efficiency.* In order to reduce the quadratic computational cost in attention modules, several studies design highly efficient attention computation methods that can make the memory consumption scales approximately linearly, exemplified by sparse or linear attentions [225, 246–249]. In addition to the algorithmic improvements, another important work, FlashAttention [228], improves the efficiency from a system-level perspective (*i.e.,* GPU memory IO efficiency). With the same computing budget, one can train LLMs with longer context windows. Several studies also aim to devise new architectures rather than conventional Transformer modules to address the efficiency issue, such as RWKV [197] and RetNet [198]. One can refer to Section 4.2.1 for a more detailed introduction of these attempts.

Why does Predicting the Next Word Works?

The essence of decoder-only architecture is to *accurately predict the next word* for reconstructing the pre-training data. Till now, there has been no formal study that theoretically demonstrates its advantage over other architectures. An interesting explanation was from Ilya Sutskever during the interview held by Jensen Huang[a]. The original transcript from the interview was copied below[b]:

```
Say you read a detective novel. It's
like complicated plot, a storyline,
different characters, lots of events,
mysteries like clues, it's unclear.
Then, let's say that at the last
page of the book, the detective has
gathered all the clues, gathered
all the people and saying, "okay,
I'm going to reveal the identity of
whoever committed the crime and that
person's name is". Predict that word.
...
Now, there are many different words.
But predicting those words better and
better, the understanding of the text
keeps on increasing. GPT-4 predicts
the next word better.
```

a. https://www.nvidia.com/en-us/on-demand/session/gtcspring23-S52092/
b. https://lifearchitect.ai/ilya/

4.3 Model Training

In this part, we review the important settings, techniques, or tricks for training LLMs.

4.3.1 Optimization Setting

For parameter optimization of LLMs, we present the commonly used settings for batch training, learning rate, optimizer, and training stability.

Batch Training. For language model pre-training, existing work generally sets the batch size to a large number (*e.g.,* 2,048 examples or 4M tokens) to improve the training stability and throughput. For LLMs such as GPT-3 and PaLM, they have introduced a new strategy that dynamically increases the batch size during training, ultimately reaching a million scale. Specifically, the batch size of GPT-3 is gradually increasing from 32K to 3.2M tokens. Empirical results have demonstrated that the dynamic schedule of batch size can effectively stabilize the training process of LLMs [56].

Learning Rate. Existing LLMs usually adopt a similar learning rate schedule with the warm-up and decay strategies during pre-training. Specifically, in the initial 0.1% to 0.5% of the training steps, a linear warm-up schedule is employed for gradually increasing the learning rate to the maximum value that ranges from approximately 5×10^{-5} to 1×10^{-4} (*e.g.,* 6×10^{-5} for GPT-3). Then, a cosine decay strategy is adopted in the subsequent steps, gradually reducing the learning rate to approximately 10% of its maximum value, until the convergence of the training loss.

Optimizer. The Adam optimizer [250] and AdamW optimizer [251] are widely utilized for training LLMs (*e.g.,* GPT-3), which are based on adaptive estimates of lower-order moments for first-order gradient-based optimization. Commonly, its hyper-parameters are set as follows: $\beta_1 = 0.9$, $\beta_2 = 0.95$ and $\epsilon = 10^{-8}$. Meanwhile, the Adafactor optimizer [252] has also been utilized in training LLMs (*e.g.,* PaLM and T5), which is a variant of the Adam optimizer specially designed for conserving GPU memory during training. The hyper-parameters of the Adafactor optimizer are set as: $\beta_1 = 0.9$ and $\beta_2 = 1.0 - k^{-0.8}$, where k denotes the number of training steps.

Stabilizing the Training. During the pre-training of LLMs, it often suffers from the training instability issue, which may cause the model collapse. To address this issue, weight decay and gradient clipping have been widely utilized, where existing studies [55, 69, 81, 84, 100] commonly set the threshold of gradient clipping to 1.0 and weight decay rate to 0.1. However, with the scaling of LLMs, the training loss spike is also more likely to occur, leading to unstable training. To mitigate this problem, PaLM [56] and OPT [81] use a simple strategy that restarts the training process from an earlier checkpoint before the occurrence of the spike and skips over the data that may have caused the problem. Further, GLM [84] finds that the abnormal gradients of the embedding layer usually lead to spikes, and proposes to shrink the embedding layer gradients to alleviate it.

4.3.2 Scalable Training Techniques

As the model and data sizes increase, it has become challenging to efficiently train LLMs under a limited computational resource. Especially, two primary technical issues are required to be resolved, *i.e.,* increasing training through-

TABLE 5: Detailed optimization settings of several existing LLMs.

Model	Batch Size (#tokens)	Learning Rate	Warmup	Decay Method	Optimizer	Precision Type	Weight Decay	Grad Clip	Dropout
GPT3 (175B)	32K→3.2M	6×10^{-5}	yes	cosine decay to 10%	Adam	FP16	0.1	1.0	-
PanGu-α (200B)	-	2×10^{-5}	-	-	Adam	-	0.1	-	-
OPT (175B)	2M	1.2×10^{-4}	yes	manual decay	AdamW	FP16	0.1	-	0.1
PaLM (540B)	1M→4M	1×10^{-2}	no	inverse square root	Adafactor	BF16	lr^2	1.0	0.1
BLOOM (176B)	4M	6×10^{-5}	yes	cosine decay to 10%	Adam	BF16	0.1	1.0	0.0
MT-NLG (530B)	64 K→3.75M	5×10^{-5}	yes	cosine decay to 10%	Adam	BF16	0.1	1.0	-
Gopher (280B)	3M→6M	4×10^{-5}	yes	cosine decay to 10%	Adam	BF16	-	1.0	-
Chinchilla (70B)	1.5M→3M	1×10^{-4}	yes	cosine decay to 10%	AdamW	BF16	-	-	-
Galactica (120B)	2M	7×10^{-6}	yes	linear decay to 10%	AdamW	-	0.1	1.0	0.1
LaMDA (137B)	256K	-	-	-	-	BF16	-	-	-
Jurassic-1 (178B)	32 K→3.2M	6×10^{-5}	yes	-	-	-	-	-	-
LLaMA (65B)	4M	1.5×10^{-4}	yes	cosine decay to 10%	AdamW	-	0.1	1.0	-
LLaMA 2 (70B)	4M	1.5×10^{-4}	yes	cosine decay to 10%	AdamW	-	0.1	1.0	-
Falcon (40B)	2M	1.85×10^{-4}	yes	cosine decay to 10%	AdamW	BF16	0.1	-	-
GLM (130B)	0.4M→8.25M	8×10^{-5}	yes	cosine decay to 10%	AdamW	FP16	0.1	1.0	0.1
T5 (11B)	64K	1×10^{-2}	no	inverse square root	AdaFactor	-	-	-	0.1
ERNIE 3.0 Titan (260B)	-	1×10^{-4}	-	-	Adam	FP16	0.1	1.0	-
PanGu-Σ (1.085T)	0.5M	2×10^{-5}	yes	-	Adam	FP16	-	-	-

put and loading larger models into GPU memory. In this part, we review several widely used approaches in existing work to address the above two challenges, namely 3D parallelism [66, 253, 254], ZeRO [255], and mixed precision training [256], and also give general suggestions about how to utilize them for training.

3D Parallelism. 3D parallelism is actually a combination of three commonly used parallel training techniques, namely data parallelism, pipeline parallelism [253, 254], and tensor parallelism [66][21]. We next introduce the three parallel training techniques.

• *Data parallelism.* Data parallelism is one of the most fundamental approaches to improving the training throughput. It replicates the model parameters and optimizer states across multiple GPUs and then distributes the whole training corpus into these GPUs. In this way, each GPU only needs to process the assigned data for it, and performs the forward and backward propagation to obtain the gradients. The computed gradients on different GPUs will be further aggregated to obtain the gradients of the entire batch for updating the models in all GPUs. In this way, as the calculations of gradients are independently performed on different GPUs, the data parallelism mechanism is highly scalable, enabling the way that increases the number of GPUs to improve training throughput. Furthermore, this technique is simple in implementation, and most of existing popular deep learning libraries have already implemented data parallelism, such as TensorFlow and PyTorch.

• *Pipeline parallelism.* Pipeline parallelism aims to distribute the different layers of a LLM into multiple GPUs. Especially, in the case of a Transformer model, pipeline parallelism loads consecutive layers onto the same GPU, to reduce the cost of transmitting the computed hidden states or gradients between GPUs. However, a naive implementation of pipeline parallelism may result in a lower GPU utilization rate as each GPU has to wait for the previous one to complete the computation, leading to the unnecessary cost of *bubbles overhead* [253]. To reduce these bubbles in pipeline parallelism, GPipe [253] and PipeDream [254] propose the techniques of padding multiple batches of data and asynchronous gradient update to improve the pipeline efficiency.

• *Tensor parallelism.* Tensor parallelism is also a commonly used technique that aims to decompose the LLM for multi-GPU loading. Unlike pipeline parallelism, tensor parallelism focuses on decomposing the tensors (the parameter matrices) of LLMs. For a matrix multiplication operation $Y = XA$ in the LLM, the parameter matrix A can be split into two submatrices, A_1 and A_2, by column, which can be expressed as $Y = [XA_1, XA_2]$. By placing matrices A_1 and A_2 on different GPUs, the matrix multiplication operation would be invoked at two GPUs in parallel, and the final result can be obtained by combining the outputs from the two GPUs through across-GPU communication. Currently, tensor parallelism has been supported in several open-source libraries, *e.g.,* Megatron-LM [66], and can be extended to higher-dimensional tensors. Also, Colossal-AI has implemented tensor parallelism for higher-dimensional tensors [257–259] and proposed sequence parallelism [260] especially for sequence data, which can further decompose the attention operation of the Transformer model.

ZeRO. ZeRO [255] technique, proposed by the DeepSpeed [65] library, focuses on the issue of memory redundancy in data parallelism. As mentioned before, data parallelism requires each GPU to store the same copy of a LLM, including model parameters, model gradients, and optimizer parameters. Whereas, not all of the above data is necessary to be retained on each GPU, which would cause a memory redundancy problem. To resolve it, the ZeRO technique aims to retain only a fraction of data on each GPU, while the rest data can be retrieved from other GPUs when required. Specifically, ZeRO provides three solutions, depending on how the three parts of the data are stored, namely optimizer state partitioning, gradient partitioning, and parameter partitioning. Empirical results indicate that the first two solutions do not increase the communication overhead, and the third solution increases about 50% communication overhead but saves memory proportional to the number of GPUs. PyTorch has implemented a similar

21. Model parallelism is a more broader term that includes tensor parallelism and pipeline parallelism in some work [66].

technique as ZeRO, called FSDP [261].

Mixed Precision Training. In previous PLMs (*e.g.,* BERT [23]), 32-bit floating-point numbers, also known as FP32, have been predominantly used for pre-training. In recent years, to pre-train extremely large language models, some studies [256] have started to utilize 16-bit floating-point numbers (FP16), which reduces memory usage and communication overhead. Additionally, as popular NVIDIA GPUs (*e.g.,* A100) have twice the amount of FP16 computation units as FP32, the computational efficiency of FP16 can be further improved. However, existing work has found that FP16 may lead to the loss of computational accuracy [59, 69], which affects the final model performance. To alleviate it, an alternative called *Brain Floating Point (BF16)* has been used for training, which allocates more exponent bits and fewer significant bits than FP16. For pre-training, BF16 generally performs better than FP16 on representation accuracy [69].

Overall Training Suggestion. In practice, the above training techniques, especially 3D parallelism, are often jointly used to improve the training throughput and large model loading. For instance, researchers have incorporated 8-way data parallelism, 4-way tensor parallelism, and 12-way pipeline parallelism, enabling the training of BLOOM [69] on 384 A100 GPUs. Currently, open-source libraries like DeepSpeed [65], Colossal-AI [153], and Alpa [262] can well support the three parallel training methods. To reduce the memory redundancy, ZeRO, FSDP, and activation recomputation techniques [68, 263] can be also employed for training LLMs, which have already been integrated into DeepSpeed, PyTorch, and Megatron-LM. In addition, the mixed precision training technique such as BF16 can be also leveraged to improve the training efficiency and reduce GPU memory usage, while it requires necessary support on hardware (*e.g.,* A100 GPU). Because training large models is a time-intensive process, it would be useful to forecast the model performance and detect abnormal issues at an early stage. For this purpose, GPT-4 [46] has recently introduced a new mechanism called *predictable scaling* built on a deep learning stack, enabling the performance prediction of large models with a much smaller model, which might be quite useful for developing LLMs. In practice, one can further leverage the supporting training techniques of mainstream deep learning frameworks. For instance, PyTorch supports the data parallel training algorithm FSDP [261] (*i.e.,* fully sharded data parallel), which allows for partial offloading of training computations to CPUs if desired.

5 Adaptation of LLMs

After pre-training, LLMs can acquire the general abilities for solving various tasks. However, an increasing number of studies have shown that LLM's abilities can be further adapted according to specific goals. In this section, we introduce two major approaches to adapting pre-trained LLMs, namely instruction tuning and alignment tuning. The former approach mainly aims to enhance (or unlock) the abilities of LLMs, while the latter approach aims to align the behaviors of LLMs with human values or preferences. Further, we will also discuss efficient tuning and quantization for model adaptation in resource-limited settings. In what follows, we will introduce the four parts in detail.

TABLE 6: A detailed list of available collections for instruction tuning.

Categories	Collections	Time	#Examples
Task	Nat. Inst. [264]	Apr-2021	193K
	FLAN [62]	Sep-2021	4.4M
	P3 [265]	Oct-2021	12.1M
	Super Nat. Inst. [79]	Apr-2022	5M
	MVPCorpus [266]	Jun-2022	41M
	xP3 [85]	Nov-2022	81M
	OIG[22]	Mar-2023	43M
Chat	HH-RLHF [267]	Apr-2022	160K
	HC3 [268]	Jan-2023	87K
	ShareGPT[23]	Mar-2023	90K
	Dolly [24]	Apr-2023	15K
	OpenAssistant [269]	Apr-2023	161K
Synthetic	Self-Instruct [129]	Dec-2022	82K
	Alpaca [123]	Mar-2023	52K
	Guanaco [25]	Mar-2023	535K
	Baize [270]	Apr-2023	158K
	BELLE [271]	Apr-2023	1.5M

5.1 Instruction Tuning

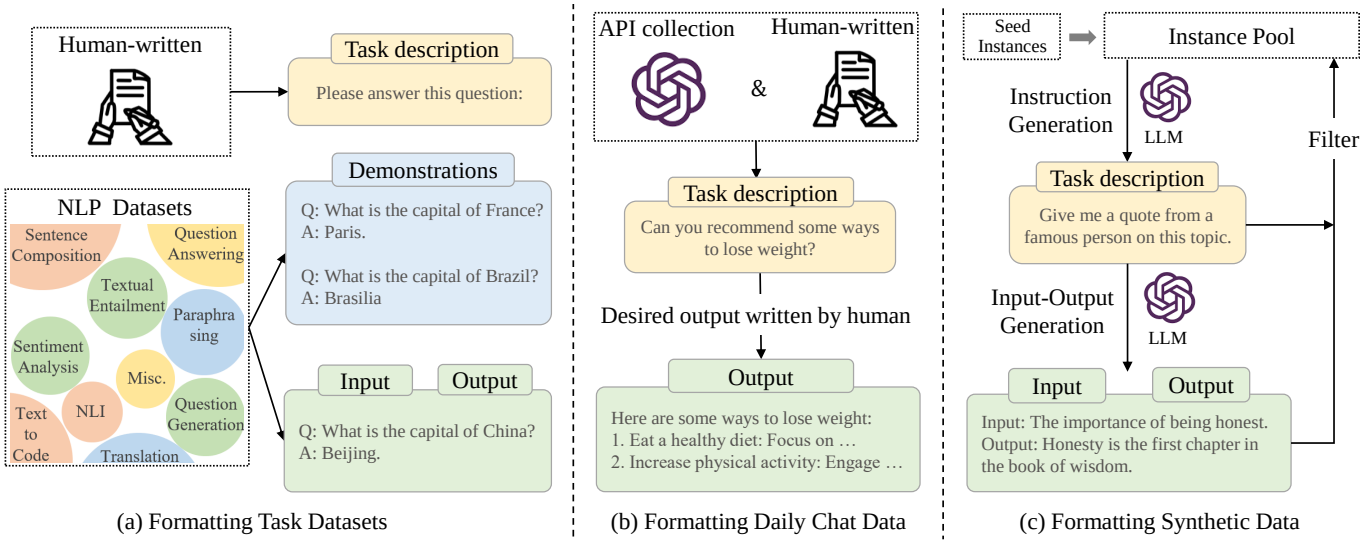
In essence, instruction tuning is the approach to fine-tuning pre-trained LLMs on a collection of formatted instances in the form of natural language [62], which is highly related to supervised fine-tuning [61] and multi-task prompted training [28]. In order to perform instruction tuning, we first need to collect or construct instruction-formatted instances. Then, we employ these formatted instances to fine-tune LLMs in a supervised learning way (*e.g.,* training with the sequence-to-sequence loss). After instruction tuning, LLMs can demonstrate superior abilities to generalize to unseen tasks [28, 62, 64], even in a multilingual setting [85].

A recent survey [272] presents a systematic overview of the research on instruction tuning. In comparison to that, we mainly focus on the effect of instruction tuning on LLMs and provide detailed guidelines or strategies for instance collection and tuning. In addition, we also discuss the use of instruction tuning for satisfying the real needs of users, which has been widely applied in existing LLMs, *e.g.,* InstructGPT [61] and GPT-4 [46].

5.1.1 Formatted Instance Construction

Generally, an instruction-formatted instance consists of a task description (called an *instruction*), an optional input, the corresponding output, and a small number of demonstrations (optional). As important public resources, existing studies have released a large number of labeled data formatted in natural language (see the list of available resources in Table 6). Next, we introduce three major methods for constructing formatted instances (see an illustration in Figure 9) and then discuss several key factors for instance construction.

22. https://laion.ai/blog/oig-dataset/
23. https://sharegpt.com/
24. https://github.com/databrickslabs/dolly
25. https://huggingface.co/datasets/JosephusCheung/GuanacoDataset



(a) Formatting Task Datasets (b) Formatting Daily Chat Data (c) Formatting Synthetic Data

Fig. 9: An illustration of instance formatting and three different methods for constructing the instruction-formatted instances.

Formatting Task Datasets. Before instruction tuning was proposed, several early studies [266, 273, 274] collected the instances from a diverse range of tasks (*e.g.*, text summarization, text classification, and translation) to create supervised multi-task training datasets. As a major source of instruction tuning instances, it is convenient to format these multi-task training datasets with natural language task descriptions. Specifically, recent work [28, 61, 62, 79] augments the labeled datasets with human-written task descriptions, which instructs LLMs to understand the tasks by explaining the task goal. For example, in Figure 9(a), a task description "*Please answer this question*" is added for each example in the question-answering task. After instruction tuning, LLMs can generalize well to other unseen tasks by following their task descriptions [28, 62, 64]. In particular, it has been shown that instructions are the crucial factor in task generalization ability for LLMs [62]: by fine-tuning the model on labeled datasets with the task descriptions removed, it results in a dramatic drop in model performance. To better generate labeled instances for instruction tuning, a crowd-sourcing platform, PromptSource [265] has been proposed to effectively create, share, and verify the task descriptions for different datasets. To enrich the training instances, several studies [28, 266, 275] also try to invert the input-output pairs of existing instances with specially designed task descriptions for instruction tuning. For instance, given a question-answer pair, we can create a new instance by predicting the answer-conditioned question (*e.g.*, *"Please generate a question based on the answer:"*).

Formatting Daily Chat Data. Despite that a large number of training instances have been formatted with instructions, they mainly come from public NLP datasets, either lacking instruction diversity or mismatching with real human needs [61]. To overcome this issue, InstructGPT [61] proposes to take the queries that real users have submitted to the OpenAI API as the task descriptions. User queries are expressed in natural languages, which are particularly suitable for eliciting the ability of instruction following for LLMs. Additionally, to enrich the task diversity, human labelers are also asked to compose the instructions for real-life tasks, including open-ended generation, open question answering, brainstorming, and chatting. Then, they let another group of labelers directly answer these instructions as the output. Finally, they pair one instruction (*i.e.*, the collected user query) and the expected output (*i.e.*, the human-written answer) as a training instance. Note that InstructGPT also employs these real-world tasks formatted in natural language for alignment tuning (discussed in Section 5.2). Further, GPT-4 [46] has designed potentially high-risk instructions and guided the model to reject these instructions through supervised fine-tuning for safety concerns. Recently, researchers also collect the users' chat requests as the input data and employ ChatGPT or GPT-4 to respond to these requests as the output data. A representative collection of such dataset is the conversational data from ShareGPT.

Formatting Synthetic Data. To reduce the burden of human annotation or manual collection, several semi-automated approaches [129] have been proposed for constructing instances by feeding existing instances into LLMs to synthesize diverse task descriptions and instances. As illustrated in Figure 9(c), the Self-Instruct method only needs around 100 instances as the initial task pool. Then, they randomly select a few instances from the pool as demonstrations and prompt a LLM to generate new instructions and corresponding input-output pairs. After the quality and diversity filtering, newly generated instances would be added into the task pool. Hence, the synthetic method is an effective and economical way to generate large-scale instruction data for LLMs.

Key Factors for Instance Construction. The quality of instruction instances has an important impact on the performance of the model. Here, we discuss some essential factors for instance construction.

- *Scaling the instructions.* It has been widely shown that

scaling the number of tasks can largely enhance the generalization ability of LLMs [28, 62, 79]. With the increasing of the task number, the model performance initially shows a continuous growth pattern, while the gain becomes negligible when it reaches a certain level [64, 79]. A plausible speculation is that a certain number of representative tasks can provide relatively sufficient knowledge and adding more tasks may not bring additional gains [64]. Also, it is beneficial to enhance the diversity of the task descriptions in several aspects, such as length, structure, and creativity [28]. As for the number of instances per task, it has been found that a small number of instances can usually saturate the generalization performance of the model [62, 64]. Whereas, increasing the number of instances for some tasks to a large number (*e.g.,* a few hundreds) could potentially result in the overfitting issue and impair the model performance [79, 276].

• *Formatting design.* As an important factor, the design of natural language format also highly impacts the generalization performance of LLMs [79]. Typically, we can add task descriptions and optional demonstrations to the input-output pairs of existing datasets, where the task description is the most key part for LLMs to understand the task [79]. Further, it can lead to substantial improvements by using an appropriate number of exemplars as demonstrations [64], which also alleviates the model sensitivity to instruction engineering [62, 64]. However, incorporating other components (*e.g.,* things to avoid, reasons, and suggestions) into instructions may have a negligible or even adverse effect on the performance of LLMs [79, 264]. Recently, to elicit the step-by-step reasoning ability of LLMs, some work [64] proposes to include chain-of-thought (CoT) examples for some reasoning datasets, such as arithmetic reasoning. It has been shown that fine-tuning LLMs with both CoT and non-CoT examples can lead to a good performance across various reasoning tasks, including those that require multi-hop reasoning ability (*e.g.,* commonsense question answering and arithmetic reasoning) as well as those without the need for such a reasoning way (*e.g.,* sentiment analysis and extractive question answering) [64, 86].

To summarize, diversity and quality of instructions seem to be more important than the number of instances [277] since the well-performing InstructGPT [61] and Alpaca [128] utilize fewer but more diverse instructions (or instances) than the Flan-series LLMs [62, 64]. Further, it is more useful to invite labelers to compose human-need tasks than using dataset-specific tasks. However, it still lacks general guidelines to annotate human-need instances, making the task composition somehow heuristic. To reduce human efforts, we can either reuse existing formatted datasets (Table 6) or automatically construct the instructions using existing LLMs [129]. We conduct a preliminary experiment to show the effectiveness of different construction methods in Section 5.1.4.

5.1.2 Instruction Tuning Strategies

Unlike pre-training, instruction tuning is often more efficient since only a moderate number of instances are used for training. Since instruction tuning can be considered as a supervised training process, its optimization is different from pre-training in several aspects [64], such as the training objective (*i.e.,* sequence-to-sequence loss) and optimization configuration (*e.g.,* smaller batch size and learning rate), which require special attention in practice. In addition to these optimization configurations, there are also four important aspects to consider for instruction tuning:

Balancing the Data Distribution. Since instruction tuning involves a mixture of different tasks, it is important to balance the proportion of different tasks during fine-tuning. A widely used method is the *examples-proportional mixing* strategy [73], *i.e.,* combining all the datasets and sampling each instance equally from the mixed datasets. Furthermore, increasing the sampling ratio of high-quality collections (*e.g.,* FLAN [62] and P3 [265]) can generally lead to performance improvement according to recent findings [64, 86]. Further, it is common to set a *maximum cap* to control the maximum number of examples that a dataset can contain during instruction tuning [73], which is set to prevent larger datasets from overwhelming the entire distribution [73, 86]. In practice, the maximum cap is typically set to several thousands or tens of thousands according to different datasets [62, 64].

Combining Instruction Tuning and Pre-Training. To make the tuning process more effective and stable, OPT-IML [86] incorporates pre-training data during instruction tuning, which can be regarded as regularization for model tuning. Further, instead of using a separate two-stage process (*pre-training* then *instruction tuning*), some studies attempt to train a model from scratch with a mixture of pre-training data (*i.e.,* plain texts) and instruction tuning data (*i.e.,* formatted datasets) using multi-task learning [73]. Specifically, GLM-130B [84] and Galactica [35] integrate instruction-formatted datasets as a small proportion of the pre-training corpora to pre-train LLMs, which potentially achieves the advantages of pre-training and instruction tuning at the same time.

Multi-stage Instruction Tuning. For instruction tuning, there are two kinds of important instruction data, namely task-formatted instructions and daily chat instructions. Generally, the former has a significantly larger volume than the latter. It is important to balance the training with the two kinds of instruction data. In addition to carefully mixing different instruction data, we can also adopt a multi-stage instruction tuning strategy[26], where LLMs are first fine-tuned with large-scale task-formatted instructions and subsequently fine-tuned on daily chat ones. To avoid the capacity forgetting issue, it is also useful to add an amount of task-formatted instructions at the second stage. Actually, such a multi-stage tuning strategy can be also applied to other settings for instruction tuning. For example, we can schedule different fine-tuning stages with progressively increased levels on difficulty and complexity, and gradually improve the capacities of LLMs to follow complex instructions.

Other Practical Tricks. In practice, there are also several useful strategies and tricks that are helpful to improve the fine-tuning performance of LLMs. We list several representative ones as follows:

26. https://github.com/RUC-GSAI/YuLan-Chat

TABLE 7: Basic statistics of the required number of GPUs, tuning time, batch size (denoted as BS) per device (full tuning and LoRA tuning), and inference rate (the number of generated tokes per second). Our experiments are conducted based on two Linux servers having 8 A800-80G SXM4 GPUs with 6 NVSwitch and 8 3090-24G GPUs, respectively. The major difference between A800 and A100 lies in the NVLink interconnect speed. Thus, our estimations about training and inference efficiency would be slightly improved for A100, while the rest memory consumption would remain the same. The full tuning experiments are conducted using data parallel training, ZeRO Stage 3, BF16, and gradient checkpointing. Additionally, the LoRA tuning can be executed on one 80G GPU utilizing INT8 quantization with the rank setting set to 16. The max sequence length for both training settings is set to 512. The inference experiments are performed with the batch size set to 1.

Models	**A800 Full Training**			**A800 LoRA Training**			**A800 Inference (16-bit)**		**3090 Inference (16-bit)**		**3090 Inference (8-bit)**	
	#GPU	BS	Time	#GPU	BS	Time	#GPU	#Token/s	#GPU	#Token/s	#GPU	#Token/s
LLaMA (7B)	2	8	3.0h	1	80	3.5h	1	36.6	1	24.3	1	7.5
LLaMA (13B)	4	8	3.1h	1	48	5.1h	1	26.8	2	9.9	1	4.5
LLaMA (30B)	8	4	6.1h	1	24	14.3h	1	17.7	4	3.8	2	2.6
LLaMA (65B)	16	2	11.2h	1	4	60.6h	2	8.8	8	2.0	4	1.5

• *Efficient training for multi-turn chat data.* Given a multi-turn chat example (the conversation between a user and chatbot), a straightforward fine-tuning way is to split it into multiple context-response pairs for training: a LLM is fine-tuned to generate the response based on the corresponding context for all splits (*i.e.,* at each utterance from the user). In such a fine-tuning way, it is apparent that there exist overlapping utterances in the split examples from a conversation. To save the training cost, Vicuna [124] has adopted an efficient way that feeds the whole conversation into the LLM, but relies on a loss mask that only computes the loss on the responses of the chatbot for training. It can significantly reduce the compute costs derived from the overlapped utterances.

• *Filtering low-quality instructions using LLMs.* After instruction data collection, it tends to contain low-quality ones, which might degrade the model performance and also increase the training cost. To address this issue, existing work [278] typically adopts powerful LLMs (*e.g.,* ChatGPT and GPT-4) to annotate a subset of instructions. It utilizes the prompts like "*determine its educational value for a student whose goal is to learn world knowledge*" to guide the LLM for annotating the quality of instructions, *e.g.,* high, middle and low. Then, these LLM-annotated instructions will be used to train a classifier to predict the quality of all the remaining instructions, and finally filter the ones that are predicted as low-quality ones.

• *Establishing self-identification for LLM.* To deploy LLMs for real-world applications, it is necessary to establish its identity and make LLMs aware of these identity information, such as name, developer and affiliation. A practical way is to create identity-related instructions for fine-tuning the LLM. It is also feasible to prefix the input with the self-identification prompt, *e.g.,* "*The following is a conversation between a human and an AI assistant called* CHATBOTNAME*, developed by* DEVELOPER.", where CHATBOTNAME and DEVELOPER refer to the name and developer of the chatbot, respectively.

In addition to the above practical strategies and tricks, existing work has also used other tricks, *e.g.,* concatenating multiple examples into a single sequence to approach the max length [279], using inference loss to evaluate the quality of instructions [280] and rewriting instructions into more complex ones [281].

5.1.3 The Effect of Instruction Tuning

In this part, we discuss the effect of instruction tuning on LLMs in three major aspects.

Performance Improvement. Despite being tuned on a moderate number of instances, instruction tuning has become an important way to improve or unlock the abilities of LLMs [64]. Recent studies have experimented with language models in multiple scales (ranging from 77M to 540B), showing that the models of different scales can all benefit from instruction tuning [64, 275], yielding improved performance as the parameter scale increases [85]. Further, smaller models with instruction tuning can even perform better than larger models without fine-tuning [28, 64]. Besides the model scale, instruction tuning demonstrates consistent improvements in various model architectures, pre-training objectives, and model adaptation methods [64]. In practice, instruction tuning offers a general approach to enhancing the abilities of existing language models [64] (including small-sized PLMs). Also, it is much less costly than pre-training, since the amount of instruction data required by LLMs is significantly smaller than pre-training data.

Task Generalization. Instruction tuning encourages the model to understand natural language instructions for task completion. It endows LLMs with the ability (often considered as an emergent ability) to follow human instructions [31] to perform specific tasks without demonstrations, even on unseen tasks [64]. A large number of studies have confirmed the effectiveness of instruction tuning to achieve superior performance on both seen and unseen tasks [86, 275]. Also, instruction tuning has been shown to be useful in alleviating several weaknesses of LLMs (*e.g.,* repetitive generation or complementing the input without accomplishing a certain task) [61, 64], leading to a superior capacity to solve real-world tasks for LLMs. Furthermore, LLMs trained with instruction tuning can generalize to related tasks across languages. For example, BLOOMZ-P3 [85] is fine-tuned based on BLOOM [69] using English-only task collection P3 [265]. Interestingly, BLOOMZ-P3 can achieve a more than 50% improvement in multilingual sentence completion tasks compared to BLOOM, which shows that instruction tuning can help LLMs acquire general task skills from English-only datasets and transfer such skills into

other languages [85]. In addition, it has been found that using English-only instructions can produce satisfactory results on multilingual tasks [85], which helps reduce the effort of instruction engineering for a specific language.

Domain Specialization. Existing LLMs have showcased superior capabilities in traditional NLP tasks (*e.g.,* generation and reasoning) and daily questions. However, they may still lack domain knowledge to accomplish specific tasks, such as medicine, law, and finance (See Section 9 for a detailed discussion of LLMs in different applications). Instruction tuning is an effective approach to adapting existing general LLMs to be domain-specific experts. For instance, researchers propose to fine-tune Flan-PaLM [64] using medical datasets to create Med-PaLM [282], a medical knowledge assistant that achieves performance levels comparable to those of expert clinicians. Furthermore, a recent study [283] fine-tunes FLAN-T5 to support e-commerce recommender systems with natural language instructions, showing strong performance in a variety of recommendation tasks. There are also several open-sourced medical models instruction-tuned based on LLaMA [57], such as BenTsao [284]. Also, researchers explore instruction tuning on law [285], finance [286], and arithmetic computation [287].

5.1.4 Empirical Analysis for Instruction Tuning

Fine-tuning LLMs with different instruction sets tend to lead to model variants with varied performance on downstream tasks. In this section, we will explore the effect of different types of instructions in fine-tuning LLMs (*i.e.,* LLaMA (7B) and LLaMA (13B)[27]), as well as examine the usefulness of several instruction improvement strategies.

Instruction Datasets. According to the discussion in Section 5.1.1, we mainly consider three common kinds of instructions as follows:

• *Task-specific instructions.* For the first type of instructions, we adopt the most commonly-used multi-task instruction dataset, *FLAN-T5* [64], which contains 1,836 tasks and over 15M instructions by combining four data mixtures from prior work.

• *Daily chat instructions.* This type of instructions are conversations posed by users about daily life, which are more closely related to real-life scenarios. We adopt the ShareGPT instruciton set[28], consisting of 63K real-user instructions. It has been used as the core instructions for Vicuna.

• *Synthetic instructions.* In addition to reusing existing instructions, we can also automatically synthesize massive instructions using LLMs. We adopt the popular synthetic instruction dataset Self-Instruct-52K [129], consisting of 52K instructions paired with about 82K instance inputs and outputs. These generated instructions have a similar data distribution as the human-written seed tasks (*e.g.,* grammar checking, brainstorming).

As the original FLAN-T5 dataset is very large (*i.e.,* over 15M), we randomly sample 80,000 instructions from it for conducting a fair comparison with other instruction datasets (*i.e.,* ShareGPT and Self-Instruct-52K) at a similar scale. In our experiments, we test on each individual instruction set to explore their own effects and also examine their combinatorial effects on model performance.

Improvement Strategies. Although real-world instructions from human users are more suitable for fine-tuning LLMs, it is difficult to collect them at a large scale. As alternatives to human-generated instructions, most existing research mainly adopts synthetic instructions generated by LLMs. However, there are some potential problems with synthetic instructions, such as poor topic diversity and uneven instruction difficulty (either too simple or too difficult). Thus, it is necessary to improve the quality of the synthetic instructions. Next, we summarize four major improvement strategies widely used in existing work as follows:

• *Enhancing the instruction complexity.* As discussed in existing work [281], enhancing the complexity of instructions can improve the model capacity of LLMs in following complex instructions, *e.g.,* including more task demands or requiring more reasoning steps. To validate this strategy, we follow WizardLM [281] by gradually increasing the complexity levels, *e.g.,* adding constraints, increasing reasoning steps, and complicating the input. We leverage the publicly released WizardLM-70K instructions[29] as the complexity-enhanced instruction dataset, which has been generated via the above enhancement approach based on the Self-Instruct-52K dataset [281].

• *Increasing the topic diversity.* In addition to the complexity, improving the topic diversity of the instruction dataset can help elicit different abilities of LLMs on diverse tasks in real world [288]. However, it is difficult to directly control the self-instruct process for generating diverse instructions. Following YuLan-Chat [289], we employ ChatGPT to rewrite the instructions from Self-Instruct-52K dataset for adapting them into 293 topics via specific prompts. Finally, we obtain 70K instructions as the diversity-increased dataset.

• *Scaling the instruction number.* In addition to the above aspects, the number of instructions is also an important factor that may affect the model performance. Specially, using more instructions can extend the task knowledge and improve the ability of instruction following for LLMs [64]. To examine this strategy, we sample new instructions from the synthesized instruction set released from the MOSS project[30], as they are also synthesized using the same self-instruct method [129]. We mix them with the Self-Instruct-52K dataset to compose a larger one containing 220K instructions.

• *Balancing the instruction difficulty.* As the synthetic instructions tend to contain too easy or too hard ones, it is likely to result in training instability or even overfitting for LLMs. To explore the potential effects, we leverage the perplexity score of LLMs to estimate the difficulty of instructions and remove too easy or too hard instructions. To generate the same scale of instructions for fair comparison, we adopt a LLaMA (7B) model to compute the perplexity for the 220K instructions from the large instruction dataset, and then keep 70K instructions of moderate perplexity scores as the difficulty-balanced dataset.

27. Due to the limit of computational resources, we cannot conduct large-scale experiments on larger LLaMA variants right now, which would be scheduled in a future version.

28. https://github.com/domeccleston/sharegpt

29. https://huggingface.co/datasets/victor123/evol_instruct_70k

30. https://github.com/OpenLMLab/MOSS

TABLE 8: Results of instruction-tuning experiments (all in a single-turn conversation) based on the LLaMA (7B) and LLaMA (13B) model under the chat and QA setting. We employ four instruction improvement strategies on the Self-Instruct-52K dataset, *i.e.,* enhancing the complexity (*w/ complexity*), increasing the diversity (*w/ diversity*), balancing the difficulty (*w/ difficulty*), and scaling the instruction number (*w/ scaling*). *Since we select the LLaMA (7B)/(13B) model fine-tuned on Self-Instruct-52K as the baseline, we omit the win rate of the fine-tuned model with Self-Instruct-52K against itself.

Models	Dataset Mixtures	Instruction Numbers	Lexical Diversity	Chat	QA	
				AlpacaFarm	MMLU	BBH3k
LLaMA (7B)	① FLAN-T5	80,000	48.48	23.77	38.58	32.79
	② ShareGPT	63,184	77.31	81.30	38.11	27.71
	③ Self-Instruct-52K	82,439	25.92	/*	37.52	29.81
	② + ③	145,623	48.22	71.36	41.26	28.36
	① + ② + ③	225,623	48.28	70.00	43.69	29.69
	③ Self-Instruct-52K	82,439	25.92	/*	37.52	29.81
	w/ complexity	70,000	70.43	76.96	39.73	33.25
	w/ diversity	70,000	75.59	81.55	38.01	30.03
	w/ difficulty	70,000	73.48	79.15	32.55	31.25
	w/ scaling	220,000	57.78	51.13	33.81	26.63
LLaMA (13B)	① FLAN-T5	80,000	48.48	22.12	34.12	34.05
	② ShareGPT	63,184	77.31	77.13	47.49	33.82
	③ Self-Instruct-52K	82,439	25.92	/*	36.73	25.43
	② + ③	145,623	48.22	72.85	41.16	29.49
	① + ② + ③	225,623	48.28	69.49	43.50	31.16
	③ Self-Instruct-52K	82,439	25.92	/*	36.73	25.43
	w/ complexity	70,000	70.43	77.94	46.89	35.75
	w/ diversity	70,000	75.59	78.92	44.97	36.40
	w/ difficulty	70,000	73.48	80.45	43.15	34.59
	w/ scaling	220,000	57.78	58.12	38.07	27.28

Experimental Setup. To conduct the experiments on the effect of instruction data, we leverage these new instruction datasets for tuning LLaMA, a popular LLM backbone that has been widely used for instruction-tuning. We use the code from YuLan-Chat [289] for our experiments, and train LLaMA (7B) and LLaMA (13B) on a server of 8 A800-80G GPUs. All the hyper-parameters settings remain the same as Stanford Alpaca. To better evaluate the instruction following ability of fine-tuned models, we consider two settings, namely *Chat setting* and *QA setting*. The chat setting mainly utilizes user instructions and queries from daily chat, whereas the QA setting mainly employs question answering examples from existing NLP datasets. The evaluation on the chat setting is conducted based on the AlpacaFarm evaluation set [290]. Instead of using a full pairwise comparison, we select the LLaMA (7B) and LLaMA (13B) models fine-tuned on Self-Instruct-52K as the reference baselines, and then compare them with other fine-tuned LLaMA (7B) and LLaMA (13B) models using different instructions, respectively. Since our focus is to examine the usefulness of different strategies to generate the instructions, the model fine-tuned on Self-Instruct-52K can serve as a good reference. Following AlpacaFarm [290], for each comparison, we employ ChatGPT to automatically annotate which response from two compared models each time is the best for the user query, and report the win rate (%) as the evaluation metric. For the QA setting, we select two benchmarks, MMLU [291] and BBH3k (a subset of BBH benchmark [292] released by YuLan-Chat), and evaluate the accuracy based on their default settings by using heuristic rules to parse the answers from these LLMs.

For both instruction tuning and evaluation, we adopt the following prompt: "*The following is a conversation between a human and an AI assistant. The AI assistant gives helpful, detailed, and polite answers to the user's questions.\n [|Human|]:{input}\n[|AI|]:*". To reproduce our results, we release the code and data at the link: https://github.com/RUCAIBox/LLMSurvey/tree/main/Experiments.

Results and Analysis. The results using different instruction datasets based on 7B and 13B LLaMA are in Table 8. Next, we summarize and analyze our findings in detail.

• *Task-formatted instructions are more proper for the QA setting, but may not be useful for the chat setting.* By comparing the performance of instruction tuning using FLAN-T5 with that of ShareGPT and Self-Instruct-52K, we can observe that FLAN-T5 mostly achieves a better performance on QA benchmarks while underperforms ShareGPT on the chat setting. The reason is that FLAN-T5 is composed of a mixture of instructions and examples from existing NLP tasks, *e.g.,* translation and reading comprehension. As a result, LLaMA fine-tuned with FLAN-T5 performs better on QA tasks, but poorly on user queries. In contrast, ShareGPT consists of real-world human-ChatGPT conversations, which is able to better elicit LLaMA to follow user instructions in daily life, while may not be suitable for accomplishing the QA tasks.

• *A mixture of different kinds of instructions are helpful to improve the comprehensive abilities of LLMs.* After mixing the three kinds of instructions for fine-tuning, we can see that the derived LLaMA variant (with FLAN-T5, ShareGPT and Self-Instruct-52K) performs well in both task settings. In MMLU, the performance of LLaMA (7B) can surpass the ones using individual instruction set by a large margin, *i.e.,* 43.69 vs. 38.58 (FLAN-T5). It shows that mixing multiple sources of instruction datasets is helpful to improve the

performance of instruction-tuned LLMs, which scales the instruction number as well as increases the diversity.

• *Enhancing the complexity and diversity of instructions leads to an improved model performance.* By increasing the complexity and diversity of the Self-Instruct-52K dataset respectively, the chat and QA performance of LLaMA can be consistently improved, *e.g.,* from 37.52 to 39.73 in MMLU for LLaMA (7B). It demonstrates that both strategies are useful to improve the instruction following ability of LLMs. Further, we can see that improving the complexity yields a larger performance improvement on QA tasks. The reason is that the QA tasks mostly consist of difficult questions for evaluating LLMs, which can be better solved by LLMs that have learned complex instructions at the fine-tuning stage.

• *Simply increasing the number of instructions may not be that useful, and balancing the difficulty is not always helpful.* As the results shown in Table 8, balancing the difficulty and increasing the number of fine-tuning instructions are not very helpful in our experiments. Especially for scaling the instruction number, it even hurts the performance, *e.g.,* a decrease from 29.81 to 26.63 in BBH3k for LLaMA (7B). It shows that simply scaling the number of synthesized instructions without quality control may not be effective to improve the performance. Furthermore, fine-tuning with the instructions of moderate difficulty also performs well in the chat setting, while slightly decreasing the performance in the QA setting. A possible reason is that we filter complex and hard instructions with large perplexity scores, hurting the model performance in answering complex questions.

• *A larger model scale leads to a better instruction following performance.* By comparing the performance of LLaMA (7B) and LLaMA (13B) models fine-tuned with the same set of instruction data, we can see that LLaMA (13B) mostly achieves a better performance. It indicates that scaling the model size is helpful for improving the instruction following capability. Besides, we can see that the QA performance has been improved a lot, *e.g.,* from 38.11 to 47.49 in MMLU. It is likely because that the larger models generally have better knowledge utilization and reasoning capability [33, 55], which can accurately answer more complex questions.

Instruction Tuning Suggestions

To conduct instruction tuning on LLMs, one can prepare the computational resources according to the basic statistics about the required number of GPUs and tuning time in Table 7. After setting up the development environment, we recommend beginners to follow the code of Alpaca repository[a] for instruction tuning. Subsequently, one should select the base model and construct the instruction datasets as we discuss in this section. When computational resources for training are constrained, users can utilize LoRA for parameter-efficient tuning (see Section 5.3). As for inference, users can further use quantization methods to deploy LLMs on fewer or smaller GPUs (see Section 5.4).

a. https://github.com/tatsu-lab/stanford_alpaca/#fine-tuning

5.2 Alignment Tuning

This part first presents the background of alignment with its definition and criteria, then focuses on the collection of human feedback data for aligning LLMs, and finally discusses the key technique of reinforcement learning from human feedback (RLHF) for alignment tuning.

5.2.1 Background and Criteria for Alignment

Background. LLMs have shown remarkable capabilities in a wide range of NLP tasks [55, 56, 62, 81]. However, these models may sometimes exhibit unintended behaviors, *e.g.,* fabricating false information, pursuing inaccurate objectives, and producing harmful, misleading, and biased expressions [61, 293]. For LLMs, the language modeling objective pre-trains the model parameters by word prediction while lacking the consideration of human values or preferences. To avert these unexpected behaviors, human alignment has been proposed to make LLMs act in line with human expectations [61, 294]. However, unlike the original pre-training and adaptation tuning (*e.g.,* instruction tuning), such an alignment requires considering very different criteria (*e.g.,* helpfulness, honesty, and harmlessness). It has been shown that alignment might harm the general abilities of LLMs to some extent, which is called *alignment tax* in related literature [295].

Alignment Criteria. Recently, there is increasing attention on developing multifarious criteria to regulate the behaviors of LLMs. Here, we take three representative alignment criteria (*i.e.,* helpful, honest, and harmless) as examples for discussion, which have been widely adopted in existing literature [61, 295]. In addition, there are other alignment criteria for LLMs from different perspectives including behavior, intent, incentive, and inner aspects [293], which are essentially similar (or at least with similar alignment techniques) to the above three criteria. It is also feasible to modify the three criteria according to specific needs, *e.g.,* substituting honesty with correctness [103]. Next, we give brief explanations about the three representative alignment criteria:

• *Helpfulness.* To be helpful, the LLM should demonstrate a clear attempt to assist users in solving their tasks or answering questions in a concise and efficient manner as possible. At a higher level, when further clarification is needed, the LLM should demonstrate the capability of eliciting additional relevant information through pertinent inquiries and exhibit suitable levels of sensitivity, perceptiveness, and prudence [295]. Realizing the alignment of helpful behavior is challenging for LLMs since it is difficult to precisely define and measure the intention of users [293].

• *Honesty.* At a basic level, a LLM aligned to be honest should present accurate content to users instead of fabricating information. Additionally, it is crucial for the LLM to convey appropriate degrees of uncertainty in its output, in order to avoid any form of deception or misrepresentation of information. This requires the model to know about its capabilities and levels of knowledge (*e.g.,* "know unknowns"). According to the discussion in [295], honesty is a more objective criterion compared to helpfulness and harmlessness, hence honesty alignment could potentially be developed with less reliance on human efforts.

• *Harmlessness.* To be harmless, it requires that the language produced by the model should not be offensive or discriminatory. To the best of its abilities, the model should be capable of detecting covert endeavors aimed at soliciting requests for malicious purposes. Ideally, when the model was induced to conduct a dangerous action (*e.g.,* committing a crime), the LLM should politely refuse. Nonetheless, *what behaviors* are deemed harmful and *to what extent* vary amongst individuals or societies [295] highly depend on who is using the LLM, the type of the posed question, and the context (*e.g.,* time) at which the LLM is being used.

As we can see, these criteria are quite subjective, and are developed based on human cognition. Thus, it is difficult to directly formulate them as optimization objectives for LLMs. In existing work, there are many ways to fulfill these criteria when aligning LLMs. A promising technique is *red teaming* [296], which involves using manual or automated means to probe LLMs in an adversarial way to generate harmful outputs and then updates LLMs to prevent such outputs.

5.2.2 Collecting Human Feedback

During the pre-training stage, LLMs are trained using the language modeling objective on a large-scale corpus. However, it cannot take into account the subjective and qualitative evaluations of LLM outputs by humans (called *human feedback* in this survey). High-quality human feedback is extremely important for aligning LLMs with human preferences and values. In this part, we discuss how to select a team of human labelers for feedback data collection.

Human Labeler Selection. In existing work, the dominant method for generating human feedback data is human annotation [61, 103, 294]. This highlights the critical role of selecting appropriate human labelers. To provide high-quality feedback, human labelers are supposed to have a qualified level of education and excellent proficiency in English. For example, Sparrow [103] requires human labelers to be UK-based native English speakers who have obtained at least an undergraduate-level educational qualification. Even then, several studies [294] have found that there still exists a mismatch between the intentions of researchers and human labelers, which may lead to low-quality human feedback and cause LLMs to produce unexpected output. To address this issue, InstructGPT [61] further conducts a screening process to filter labelers by assessing the agreement between human labelers and researchers. Specifically, researchers first label a small amount of data and then measure the agreement between themselves and human labelers. The labelers with the highest agreement will be selected to proceed with the subsequent annotation work. In some other work [297], "super raters" are used to ensure the high quality of human feedback. Researchers evaluate the performance of human labelers and select a group of well-performing human labelers (*e.g.,* high agreement) as super raters. The super raters will be given priority to collaborate with the researchers in the subsequent study. When human labelers annotate the output of LLMs, it is helpful to specify detailed instructions and provide instant guidance for human labelers, which can further regulate the annotation of labelers.

Human Feedback Collection. In existing work, there are mainly three kinds of approaches to collecting feedback and preference data from human labelers.

• *Ranking-based approach.* In early work [294], human labelers often evaluate model-generated outputs in a coarse-grained manner (*i.e.,* only selecting the best) without taking into account more fine-grained alignment criteria. Nonetheless, different labelers may hold diverse opinions on the selection of the best candidate output, and this method disregards the unselected samples, which may lead to inaccurate or incomplete human feedback. To address this issue, subsequent studies [103] introduce the Elo rating system to derive the preference ranking by comparing candidate outputs. The ranking of outputs serves as the training signal that guides the model to prefer certain outputs over others, thus inducing outputs that are more reliable and safer.

• *Question-based approach.* Further, human labelers can provide more detailed feedback by answering certain questions designed by researchers [72], covering the alignment criteria as well as additional constraints for LLMs. Specially, in WebGPT [72], to assist the model in filtering and utilizing relevant information from retrieved documents, human labelers are required to answer questions with multiple options about whether the retrieved documents are useful for answering the given input.

• *Rule-based approach.* Many studies also develop rule-based methods to provide more detailed human feedback. As a typical case, Sparrow [103] not only selects the response that labelers consider the best but also uses a series of rules to test whether model-generated responses meet the alignment criteria of being helpful, correct, and harmless. In this way, two kinds of human feedback data can be obtained: (1) the response preference feedback is obtained by comparing the quality of model-generated output in pairs, and (2) the rule violation feedback is obtained by collecting the assessment from human labelers (*i.e.,* a score indicating to what extent the generated output has violated the rules). Furthermore, GPT-4 [46] utilizes a set of zero-shot classifiers (based on GPT-4 itself) as rule-based reward models, which can automatically determine whether the model-generated outputs violate a set of human-written rules.

In the following, we focus on a well-known technique, reinforcement learning from human feedback (RLHF), which has been widely used in the recent powerful LLMs such as ChatGPT. As discussed below, the alignment criteria introduced in Section 5.2.1 can be fulfilled by learning from human feedback on the responses of LLMs to users' queries.

5.2.3 Reinforcement Learning from Human Feedback

To align LLMs with human values, reinforcement learning from human feedback (RLHF) [70, 294] has been proposed to fine-tune LLMs with the collected human feedback data, which is useful to improve the alignment criteria (*e.g.,* helpfulness, honesty, and harmlessness). RLHF employs reinforcement learning (RL) algorithms (*e.g.,* Proximal Policy Optimization (PPO) [115]) to adapt LLMs to human feedback by learning a reward model. Such an approach incorporates humans in the training loop for developing well-aligned LLMs, as exemplified by InstructGPT [61].

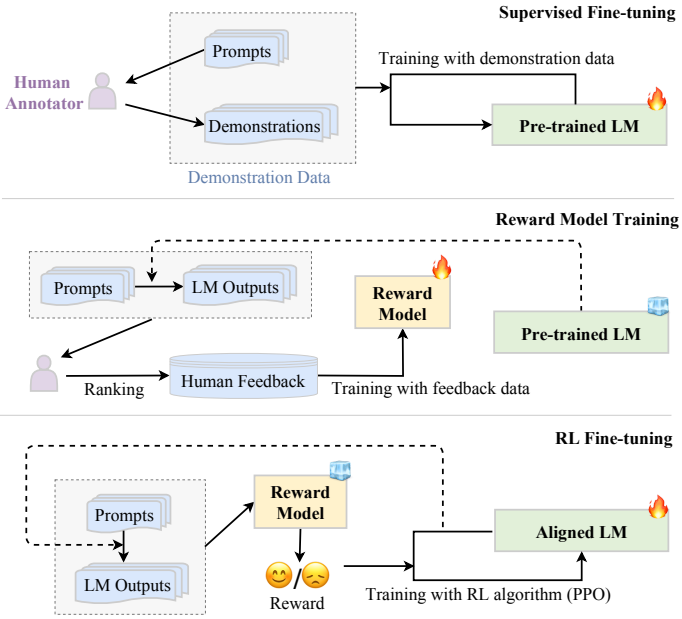


Fig. 10: The workflow of the RLHF algorithm.

RLHF System. The RLHF system mainly comprises three key components: a pre-trained LM to be aligned, a reward model learning from human feedback, and a RL algorithm training the LM. Specifically, the *pre-trained LM* is typically a generative model that is initialized with existing pre-trained LM parameters. For example, OpenAI uses 175B GPT-3 for its first popular RLHF model, InstructGPT [61], and DeepMind uses the 280 billion parameter model Gopher [59] for its GopherCite model [297]. Further, the *reward model (RM)* provides (learned) guidance signals that reflect human preferences for the text generated by the LM, usually in the form of a scalar value. The reward model can take on two forms: a fine-tuned LM or a LM trained de novo using human preference data. Existing work typically employs reward models having a parameter scale different from that of the aligned LM [61, 297]. For example, OpenAI uses 6B GPT-3 and DeepMind uses 7B Gopher as the reward model, respectively. Finally, to optimize the pre-trained LM using the signal from the reward model, a specific *RL algorithm* is designed for large-scale model tuning. Specifically, Proximal Policy Optimization (PPO) [115] is a widely used RL algorithm for alignment in existing work [61, 103, 297].

Key Steps for RLHF. Figure 10 illustrates the overall three-step process of RLHF [61] as introduced below.

• *Supervised fine-tuning.* To make the LM initially perform desired behaviors, it usually needs to collect a supervised dataset containing input prompts (instruction) and desired outputs for fine-tuning the LM. These prompts and outputs can be written by human labelers for some specific tasks while ensuring the diversity of tasks. For example, InstructGPT [61] asks human labelers to compose prompts (*e.g.*, "*List five ideas for how to regain enthusiasm for my career*") and desired outputs for several generative tasks such as open QA, brainstorming, chatting, and rewriting. Note that the first step is optional in specific settings or scenarios.

• *Reward model training.* The second step is to train the RM using human feedback data. Specifically, we employ the LM to generate a certain number of output texts using sampled prompts (from either the supervised dataset or the human-generated prompt) as input. We then invite human labelers to annotate the preference for these pairs. The annotation process can be conducted in multiple forms, and a common approach is to annotate by ranking the generated candidate texts, which can reduce the inconsistency among annotators. Then, the RM is trained to predict the human-preferred output. In InstructGPT, labelers rank model-generated outputs from best to worst, and the RM (*i.e.*, 6B GPT-3) is trained to predict the ranking. Note that, in recent work [298], the annotation of preference on response pairs has been conducted by an AI agent (usually an aligned LLM) instead of humans, which is called "*reinforcement learning from AI feedback (RLAIF)*".

• *RL fine-tuning.* At this step, aligning (*i.e.*, fine-tuning) the LM is formalized as an RL problem. In this setting, the pre-trained LM acts as the policy that takes as input a prompt and returns an output text, the action space of it is the vocabulary, the state is the currently generated token sequence, and the reward is provided by the RM. To avoid eviating significantly from the initial (before tuning) LM, a penalty term is commonly incorporated into the reward function. For example, InstructGPT optimizes the LM against the RM using the PPO algorithm. For each input prompt, InstructGPT calculates the KL divergence between the generated results from the current LM and the initial LM as the penalty. It is noted that the second and final steps can be iterated in multiple turns for better aligning LLMs. Due to the instability of the RL algorithm, recent work [299] replaces the RL tuning with another supervised fine-tuning by reusing the best ranked samples with higher rewards.

Practical Strategies for RLHF. Although RLHF is promising to effectively improve the alignment of LLMs with humans, it is practically challenging for researchers to successfully implement it. In this part, we focus on discussing several useful strategies and tricks for improving the effectiveness and efficiency of RLHF. Concretely, we focus on the effective training of reward models, efficient and effective RL training, respectively.

• *Effective reward model training.* Despite that InstructGPT used a small reward model (6B GPT model), increasing work [90] has shown it is often more effective to use a large reward model (*e.g.*, equal or greater than the original model size), since large reward models generally perform better in judging the quality of the LLM generated outputs. In LLaMa 2 [90], pretrained chat model checkpoints are used to initialize the reward model, they argue that such an approach can effectively reduce the information mismatch between the model to be aligned and the reward model by sharing the same pre-training knowledge. Whereas, it is common to encounter the overfitting problem when training large-scale reward models. As a simple yet effective solution, existing work [300, 301] has introduced the LM loss on the preferred response of the input prompt from the human-annotated alignment dataset as a regularizer, which alleviates the overfitting of the reward model on the binary classification task. In addition, as there are multiple criteria for alignment (*e.g.*, helpfulness and honesty), it is

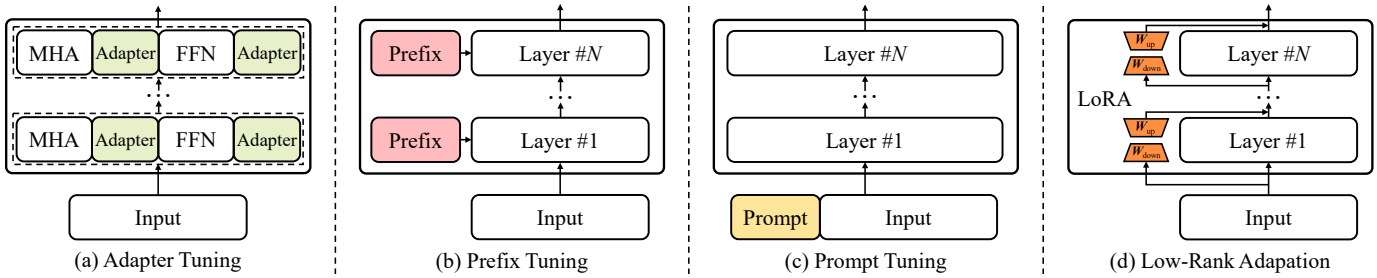


Fig. 11: An illustration of four different parameter-efficient fine-tuning methods. MHA and FFN denote the multi-head attention and feed-forward networks in the Transformer layer, respectively.

often difficult to train a single reward model that can satisfy all the alignment criteria. Therefore, it is useful to train multiple reward models that focus on different alignment criteria [90], and compute the final reward based on the produced ones from them via special combination strategies (*e.g.,* mean pooling and weighted sum). Such a way enables more flexible rules or standards on multiple criteria, *e.g.,* relaxing the requirement on helpfulness while posing more strict limits on harmfulness.

• *Effective RL training.* As the RL training process tends to be unstable and hyper-parameter sensitive, it is suggested that the language model should be well supervised fine-tuned before RL training, so as to reaching a good model capacity. A commonly-used way is to fine-tune the LLM on its best outputs of the prompts (referred to as *rejection sampling* or *best-of-N*) from the alignment dataset until convergence before RL. Given a prompt, the LLM would first produce N outputs via the sampling algorithm, and then the best candidate from the model will be selected by the reward model for learning. After fine-tuning the LLM on the best samples until convergence, the RL process will be performed to further improve the performance. LLaMA 2 [90] has successively trained five versions of RLHF models, where the LLM has been progressively improved with the improvement of the reward models. In this way, the collected prompts and annotations of human preference data can better reflect the issues of the current model checkpoint, thus making special tuning to address these issues. In addition, LLaMA 2 also adds samples from prior iterations into the subsequent ones, to alleviate the possible capacity regression issue during iterative optimization.

• *Efficient RL training.* As the RL training requires to iterate the inference process of both the LLM and reward models, it would greatly increase the total memory and computation cost, especially for larger reward models and LLMs. As a practical trick, we can deploy the reward model on a separate server, and invoke the corresponding API to work with the LLM on its own server. In addition, as RLHF requires the LLM to generate multiple candidate outputs, instead of calling the sample decoding procedure for multiple times, it is more efficient to utilize the beam search decoding algorithm[31]. It only needs to perform one-pass decoding for response generation, meanwhile such a strategy can also enhance the diversity of the generated candidate responses.

31. https://huggingface.co/docs/transformers/v4.31.0/en/main_classes/text_generation#transformers.GenerationMixin.group_beam_search

5.2.4 Alignment without RLHF

Although RLHF has achieved great success in aligning the behaviors of LLMs with human values and preferences, it also suffers from notable limitations. First, RLHF needs to train multiple LMs including the model being aligned, the reward model, and the reference model at the same time, which is tedious in algorithmic procedure and memory-consuming in practice. Besides, the commonly-used PPO algorithm in RLHF is rather complex and often sensitive to hyper-parameters. As an alternative, increasing studies explore to directly optimize LLMs to adhere to human preferences, using supervised fine-tuning without reinforcement learning.

Overview. The basic idea of non-RL alignment approaches is to directly fine-tune LLMs with *supervised learning* on high-quality *alignment dataset*. It basically assumes that response feedback or golden rules to avert unsafe behaviors have been injected or included in the specially curated alignment dataset, so that LLMs can directly learn aligned behaviors from these demonstration data via suitable fine-tuning strategies. Thus, to implement this approach, two key issues are the construction of alignment dataset and the design of fine-tuning loss. For the first issue, the alignment dataset can be automatically constructed by an aligned LLMs according to human-written safety principles [288] or refining existing examples using edits operations [302]. In addition, we can also reuse existing reward models to select high-rated responses from existing human feedback data [299]. For the second issue, non-RL alignment approaches mainly fine-tune LLMs in a supervised learning way (the same as the original instruction tuning loss) on a high-quality alignment dataset, meanwhile auxiliary learning objectives can be used to enhance the alignment performance, *e.g.,* ranking responses or contrasting instruction-response pairs.

Alignment Data Collection. The construction of alignment data is important to effectively align the behaviors of LLMs with human preferences. To collect high-quality alignment data, some work tries to reuse existing reward models to select high-rated responses, and others explore to leverage powerful LLMs (*e.g.,* ChatGPT) or build a simulated environment to generate synthetic alignment examples. Next, we will discuss these three lines of research.

• *Reward model based approaches.* The reward model in RLHF has been trained to measure the alignment degree

on the responses of LLMs. It is straightforward to leverage existing reward models to select high-quality responses as alignment data for subsequent fine-tuning. Based on this idea, RAFT [299] adopts reward models trained on human preference data to rank the responses of LLMs and collect those with higher rewards for supervised fine-tuning. In addition, the reward model can be also used to score model responses and assign them into different quality groups. Quark [303] sorts the responses of LLMs into different quantiles based on the reward scores. Each quantile is attached with a special reward token to represent the reward level of the quantile. Conditioned on the highest-reward tokens, LLMs are subsequently prompted to generate high-quality responses. As valuable resources for aligning LLMs, several reward models have been released, including DeBERTa-base/large/xxlarge from OpenAssistant[32], Moss-7B from Fudan[33], and Flan-T5-xl from Stanford[34].

• *LLM based generative approaches.* Reward models help to select aligned data from model responses. However, training reward models itself necessitates substantial high-quality human-labeled data, which is typically expensive and in short supply. In addition, although existing reward models can be reused, they might not be able to accurately capture the nonalignment behaviors in another separately trained LLM. Therefore, some work explores leveraging powerful LLMs to automatically generate human-aligned data. As a representative work, constitutional AI [298] proposes that human supervision comes from a set of principles (*i.e.,* natural language instructions) governing AI behaviors. Based on these principles, LLMs will critique their own harmful responses and revise them repeatedly into finally aligned responses. Similarly, Self-Align [288] first adopts self-instruct [129] to generate instructions focusing on covering diverse topics. Then, the model is also prompted with multiple human-written principles that describe the rules of expected model behaviors (also with several in-context exemplars), to generate helpful, ethical, and reliable responses as alignment data.

• *LLM based interactive approaches.* Most existing approaches train LLMs in isolation, where LLMs are not present in actual environments to improve themselves through external feedback signals. As a comparison, humans learn social norms and values from interactions with others in social environments [304]. To mimic such a learning approach, Stable Alignment [305] builds a simulated interaction environment consisting of a number of LLM agents, where AI agents keep interacting with and each other, receiving feedback on improvement. Once a central agent receives an instruction, it produces a response and shares it with nearby agents. These critic agents generate feedback comprising ratings about the response and revision suggestions. Then the central agent would revise the original response following these suggestions. Such an alignment approach can be also extended to real-world environment with humans.

Supervised Alignment Tuning. After obtaining alignment data, it is also key to design suitable fine-tuning strategies for direct alignment. A straightforward approach is to optimize LLMs using the conventional sequence-to-sequence objective based on the alignment data. In addition to the conventional optimization objective, several studies further explore auxiliary losses that enhance the learning from the alignment data.

• *Primary training objective.* Since the alignment data typically consists of an input instruction and an output response, the primary training loss is still the traditional cross-entropy loss for sequence-to-sequence learning. Based on this loss, many studies propose a number of improvement variants for enhancing the supervised alignment tuning. For example, CoH [306] constructs the training data by prepending "*A helpful answer:*" and "*An unhelpful answer:*" to the annotated good and bad responses, respectively, and only compute losses for those response tokens with special masking. Quark [303] sorts model responses into different quantiles with varying alignment quality, it prepends a special reward token to each model response to represent the reward level of the response. Further, to enable the preference modeling via the maximum likelihood objective, DPO [307] first reparameterizes the response rewards using the policy model (*i.e.,* the language model being optimized), and then the original reward modelling objective can be reformulated only based on the policy model. In this way, DPO removes the explicit reward modeling step, and optimizing the new learning objective only involving the policy model is equivalent to optimizing the rewards.

• *Auxiliary optimization objectives.* Besides the primary cross-entropy loss, several studies propose auxiliary training loss to enhance the learning from the alignment data. First, since the responses of each instruction can be scored by the reward model, the ranking loss can be used to train the model to preserve the ranking order of these responses. For example, the study [308] samples responses from multiple sources, including model-generated responses, such as those derived from the model itself, ChatGPT, and GPT-4, as well as human-written responses, spanning both high-quality and low-quality instances. To align with the scores from reward models, it further optimizes the ranking loss by encouraging the model to have higher conditional log probability for the response with a higher ranking. Second, to enhance the relatedness between the response and the instruction, some work adopts contrastive learning to push up the probability of correct instruction-response pairs while push down incorrect instruction-response pairs. Specially, for an output response, the proposed approach in [309] contrast the target instruction to the other irrelevant instructions. By doing so, it can enable the model to learn the right correlation between instructions and responses.

5.2.5 Remarks on SFT and RLHF

As discussed in Section 5.1, instruction tuning is the process of training pre-trained language models with formatted demonstration data (instructions paired with desired outputs). At early exploration, instruction data was mainly collected from NLP tasks [62], while it has been now extended to more diverse supervision data that pairs input and output texts (*e.g.,* the utterances of open-ended dialogues). Training with such paired texts is also called *supervised fine-tuning (SFT)* in the context of LLMs [61]. In this part, we

32. https://huggingface.co/OpenAssistant
33. https://github.com/OpenLMLab/MOSS-RLHF
34. https://huggingface.co/stanfordnlp/SteamSHP-flan-t5-xl

mainly use the abbreviation *SFT* for discussion but not instruction tuning, due to the simplicity and popularity.

Since SFT and RLHF are two major adaptation tuning methods for LLMs, it is important to understand the connections and difference between them. Next, we make some discussions on this issue[35].

Overall Comparison with RL Formulation. Following the discussion in Section 5.2.3 (the part related to RL training), the text generation problem can be formulated as a decision-making process based on RL. Taking a prompt as input, the task of a LLM is to generate a text completion that appropriately responds to the prompt. This task would be completed step by step. At each step, an agent (*i.e.,* LLM) will perform an action (*i.e.,* generating a token) according to the policy (*i.e.,* the generative probability distribution of LLM) conditioned on the current state (currently generated token sequence and other available context information). It is expected that a high-quality output text would be produced by the LLM, which can earn a large reward score based on the entire response. Overall, RLHF and SFT can be considered as two different training approaches to optimizing the above decision making process for LLMs. Specially, RLHF firstly learns the reward model, and then employs it to improve the LLM with RL training (*e.g.,* PPO). As a comparison, SFT adopts a teacher-forcing approach, which directly optimizes the likelihood of a demonstration output. Such a token-level training way essentially does *behavior cloning* (a special algorithm of imitation learning [310]): it utilizes the expert's action (*i.e.,* the target token at each step) as the supervision label and directly learns to imitate the demonstrations from experts without specifying a reward model as in typical RL algorithms. To learn the desired policies, SFT adopts a "local" optimization way (*i.e.,* token-level loss) based on demonstration data, while RLHF takes a "global" optimization way (*i.e.,* text-level loss) by involving human preference. More theoretical analysis about imitation learning and reinforcement learning can be referred to the related RL literature [310, 311].

Pros and Cons of SFT. SFT has been shown to be an effective approach to boosting the performance of LLMs on various benchmarks [62, 64, 123, 124], which can largely enhance the task generalization ability and flexibly endow specific functions (*e.g.,* establishing the chatbot's identity). More discussions about the usefulness of SFT can be found in Section 5.1.3. It has been widely recognized that SFT mainly *unlocks* the abilities but not *inject* new abilities into LLMs. Thus, it might become problematic when one tries to stimulate the non-endogenous abilities of LLMs via SFT. As a concrete scenario, it would potentially advocate the hallucination behaviors when demonstration data is beyond the knowledge or ability scope of LLMs, *e.g.,* training a LLM to answer questions about its unknown facts. An interesting viewpoint from John Schulman's talk on RLHF [312] is that distilling superior models to train less capable models (*e.g.,* prompting GPT-4 to generate the response as fine-tuning data) might increase the possibilities of generating the hallucinated texts, thus likely affecting the factual accuracy of LLMs. Furthermore, as a behavior cloning method, SFT aims to imitate the behaviors (without explorations) of the experts who construct the demonstration data. However, there often exist variations among different annotators on the writing styles, quality, and preferences of demonstration data, which tends to affect the learning performance of SFT. Thus, high-quality instruction data (but not the quantity) is the primary factor for effective training of LLMs during the SFT stage [90].

Pros and Cons of RLHF. RLHF was early explored in the literature of deep RL [70], then borrowed to improve the capacity of language models (*e.g.,* summarization [116]), and subsequently adopted as the fundamental technique to develop InstructGPT [61]. Recently, increasing evidence [90, 298] has demonstrated the effectiveness of RLHF in mitigating the harmful responses and enhancing the model capacity. Specially, LLaMA 2 has demonstrated that RLHF can improve both the helpfulness and harmlessness scores [90], and attributed this to a better human-LLM synergy for data annotation. They explain this reason in two major aspects as follows. First, since human annotators mainly provide preference annotations for RLHF, it can largely alleviate the discrepancies of annotators as that in SFT. Secondly, preference annotation is much easier than writing the demonstration data, and annotators can even judge the quality of more superior generations than those they create, making it possible to explore a broader state space beyond what can be demonstrated by human annotators. Another key point is that RLHF essentially encourages LLMs to learn correct policies by contrasting the self-generated responses (discriminating between good and bad responses). It no longer forces the model to imitate external demonstration data, and thus can mitigate the hallucination issues with SFT as discussed above[36]. Actually, RLHF has been demonstrated to be an important approach to reduce the hallucination behaviors in GPT-4 [46]. However, RLHF inherits the drawbacks of classic RL algorithms, *e.g.,* sample inefficiency and training instability. When adapted to LLMs, RLHF further relies on a strong SFT model as initial model checkpoint for efficiently achieving good performance. In addition, human annotators are involved in a complex iterative optimization process, in which a number of important details (*e.g.,* the prompt selection, the schedule of reward model training and PPO training, and the settings of hyper-parameters) have important impact on the whole model performance.

Overall, SFT is particularly useful to increase the model capacity of pre-trained model checkpoints right after pre-training, while RLHF is promising to further improve the model capacity of SFT models. However, RLHF has been difficult to implement, and far from well explored (according to public literature), and more improvements (*e.g.,* efficient and reliable annotation [298] and simplified optimization [307]) are still needed for further research.

35. This part would be somehow subjective, mainly based on the authors' opinions and experiences. Comments or corrections are welcome to enhance this part.

36. In RLHF, it seems to be also important that reward models should be aware of the knowledge or ability of a LLM to be aligned. For example, LLaMA 2 adopts pre-trained chat model checkpoints to initialize reward models [90].

5.3 Parameter-Efficient Model Adaptation

In the above, we have discussed the approaches of instruction tuning and alignment tuning to adapt LLMs according to specific goals. Since LLMs consist of a huge amount of model parameters, it would be costly to perform the full-parameter tuning. In this section, we will discuss how to conduct efficient tuning on LLMs. We first review several representative parameter-efficient fine-tuning methods for Transformer language models, and then summarize existing work on parameter-efficient fine-tuned LLMs.

5.3.1 Parameter-Efficient Fine-Tuning Methods

In existing literature, parameter-efficient fine-tuning [131, 313, 314] has been an important topic that aims to reduce the number of trainable parameters while retaining a good performance as possible. In what follows, we briefly review four parameter-efficient fine-tuning methods for Transformer language models, including adapter tuning, prefix tuning, prompt tuning and LoRA. The illustration of these four methods are shown in Figure 11.

Adapter Tuning. Adapter tuning incorporates small neural network modules (called *adapter*) into the Transformer models [315]. To implement the adapter module, a bottleneck architecture has been proposed in [315, 316], which first compresses the original feature vector into a smaller dimension (followed by a nonlinear transformation) and then recovers it to the original dimension. The adapter modules would be integrated into each Transformer layer, typically using a serial insertion after each of the two core parts (*i.e.,* attention layer and feed-forward layer) of a Transformer layer. Alternatively, parallel adapters [317] can be also used in Transformer layers, where it places two adapter modules in parallel with the attention layer and feed-forward layer accordingly. During fine-tuning, the adapter modules would be optimized according to the specific task goals, while the parameters of the original language model are frozen in this process. In this way, we can effectively reduce the number of trainable parameters during fine-tuning.

Prefix Tuning. Prefix tuning [313] prepends a sequence of prefixes, which are a set of trainable continuous vectors, to each Transformer layer in language models. These prefix vectors are task-specific, which can be considered as virtual token embeddings. To optimize the prefix vectors, a reparameterization trick [313] has been proposed by learning a MLP function that maps a smaller matrix to the parameter matrix of prefixes, instead of directly optimizing the prefixes. It has been shown that this trick is useful for stable training. After optimization, the mapping function would be discarded, and only the derived prefix vectors are kept to enhance task-specific performance. Since only the prefix parameters would be trained, it can lead to a parameter-efficient model optimization. Similar to prefix tuning, p-tuning v2 [318] incorporates layer-wise prompt vectors into the Transformer architecture specially for natural language understanding, which also utilizes multi-task learning for jointly optimizing shared prompts. It has been shown to be useful in improving the model performance of different parameter scales on natural language understanding tasks.

Prompt Tuning. Different from prefix tuning, prompt tuning [314, 319] mainly focuses on incorporating trainable prompt vectors at the input layer[37]. Based on the discrete prompting methods [321, 322], it augments the input text by including a group of soft prompt tokens (either in a free form [319] or a prefix form [314]), and then takes the prompt-augmented input to solve specific downstream tasks. In implementation, task-specific prompt embeddings are combined with the input text embeddings, which are subsequently fed into language models. P-tuning [319] has proposed a free form to combine the context, prompt and target tokens, which can be applied to the architectures for both natural language understanding and generation. They further learn the representations of soft prompt tokens by a bidirectional LSTM. Another representative approach [314] named *prompt tuning* directly prepends prefix prompts to the input. During training, only the prompt embeddings would be learned according to task-specific supervisions. Since this method only includes a small number of trainable parameters at the input layer, it has been found that the performance highly relies on the model capacity of the underlying language models [314].

Low-Rank Adaptation (LoRA). LoRA [131] imposes the low-rank constraint for approximating the update matrix at each dense layer, so as to reduce the trainable parameters for adapting to downstream tasks. Consider the case of optimizing a parameter matrix $\mathbf{W}$. The update process can be written in a general form as: $\mathbf{W} \leftarrow \mathbf{W} + \Delta\mathbf{W}$. The basic idea of LoRA is to freeze the original matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$ while approximating the parameter update $\Delta\mathbf{W}$ by low-rank decomposition matrices, *i.e.,* $\Delta\mathbf{W} = \mathbf{A} \cdot \mathbf{B}^\top$, where $\mathbf{A} \in \mathbb{R}^{m\times k}$ and $\mathbf{B} \in \mathbb{R}^{n\times k}$ are the trainable parameters for task adaptation and $k \ll \min(m, n)$ is the reduced rank. The major merit of LoRA is that it can largely save the memory and storage usage (*e.g.,* VRAM). Further, one can only keep a single large model copy, while maintaining a number of task-specific low-rank decomposition matrices for adapting to different downstream tasks. Further, several studies have also discussed how to set the rank in a more principled approach, *e.g.,* importance score based allocation [323] and search-free optimal rank selection [324].

Besides the above methods, there is extensive research on efficient tuning of Transformer language models. However, a more comprehensive discussion of efficient tuning is beyond the scope of this article, which can be found in the related papers on this topic [317, 325].

5.3.2 Parameter-Efficient Fine-Tuning on LLMs

With the rising of LLMs, efficient tuning has attracted increasing research attention for developing a more lightweight adaptation approach in downstream tasks.

In particular, LoRA [131] has been widely applied to open-source LLMs (*e.g.,* LLaMA and BLOOM) for

37. Here, prompt tuning denotes a category of related efficient tuning methods exemplified by the work [314, 319, 320], instead of a specific method as used in [314]. Indeed, the prefix based tuning methods [313, 318] can be also considered as prompting methods, which are called *deep prompting tuning* in [318]. In this survey, prompt tuning specially refer to the methods that only include the prompt tokens at the input layer, in the context of LLMs. We assign p-tuning v2 [318] to the category of prefix tuning, because it incorporates layerwise prompts in langauge models.

parameter-efficient fine-tuning. Among these research attempts, LLaMA and its variants have gained much attention for parameter-efficient tuning. For example, Alpaca-LoRA [130] has been trained using LoRA as a lightweight tuned version of Alpaca [128] (a fine-tuned 7B LLaMA model with 52K human demonstrations of instruction following). There are extensive explorations of Alpaca-LoRA ranging in different languages or model sizes, which can be found in the collection page[38]. A recent study LLaMA-Adapter [326] inserts learnable prompt vectors into each Transformer layer, in which zero-initialized attention has been proposed to improve the training by mitigating the influence of under-fitted prompt vectors. They also extend this approach to a multi-modal setting, *e.g.,* visual question answering.

Further, an empirical study [316] has been conducted to examine the effect of different tuning methods on language models. They compare four efficient tuning methods including serial adapter tuning [315], parallel adapter tuning [317, 327], and LoRA [131], on three open-source LLMs, namely GPT-J (6B), BLOOM (7.1B) and LLaMA (7B), for evaluation. Based on the experimental results on six math reasoning datasets, they show that these efficient-tuning methods under-perform the reference baseline GPT-3.5 on difficult tasks, while achieving a comparable performance on simple tasks. Overall, LoRA performs relatively well among these comparison methods, using significantly fewer trainable parameters.

As an important resource, the library *PEFT* [328] (standing for parameter-efficient fine-tuning) has been released on GitHub[39]. It has included several widely used efficient tuning methods, including LoRA [131]/AdaLoRA [323], prefix-tuning [313, 318], P-Tuning [319], and prompt-tuning [314]. Further, it supports a number of language models such as GPT-2 and LLaMA, and also covers several representative vision Transformer models (*e.g.,* ViT and Swin Transformer).

As discussed in Section 5.3.1, there have been a large number of efficient tuning methods proposed in the existing literature. However, most of these approaches are tested on small-sized pre-trained language models, instead of the LLMs. So far, there still lacks a thorough investigation on the effect of different efficient tuning methods on large-sized language models at different settings or tasks.

5.4 Memory-Efficient Model Adaptation

Due to the huge number of model parameters, LLMs take a significant memory footprint for inference, making it very costly to be deployed in real-world applications. In this section, we discuss how to reduce the memory footprint of LLMs via a popular model compression approach (*i.e.,* model quantization), so that large-sized LLMs can be used in resource-limited settings, which also likely reduces the inference latency.

5.4.1 Background for Quantization

In this part, we present a general introduction of quantization techniques for neural networks.

In neural network compression, quantization often refers to the mapping process from floating-point numbers to integers [329], especially the 8-bit integer quantization (*i.e., INT8 quantization*). For neural network models, there are typically two kinds of data to be quantized, namely *weights* (model parameters) and *activations* (hidden activations), which are originally represented in floating-point numbers. To illustrate the essential idea of model quantization, we introduce a simple yet popular quantization function: $x_q = R(x/S) - Z$, which transforms a floating number x into a quantized value x_q. In this function, S and Z denote the scaling factor (involving two parameters α and β that determine the clipping range) and zero-point factor (determining symmetric or asymmetric quantization), respectively, and $R(\cdot)$ denotes the rounding operation that maps a scaled floating value to an approximate integer.

As the reverse process, *dequantization* recovers the original value from the quantized value accordingly: $\tilde{x} = S \cdot (x_q + Z)$. The quantization error is calculated as the numerical difference between the original value x and the recovered value $\tilde{x}$. The range parameters α and β have a large impact on the quantization performance, which often need to be *calibrated* according to real data distributions, in either a *static* (offline) or *dynamic* way (runtime).

For more details, we refer to the readers to the excellent survey [329] about quantization methods on neural networks.

5.4.2 Quantization Methods for LLMs

There are generally two major model quantization approaches, namely *quantization-aware training (QAT)* (requiring additional full model retraining) and *post-training quantization (PTQ)* (requires no model retraining). Compared with small-sized language models, two major differences need to be considered when designing or selecting quantization methods for LLMs. Firstly, LLMs consist of a huge number of parameters, and thus PTQ methods are more preferred due to a much lower computational cost than QAT methods. Secondly, LLMs exhibit very different activation patterns (*i.e.,* large outlier features), and it becomes more difficult to quantize LLMs, especially hidden activations. Next, we will briefly review several representative PTQ methods[40] for LLMs.

Post-Training Quantization (PTQ). We first introduce the PTQ methods for LLMs.

- *Mixed-precision decomposition*. As observed in [330], extreme large values occur in hidden activations (called *the emergence of outliers*) when the model size reaches 6.7B parameters or above. Interestingly, these outliers are mainly distributed in some specific feature dimensions at Transformer layers. Based on this finding, a vector-wise quantization approach, called *LLM.int8()*, has been proposed in [330], which separates the feature dimensions with outliers and the rest dimensions in matrix multiplication. Then, the calculations for the two parts are performed with *16-bit floating numbers* and *8-bit integers*, respectively, so as to recover these outliers in a high precision.

38. https://github.com/tloen/alpaca-lora
39. https://github.com/huggingface/peft

40. Since we mainly focus on discussing quantization methods in the context of LLMs, the line of quantization work on small-sized language models (*e.g.,* BERT) has not been included in this survey.

• *Fine-grained quantization*. For Transformer models, weights and activations are usually represented in the form of tensors. A straightforward approach is to use coarse-grained quantization parameters for the whole tensor (*i.e.,* per-tensor quantization) [331]. However, it usually leads to inaccurate reconstruction results. Thus, fine-grained methods are proposed to reduce the quantization error. ZeroQuant [332] adopts a token-wise quantization approach with dynamic calibration for compressing activations. Whereas for weights (easier to be quantized), it uses a group-wise quantization. In practice, a group size of 128 [332, 333] is commonly used for model quantization.

• *Balancing the quantization difficulty*. Considering that weights are easier to be quantized than activations, SmoothQuant [331] proposes to migrate the difficulty from activations to weights. Specially, they incorporate a scaling transformation to balance the difficulty between weights and activations in a linear layer: $\mathbf{Y} = (\mathbf{X}\text{diag}(\mathbf{s})^{-1}) \cdot (\text{diag}(\mathbf{s})\mathbf{W})$. By introducing an mathematically equivalent transformation, this formula controls the quantization difficulty through the scaling factor $\mathbf{s}$. To set $\mathbf{s}$, it incorporates a migration strength parameter α to balance the difficulties, where each entry $s_j = \max(\mathbf{x}_j)^{\alpha} / \max(\mathbf{w}_j)^{(1-\alpha)}$ is determined by the migration strength.

• *Layerwise quantization*. This approach finds optimal quantized weights that minimize a layerwise reconstruction loss: $\arg\min_{\widehat{\mathbf{W}}} \| \mathbf{WX} - \widehat{\mathbf{W}}\mathbf{X} \|_2^2$. To efficiently optimize this objective, GPTQ [334] improves the original optimal brain quantization (OBQ) [335] method by fixing the quantization order of weights for all rows. Further, with specially designed methods (*i.e.,* lazy batch-updates and Cholesky reformulation), GPTQ is feasible to quantize very large models (*e.g.,* 175B OPT) in 3 or 4 bit precision. More recently, AWQ [333] further simplifies the optimization form by incorporating activation-aware scaling for weights, which resembles the idea of SmoothQuant [331]: weights corresponding to outlier activations are more important to be precisely quantized. It does not directly optimize the reconstruction loss, but instead performs simple hyper-parameter search to achieve the minimal loss on calibration data.

These strategies in the above methods can be jointly used to improve the quantization performance. In order to achieve high-efficiency implementation, quantization methods also rely on hardware- or system-level support (*e.g.,* efficient GPU kernels or hardware-friendly group partition).

Other Quantization Methods. In the above, we mainly focus on PTQ methods, and next introduce two recent studies that explore efficient fine-tuning methods or QAT methods for quanitizing LLMs.

• *Efficient fine-tuning enhanced quantization.* For post-training quantization, direct low-bit quantization (*e.g.,* INT4 quantization) often results in large performance degradation. To overcome this challenge, QLoRA [336] incorporates additional small tunable adapters (16-bit precision) into the quantized models, to achieve an efficient, high-precision model fine-tuning. It combines the merits of LoRA (See Section 5.3.1) and quantization methods. The experiment results show that 4-bit quantized models can achieve the full 16-bit fine-tuning performance by QLoRA.

• *Quantization-aware training (QAT) for LLMs*. A recent study [337] explores the effect of QAT methods by applying a data-free distillation method to compress the weights, activations as well as key-value cache. By conducting extensive experiments based on LLaMA, they show promising results with 4-bit quantization on both weights and key-value cache, but not on 4-bit activation quantization, which still needs more exploration.

5.4.3 Empirical Analysis and Findings

Quantization has currently become a common technique to reduce the memory footprint and latency of LLMs in deployment. In particular, it is important to understand what level of precision (*e.g.,* INT8 or INT4) can be applied to quantize different parts of LLMs (*e.g.,* weights or activations), while retaining a high accuracy.

Recently, a very comprehensive evaluation [338] has been conducted about the impact of multiple factors (*e.g.,* model size and sensitivity) on the post-training quantization methods. Another study [339] examines the scaling law of k-bit quantization in inference performance. In addition to the overall performance, the study [340] specifically focuses on the potential impact of quantification on emergent capabilities, as well as the levels of performance that can be achieved across various levels of bit precision. Also, prior work (*e.g.,* LLM.int8() [341], GPTQ [334], QLoRA [336], and GLM [84]) has also extensively examined the performance of quantization methods in various settings. Next, we summarize several important findings from these studies, which will be useful for those who may not want to delve into the technical details of quantization methods.

• *INT8 weight quantization can often yield very good results on LLMs, while the performance of lower precision weight quantization depends on specific methods* [331, 333, 334, 338]. In most cases, INT8 weight quantization can be effectively applied to reduce the memory footprint without performance degradation. While for INT4 (or INT3) weight quantization, existing methods rely on specific strategies to reduce the performance degradation, *e.g.,* layerwise method [332, 334], activation-aware scaling [333] and low-rank adapter tuning [336]. Interestingly, LLMs seem to be less sensitive to low-bit weight quantization than small-sized language models [338]. In practice, with the same memory cost, it is suggested to use a larger language model with a lower quantization precision rather than a smaller language model with a higher quantization precision. For example, a 4-bit 60GB LLM is demonstrated to have better performance than a 8-bit 30GB LLM [339]. Moreover, focusing on emergent capabilities, the study [340] finds that in-context learning, step-by-step reasoning, and instruction following all seem to be seldom affected with 4-bit weight quantization. This result suggests that INT4 quantization exhibits a favorable trade-off in terms of both total bits and performance of emergent abilities.

• *Activations are more difficult to be quantized than weights* [330, 331, 338]. It has been found that large outliers would occur for Transformer language models having a size of 6.7B or above [330]. This issue has been one of the most fundamental difficulties to quantize LLMs. To overcome this issue, various methods, *e.g.,* mixed-precision decomposition [330], fine-grained quantization [330, 342] and difficulty migration [331], can be applied to alleviate the

influence of outlier values. Since large outliers mainly exist in the activations of LLMs, small language models are more resistant to activation quantization [338, 340]. In practice, high-quality INT8 activation quantization is still a difficult task, though several methods can attain satisfying results. Further, lower precision activation quantization has still not been successfully explored, even for QAT methods [337].

• *Efficient fine-tuning enhanced quantization is a good option to enhance the performance of quantized LLMs* [131, 336]. The benefits of efficient fune-tuning methods in quantization can be twofold. Firstly, it can directly compensate the performance degradation suffered from low-bit quantization [338, 340], by increasing the fitting capacity by updating high precision adapters. Secondly, it is flexible to support task-specific or goal-specific fine-tuning of LLMs in a lightweight way [336], *e.g.,* instruction tuning or chat-oriented tuning, by only tuning the small adapters. Overall, it makes a good trade-off between the effectiveness and training cost, which provides a promising approach to enhancing the performance of quantized LLMs.

5.4.4 Open-source Libraries and Quantized LLMs

In this part, we briefly introduce the available open-source quantization libraries and quantized LLMs.

Quantization Libraries. Next, we introduce three major quantization libraries for LLMs, including:

• *Bitsandbytes*[41] is developed based on the methods introduced in the papers of LLM.int8() [330] and 8-bit optimizers [343]. It focuses on INT8 quantization for LLMs, which mainly provides the support on 8-bit matrix multiplication and 8-bit optimizer.

• *GPTQ-for-LLaMA*[42] is developed specially for quantizing LLaMA models. It enables 4-bit quantization of LLaMA models of varied sizes based on the GPTQ algorithm [334]. Also, it provides a comparison with bitsandbytes in both memory and performance (PPL) on the project website.

• *AutoGPTQ*[43] is a quantization package developed based on the GPTQ algorithm [334], which supports INT4 quantization for LLMs. It includes a number of quantized models in the library, and supports LoRA by integrating with HuggingFace PEFT library.

• *llama.cpp*[44] makes it feasible to run quantized LLaMA models on a MacBook device. It supports INT4, INT5 and INT8 quantization, which is developed in efficient C/C++ implementation. It also supports a number of LLaMA based models, such as Alpaca and Vicuna.

Quantized LLMs. Compared with original models, quantized language models take a smaller memory footprint, and likely have a faster inference speed [84, 330, 344]. Recently, a nubmer of quantized model copies of several publicly available language models have been released on HuggingFace, including BLOOM, GPT-J, and ChatGLM. In particular, GPTQ [334] has been widely used to quantize generative language models, leading to various quantized variants for LLaMA and OPT. Further, it has been also applied to quantize instruction-tuned models, such as Vicuna and WizardLM. Due to the large number of quantized LLMs, we do not directly incorporate the corresponding links of these models. The readers can easily find them by searching on HuggingFace.

41. https://github.com/TimDettmers/bitsandbytes
42. https://github.com/qwopqwop200/GPTQ-for-LLaMa
43. https://github.com/PanQiWei/AutoGPTQ
44. https://github.com/ggerganov/llama.cpp

6 Utilization

After pre-training or adaptation tuning, a major approach to using LLMs is to design suitable prompting strategies for solving various tasks. A typical prompting method is *in-context learning* [50, 55], which formulates the task description and/or demonstrations in the form of natural language text. In addition, *chain-of-thought prompting* [33] can be employed to enhance in-context learning by involving a series of intermediate reasoning steps in prompts. Furthermore, *planning* [357] is proposed for solving complex tasks, which first breaks them down into smaller sub-tasks and then generates a plan of action to solve these sub-tasks one by one. We summarize representative work for these prompting approaches in Table 9. Next, we will elaborate on the details of the three techniques.

6.1 In-Context Learning

As a special prompting form, in-context learning (ICL) is first proposed along with GPT-3 [55], which has become a typical approach to utilizing LLMs.

6.1.1 Prompting Formulation

As stated in [55], ICL uses a formatted natural language prompt, consisting of the task description and/or a few task examples as demonstrations. Figure 12 presents the illustration of ICL. First, starting with a task description, a few examples are selected from the task dataset as demonstrations. Then, they are combined in a specific order to form natural language prompts with specially designed templates. Finally, the test instance is appended to the demonstration as the input for LLMs to generate the output. Based on task demonstrations, LLMs can recognize and perform a new task without explicit gradient update.

Formally, let $D_k = \{f(x_1, y_1), \ldots, f(x_k, y_k)\}$ represent a set of demonstrations with k examples, where $f(x_k, y_k)$ is the prompt function that transforms the k-th task example into natural language prompts. Given the task description I, demonstration D_k, and a new input query x_{k+1}, the prediction of the output $\hat{y}_{k+1}$ generated from LLMs can be formulated as follows[45]:

$$\text{LLM}(I, \underbrace{f(x_1, y_1), \ldots, f(x_k, y_k)}_{\text{demonstrations}}, f(\underbrace{x_{k+1}}_{\text{input}}, \underbrace{\_\_\_}_{\text{answer}})) \rightarrow \hat{y}_{k+1}. \tag{9}$$

where the actual answer y_{k+1} is left as a blank to be predicted by the LLM. Since the performance of ICL heavily relies on demonstrations, it is important to properly design

45. When ICL was introduced in the GPT-3's paper [55], it was originally defined to be a combination of the task description and demonstration examples, wherein either component is dispensable. Following this definition, when a LLM is required to solve an unseen task by using only task descriptions, it can be also considered to perform ICL for task solving, whereas the ICL ability can be enhanced by instruction tuning.

TABLE 9: Typical LLM utilization methods and their key points for ICL, CoT, and planning. Note that the key points only highlight the most important technical contribution.

Approach	Representative Work	Key Point
In-context Learning (ICL)	KATE [345]	Demonstration selection (similar; k-NN)
	EPR [346]	Demonstration selection (dense retrieval; constrative learning)
	SG-ICL [347]	Demonstration selection (LLM as the demonstration generator)
	APE [348]	Demonstration format (automatic generation & selection)
	Structured Prompting [349]	Demonstration format (grouped context encoding; rescaled attention)
	GlobalE & LocalE [350]	Demonstration order (entropy-based metric; probing set generation with LLM)
Chain-of-thought Prompting (CoT)	Complex CoT [351]	Demonstration (complexity-based selection)
	Auto-CoT [352]	Demonstration (automatic generation)
	Selection-Inference [353]	Generation (alternate between selection and inference)
	Self-consistency [354]	Generation (diverse paths; self-ensemble)
	DIVERSE [355]	Generation (diverse paths); Verification (step-wise voting)
	Rationale-augmented ensembles [356]	Generation (rationale sampling)
Planning	Least-to-most prompting [357]	Plan generation (text-based; problem decomposition)
	DECOMP [358]	Plan generation (text-based; problem decomposition)
	PS [359]	Plan generation (text-based)
	Faithful CoT [360]	Plan generation (code-based)
	PAL [361]	Plan generation (code-based; Python)
	HuggingGPT [362]	Plan generation (code-based; models from HuggingFace)
	AdaPlanner [363]	Plan refinement (skill memory)
	TIP [364]	Feedback acquisition (visual perception)
	RAP [365]	Feedback acquisition (LLM as the world model); Plan refinement (Monte Carlo Tree Search)
	ChatCoT [366]	Feedback acquisition (tool); Plan refinement (conversation between LLM and tools)
	ReAct [367]	Feedback acquisition (tool); Plan refinement (synergizing reasoning and acting)
	Reflexion [368]	Feedback acquisition (text-based self-reflection); Plan refinement (dynamic memory)
	Tree of Thoughts [369]	Feedback acquisition (vote comparison); Plan refinement (tree-based search)

them in the prompts. According to the construction process in Equation (9), we focus on three major aspects of formatting demonstrations in the prompts, including how to select examples that make up demonstrations, format each example into the prompt with the function $f(\cdot)$, and arrange demonstrations in a reasonable order.

A comprehensive review of ICL has been presented in the survey paper [50], and we suggest the readers referring to it for a more general, detailed discussion on this topic. Compared with this survey, we specially focus on the discussion of applying ICL to LLMs in two major aspects, *i.e.,* demonstration design and the underlying mechanism of ICL. Also, ICL has a close connection with instruction tuning (discussed in Section 5.1) in that both utilize natural language to format the task or instances. However, instruction tuning needs to fine-tune LLMs for adaptation, while ICL only prompts LLMs for utilization. Furthermore, instruction tuning can enhance the ICL ability of LLMs to perform target tasks, especially in the zero-shot setting (only using task descriptions) [64].

6.1.2 Demonstration Design

Several studies have shown that the effectiveness of ICL is highly affected by the design of demonstrations [350, 370, 371] Following the discussion in Section 6.1.1, we will introduce the demonstration design of ICL from three major aspects, *i.e.,* demonstration selection, format, and order.

Demonstration Selection. The performance of ICL tends to have a large variance with different demonstration examples [345], so it is important to select a subset of examples that can effectively leverage the ICL capability of LLMs. There are two main demonstration selection approaches, namely heuristic and LLM-based approaches:

• *Heuristic approaches.* Due to their simplicity and low costs, existing work widely adopts heuristic methods to select demonstrations. Several studies employ a k-NN based retriever to select examples that are semantically relevant to the query [345, 372]. However, they perform the selection individually for each example, rather than evaluating the example set as a whole. To resolve this issue, diversity-based selection strategies are proposed to choose the most representative set of examples for specific tasks [373, 374]. Furthermore, in [375], both relevance and diversity are taken into consideration when selecting demonstrations.

• *LLM-based approaches.* Another line of work selects demonstrations by making use of LLMs. For example, LLMs can be utilized to directly measure the informativeness of each example according to the performance gain after adding the example [376]. In addition, EPR [346] proposes a two-stage retrieval approach that first recalls similar examples with an unsupervised method (*e.g.,* BM25) and then ranks them using a dense retriever (trained with positive and negative examples labeled by LLMs). As an alternative approach, the task of demonstration selection can be formulated into a RL problem, where LLMs serve as the reward function to provide feedback for training the policy model [377]. Since LLMs perform well for text annotation [378], some recent studies employ LLM itself as the demonstration generator without human intervention [379].

To summarize, as discussed in [380], the selected demonstration examples in ICL should contain sufficient information about the task to solve as well as be relevant to the test query, for the above two selection approaches.

Demonstration Format. After selecting task examples, the next step is to integrate and format them into a natural language prompt for LLMs. A straightforward method is to instantiate a pre-defined template with the corresponding input-output pairs [36]. To construct more informative templates, recent studies consider adding task descriptions [64] or enhancing the reasoning capability of LLMs with chain-

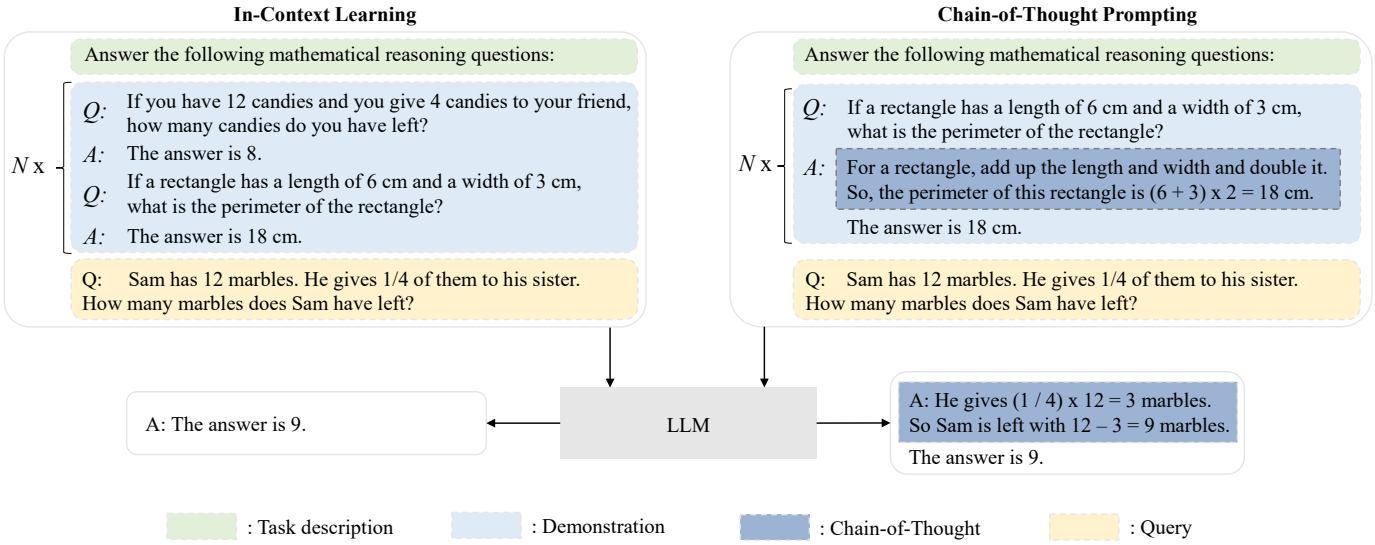


Fig. 12: A comparative illustration of in-context learning (ICL) and chain-of-thought (CoT) prompting. ICL prompts LLMs with a natural language description, several demonstrations, and a test query, while CoT prompting involves a series of intermediate reasoning steps in prompts.

of-thought prompts [33]. For instance, in [264], the authors collect a large-scale dataset with task descriptions written by humans. After tuning with this dataset, the performance on seen tasks can be boosted, and LLMs can also generalize to unseen tasks to some extent. To reduce the annotation costs, a semi-automated approach has been proposed in [129] by employing a seed set consisting of human-written task descriptions to guide LLMs to generate task descriptions for new tasks. Since it is costly to manually annotate demonstration formats for different tasks, some work also studies how to automatically generate high-quality ones. As two representative methods, Auto-CoT [352] leverages LLMs with the zero-shot prompt "*Let's think step by step*" for generating intermediate reasoning steps, while least-to-most prompting [357] first queries LLMs to perform problem decomposition and then utilizes LLMs to sequentially solve sub-problems based on the intermediate answers to previously solved ones.

Demonstration Order. LLMs are shown to sometimes suffer from the recency bias, *i.e.*, they are prone to repeat answers that are near the end of demonstrations [371]. Thus, it is important to arrange demonstrations (*i.e.*, task examples) in a reasonable order. Early work proposes several heuristic methods to quickly find a good order. For example, demonstrations can be directly organized according to their similarity to the query in the embedding space [345]: the more similar, the closer to the end. In addition, global and local entropy metrics can be used to score different demonstration orders [350]. To integrate more task information, some recent studies propose to minimize the code length required to compress and transmit task labels, which is inspired by information theory [381]. However, these methods need additional labeled data as the validation set to evaluate the performance of specific demonstration orders. To eliminate this need, the authors in [350] propose to sample the validation data from the LLM itself.

6.1.3 Underlying Mechanism

After pre-training, LLMs can exhibit intriguing ICL capability without being updated. In what follows, we discuss two key questions about the ICL ability of LLMs, *i.e.*, "*how does pre-training affect the ICL ability*" and "*how do LLMs perform ICL during inference*".

How Pre-Training Affects ICL? ICL is first proposed in GPT-3 [55], and it has been shown that the ICL ability becomes more significant with a larger model size. Further, some studies reveal that small-scale PLMs can also demonstrate a strong ICL ability by continual pre-training [382] or fine-tuning [383] on specially designed training tasks, which typically involve additional task examples in the input during the training process. It suggests that the design of training tasks is an important influence factor on the ICL capability of LLMs. Besides training tasks, recent studies have also investigated the relationship between ICL and pre-training corpora [380, 384]. For example, ICL can be theoretically explained as the product of pre-training on documents that exhibit long-range coherence [380]. Further, another study [384] theoretically analyzes that when scaling parameters and data, LLMs based on next-word prediction can emerge the ability of ICL by learning from the compositional structure (*e.g.*, how words and phrases are combined to form larger linguistic units like sentences) present in language data.

How LLMs Perform ICL? At the inference stage, researchers focus on analyzing how the ICL capability operates based on given demonstrations since no explicit learning or updating is involved. According to the discussion in [385], there are two main ways for LLMs to utilize demonstrations: task recognition and task learning.

- *Task recognition.* In the first way, LLMs recognize the task from demonstrations and utilize the prior knowledge obtained from pre-training to solve new test tasks. A Probably Approximately Correct (PAC) framework [386] has been

proposed to assess the learnability of ICL. It assumes that there exists a latent variable representing the task in the pre-training data, and LLMs have been shown to be capable of capturing this variable from demonstrations, enabling them to recognize the task in ICL. Also, the interpretation of ICL as task recognition is supported by several empirical studies [370, 387]. For example, it has been observed that replacing the inputs or labels of demonstrations with random ones sampled from the input or label space does not seriously hurt the performance of LLMs, indicating that LLMs mainly recognize the target task from demonstrations instead of learning from them [370, 385]. Similarly, LLMs can exhibit decent performance even if the prompt template is irrelevant or misleading [387].

• *Task learning.* In the second way, LLMs learn new tasks unseen in the pre-training stage only through demonstrations. Specially, task learning is analyzed mainly from the perspective of gradient descent and considered as implicit fine-tuning [60, 388]. Then, ICL can be explained as follows: by means of forward computation, LLMs generate meta-gradients with respect to demonstrations and implicitly perform gradient descent via the attention mechanism. Experiments also show that certain attention heads in LLMs are capable of performing task-agnostic atomic operations (*e.g.,* copying and prefix matching), which are closely related to the ICL ability [389]. Furthermore, some studies abstract ICL as an algorithm learning process [390]. For example, the authors in [390] find that LLMs essentially encode implicit models through their parameters during pre-training. With the examples provided in ICL, LLMs can implement learning algorithms such as gradient descent or directly compute the closed-form solution to update these models during forward computation. Under this explanation framework, it has been shown that LLMs can effectively learn simple linear functions and even some complex functions like decision trees with ICL [390].

As discussed in a recent study [385], LLMs exhibit the abilities of both task recognition and task learning in ICL, but the two abilities seem to be possessed with different model scales. As shown in the experiments [385], the ability of task recognition is easier to obtain, and even a small LM with only 350M parameters can exhibit this ability, while task learning can only emerge for LLMs with at least 66B parameters. Another study [391] also supports this finding with specially designed experiments. They set up the tasks with flipped and semantically unrelated labels in the experiment, which require task learning when performing ICL. The results suggest that small LMs tend to disregard the labels and mainly depend on their prior knowledge to accomplish the task, while LLMs have the ability to surpass their prior knowledge and acquire new knowledge from demonstrations, resulting in better outcomes. Furthermore, to improve the task learning ability, Meta-In-Context Learning [392] proposes to include multiple related tasks instead of just a single one in the prompt. In addition, Symbol Tuning [393] fine-tunes LLMs on demonstrations with semantically unrelated labels (*e.g.,* foo/bar instead of positive/negative for sentiment analysis), forcing LLMs to learn the task from demonstrations instead of relying on prior knowledge.

6.2 Chain-of-Thought Prompting

Chain-of-Thought (CoT) [33] is an improved prompting strategy to boost the performance of LLMs on complex reasoning tasks, such as arithmetic reasoning [394], commonsense reasoning [395], and symbolic reasoning [33]. Instead of simply constructing the prompts with input-output pairs as in ICL, CoT incorporates intermediate reasoning steps that can lead to the final output into the prompts. In the following, we will elaborate on the usage of CoT with ICL and discuss when and why CoT prompting works.

6.2.1 In-context Learning with CoT

Typically, CoT can be used with ICL in two major settings, namely the few-shot and zero-shot settings, as introduced below.

Few-shot CoT. Few-shot CoT is a special case of ICL, which augments each demonstration $\langle input, output \rangle$ as $\langle input, CoT, output \rangle$ by incorporating the CoT reasoning steps. To apply this strategy, we next discuss two key issues, *i.e.,* how to design appropriate CoT prompts and how to utilize the generated CoTs for deriving the final answer.

• *CoT prompt design.* It is critical to design appropriate CoT prompts for effectively eliciting the complex reasoning abilities of LLMs. As a direct approach, it is shown that using diverse CoTs (*i.e.,* multiple reasoning paths for each problem) can effectively enhance their performance [355]. Another intuitive idea is that prompts with more complex reasoning paths are more likely to elicit the reasoning ability of LLMs [351], which can result in higher accuracy in generating correct answers. However, all these approaches rely on annotated CoT datasets, which limits their use in practice. To overcome this limitation, Auto-CoT [352] proposes to utilize Zero-shot-CoT [396] (detailed in the following part "*Zero-shot CoT*") to generate CoT reasoning paths by specially prompting LLMs, thus eliminating manual efforts. In order to boost the performance, Auto-CoT further divides the questions in the training set into different clusters and then chooses the questions that are closest to the centroid of each cluster, which is supposed to well represent the questions in the training set. Although few-shot CoT can be considered as a special prompt case of ICL, the ordering of demonstrations seems to have a relatively small impact compared to the standard prompt in ICL: reordering the demonstrations only results in a performance variation of less than 2% in most tasks [33].

• *Enhanced CoT strategies.* In addition to enriching the contextual information, CoT prompting also provides more options to infer the answer given a question. Existing studies mainly focus on generating multiple reasoning paths, and try to find a consensus among the derived answers [354, 356]. For instance, *self-consistency* [354] is proposed as a new decoding strategy when generating CoT and the final answer. It first generates several reasoning paths and then takes an ensemble over all the answers (*e.g.,* selecting the most consistent answer by voting among these paths). Self-consistency boosts the performance in CoT reasoning by a large margin, and can even improve some tasks where CoT prompting is usually worse than standard prompting (*e.g.,* closed-book question answering and natural language

inference). Further, the authors in [356] expand the self-consistency strategy to a more general ensemble framework (extending to ensemble on the prompts), and they find that diverse reasoning paths are the key to the performance improvement in CoT reasoning. The above methods can be easily integrated into CoT prompting to enhance the performance without additional training. In contrast, other studies train a scoring model to measure the reliability of the generated reasoning paths [355] or continually train LLMs on the reasoning paths generated by themselves [397] to improve the performance.

Zero-shot CoT. Different from few-shot CoT, zero-shot CoT does not include human-annotated task demonstrations in the prompts. Instead, it directly generates reasoning steps and then employs the generated CoTs to derive the answers. Zero-shot CoT is first proposed in [396], where the LLM is first prompted by "*Let's think step by step*" to generate reasoning steps and then prompted by "*Therefore, the answer is*" to derive the final answer. They find that such a strategy drastically boosts the performance when the model scale exceeds a certain size, but is not effective with small-scale models, showing a significant pattern of emergent abilities. In order to unlock the CoT ability on more tasks, Flan-T5 and Flan-PaLM [64] further perform instruction tuning on CoT annotations and the zero-shot performance on unseen tasks has been improved.

6.2.2 Further Discussion on CoT

In this part, we present discussions regarding two fundamental questions related to CoT, *i.e.*, "*when does CoT work for LLMs*" and "*why can LLMs perform CoT reasoning*".

When CoT works for LLMs? Since CoT is an emergent ability [31], it only has a positive effect on sufficiently large models (typically containing 10B or more parameters [33]) but not on small models. Moreover, since CoT augments the standard prompting with intermediate reasoning steps, it is mainly effective for the tasks that require step-by-step reasoning [33], *e.g.*, arithmetic reasoning, commonsense reasoning, and symbolic reasoning. Whereas, for other tasks that do not rely on complex reasoning, CoT might lead to worse performance than standard prompting [356], *e.g.*, MNLI-m/mm, SST-2, and QQP from GLUE [206]. Interestingly, it seems that the performance gain brought by CoT prompting could be significant only when standard prompting yields poor results [33].

Why LLMs Can Perform CoT Reasoning? As the second question, we discuss the underlying mechanism of CoT in the following two aspects.

• *The source of CoT ability*. Regarding the source of CoT capability, it is widely hypothesized that it can be attributed to training on code since models trained on it show a strong reasoning ability [47, 398]. Intuitively, code data is well organized with algorithmic logic and programming flow, which may be useful to improve the reasoning performance of LLMs. However, this hypothesis still lacks publicly reported evidence of ablation experiments (*with* and *without* training on code). In addition, instruction tuning seems not to be the key reason to obtain the CoT ability, since it has been empirically shown that instruction tuning on non-CoT data does not improve the performance on held-out CoT benchmarks [64].

• *The effect of prompting components*. The major distinction between CoT prompting and standard prompting is the incorporation of reasoning paths prior to the final answer. Thus, some researchers investigate the effects of different components in the reasoning paths. Specifically, a recent study identifies three key components in CoT prompting, namely *symbols* (*e.g.*, numerical quantities in arithmetic reasoning), *patterns* (*e.g.*, equations in arithmetic reasoning), and *text* (*i.e.*, the rest of tokens that are not symbols or patterns) [399]. It is shown that the latter two parts (*i.e.*, patterns and text) are essential to the model performance, and removing either one would lead to a significant performance drop. However, the correctness of symbols and patterns does not seem critical. Further, there exists a symbiotic relationship between text and patterns: the text helps LLMs to generate useful patterns, and patterns aid LLMs to understand tasks and generate texts that help solve them [399].

In summary, CoT prompting provides a general yet flexible approach to eliciting the reasoning ability of LLMs. There are also some preliminary attempts to extend this technique to solve multimodal [400] and multilingual tasks [401].

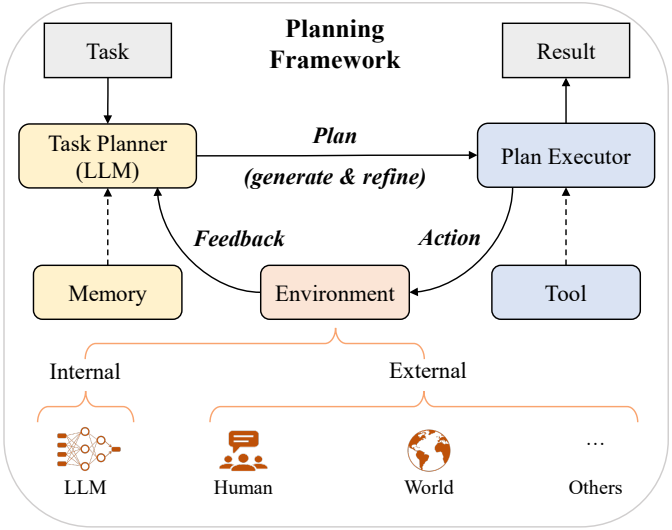


Fig. 13: An illustration of the formulation for prompt based planning by LLMs for solving complex tasks.

6.3 Planning for Complex Task Solving

Prompting with ICL and CoT is a conceptually simple yet general approach to solving various tasks. However, this approach struggles with complex tasks like mathematical reasoning [402] and multi-hop question answering [403]. As an enhanced approach, prompt-based planning has been proposed to break down complex tasks into smaller subtasks and generate a plan of actions to accomplish the task.

6.3.1 The Overall Framework

In this part, we first formulate the general planning paradigm of LLMs for solving complex tasks, which is illustrated in Figure 13.

In this paradigm, there are typically three components: *task planner*, *plan executor*, and *environment*[46]. Specifically, task planner, which is played by LLMs, aims to generate the whole plan to solve a target task. The plan can be presented in various forms, *e.g.,* an action sequence in the form of natural language [357] or an executable program written in programming language [361]. The LLM-based task planner can be enhanced with the memory mechanism for plan storage and retrieval, which is helpful for long-horizon tasks. Then, plan executor is responsible for executing the actions in the plan. It can be implemented by models like LLMs for textual tasks [359] or by tools like code interpreters for coding tasks [368]. Furthermore, environment refers to where the plan executor carries out the actions, which can be set differently according to specific tasks, *e.g.,* the LLM itself [404] or an external virtual world like Minecraft [405]. It provides *feedback* about the execution result of the action to the task planner, either in the form of natural language [368] or from other multimodal signals [364].

For solving a complex task, the task planner first needs to clearly understand the task goal and generate a reasonable plan based on the reasoning of LLMs (See Section 6.3.2). Then, the plan executor acts according to the plan in the environment, and the environment will produce feedback for the task planner (See Section 6.3.3). The task planner can further incorporate the feedback obtained from the environment to refine its initial plan and iteratively perform the above process to get better results as the task solution (See Section 6.3.4).

Next, we will introduce the three key steps in planning based task solving.

6.3.2 Plan Generation

Plan generation focuses on directly generating action sequences by prompting LLMs. Based on the format of the generated plans, existing work can be divided into two groups: text-based and code-based approaches.

Text-based Approaches. It is straightforward for LLMs to generate plans in the form of natural language. In this approach, LLMs are prompted to generate a sequence of actions for the plan executor to perform and solve the complex task. For example, Plan-and-Solve [359] adds explicit instructions like "`devise a plan`" to directly prompt the LLM for planning in a zero-shot manner, while Self-planning [406] and DECOMP [358] add demonstrations in the prompt to guide the LLM to devise a plan through ICL. Following this way, some work further considers incorporating extra tools or models when planning. For example, ToolFormer [71] first annotates a pre-training corpus with potential API calls using LLMs, and then fine-tunes LLMs on it, so that LLMs can learn when and how to call APIs and incorporate the results returned by APIs during generation. HuggingGPT [362] introduces the models available in HuggingFace and regards LLMs as the controller to select suitable models based on their descriptions and aggregate their results as the final solution.

46. Despite the similarity with RL, our formulation decouples the planning and execution phases, whereas in RL, they are typically interleaved in the agent. This paradigm is defined in a general yet slightly loose way, and it mainly aims to help readers understand the key idea underlying the planning approaches of LLMs.

Code-based Approaches. Although text-based approaches sound intuitive, they cannot guarantee faithful execution of the plan, which may lead to failure even when the plan is sound. To address this issue, code-based approaches have been proposed to generate more verifiable plans in the form of executable code in programming languages, *e.g.,* Python or PDDL. In this way, LLMs are first prompted to generate the program and then utilize a deterministic solver to execute it. For example, Faithful CoT [360] and PAL [361] decompose a reasoning task into two stages: at the first stage, the LLM generates a plan conditioned on the query; at the second stage, a deterministic solver executes the plan to derive the final answer. Furthermore, code-based approaches can be applied to embodied agents in a similar way. For example, PROGPROMPT [407] and LLM+P [408] first utilize LLMs to generate plans in the form of python functions or PDDL files, and then leverage a virtual agent or classical planner to solve the problem according to the code-based plans.

6.3.3 Feedback Acquisition

After executing the generated plan, the environment would produce the feedback signal to the LLM-based task planner, which can be used to refine its initial plan for better results. In existing work, there are typically two sources of feedback from the environment, depending on their relationship with the LLM-based task planner: internal (*i.e.,* the LLM itself) and external (*e.g.,* tools or virtual worlds) feedback.

Internal Feedback. The LLM itself can be utilized as a feedback provider. One straightforward way is to directly evaluate the quality of the generated plans through prompting. For example, RAP [365] evaluate the likelihood that each candidate plan can lead to task success, while Tree of Thoughts [404] proposes to vote across plans by making comparisons between them. Further, LLMs can provide feedback based on the intermediate results from the plan executor. For example, Reflexion [368] utilizes LLMs to transform sparse result signals (*e.g.,* success or failure) into concrete text-based feedback (*e.g.,* "*You should recommend comedies that the user mentions in the query instead of horror movies*") and stores this feedback in long-term memory for future planning.

External Feedback. In addition to LLMs, external objects can also provide feedback signals. For example, tools like code interpreters are widely used in programming tasks to provide real-time error messages [368], models like stable diffusion [409] can be used in multimodal tasks to provide visual perception [364], and virtual worlds like Minecraft can provide immersive experiences [405]. Besides, some work (*e.g.,* Generative Agents [410]) explores multi-agent collaboration in simulated environments, where each agent receives feedback not only from interaction with the environment but also from communication with other agents.

6.3.4 Plan Refinement

With access to feedback from the environment, the task planner can accordingly refine its current plan and iteratively go through the "*planning – execution – refinement*" loop for better results. In this part, we summarizes three major refinement approaches in existing work.

Reasoning. The feedback data from the environment may not be directly suitable to be utilized by LLMs for plan refinement, *e.g.,* containing irrelevant information or taking a non-language form. To solve this, some work adds the explicit reasoning process to extract critical information from feedback [366, 367]. For example, React [367] prompts LLMs with demonstrations to generate reasoning traces over feedback. It has been widely used in autonomous agent projects, such as AutoGPT[47], which can automatically reason over the observed feedback to revise the initial plan for solving various user requests. However, these approaches typically fix the order of reasoning and planning. To support flexible switching between the two processes for better performance, ChatCoT [366] further unifies the tool-augmented reasoning process into a multi-turn conversation between the LLM-based task planner and the tool-based environment.

Backtracking. Early methods mainly consider planning forward actions while maintaining the existing plan, thus likely leading to local optimal plans based on a short-term evaluation. To solve this, Tree of Thoughts [404] allows backtracking with search algorithms like breadth-first and depth-first search to make global planning. It refines the plan step by step by backtracking to the last state in the initial plan and choosing the next unexplored action. Furthermore, some studies [364, 411] utilize feedback signals to revise the entire plan. For example, DEPS [411] selects a better plan according to feedback signals, while TIP [364] adds feedback signals to prompts for the LLM-based planner to revise each step in the initial plan.

Memorization. In order to handle long-horizon tasks, it has become a key approach to aid plan refinement with *long-term memory* in addition to utilizing the *short-term memory* of LLMs through ICL. For example, Reflexion [368] stores the feedback from self-reflection into the memory, so previous feedback can be retrieved for plan refinement. Generative Agents [410] designs the memory stream mechanism as the core component of agents for action planning and reflection. Further, the skill library mechanism [363, 405] is proposed to store successful plans in the library, which can be reused and synthesized as complex plans for novel tasks. To implement the long-term memory mechanism, tools like vector databases (*e.g.,* milvus [412]) can be used to encode plans or feedbacks into high-dimensional vectors for efficient storage and retrieval at a large scale. MemoryBank [413] further proposes the memory updating mechanism to allow memory forgetting and strengthening following the Ebbinghaus Forgetting Curve theory.

7 Capacity and Evaluation

To examine the effectiveness and superiority of LLMs, a surge of tasks and benchmarks have been proposed for conducting empirical ability evaluation and analysis. In this section, we first introduce three types of basic ability evaluation of LLMs for language generation and understanding, then present several advanced ability evaluations of LLMs with more complicated settings or goals, and finally discuss existing benchmarks, evaluation approaches, and empirical analysis.

47. https://github.com/Significant-Gravitas/Auto-GPT

7.1 Basic Ability

In this part, we mainly focus on three basic types of ability evaluation for LLMs, *i.e.,* language generation, knowledge utilization, and complex reasoning. It is noted that we do not intend to have complete coverage of all the related tasks, but instead only focus on the most widely discussed or studied tasks for LLMs. Next, we introduce these tasks in detail.

7.1.1 Language Generation

According to the task definition, existing tasks about language generation can be roughly categorized into language modeling, conditional text generation, and code synthesis tasks. Note that code synthesis is not a typical NLP task, we include it for discussion because it can be directly solved by a number of LLMs (trained on code data) in a similar generation approach as natural language text.

Language Modeling. As the most fundamental ability of LLMs, *language modeling* aims to predict the next token based on the previous tokens [15], which mainly focuses on the capacity of basic language understanding and generation. For evaluating such an ability, typical language modeling datasets that existing work uses include Penn Treebank [414], WikiText-103 [415], and the Pile [146], where the metric of *perplexity* is commonly used for evaluating the model performance under the zero-shot setting. Empirical studies [55, 84] show that LLMs bring substantial performance gains over the previous state-of-the-art methods on these evaluation datasets. To better test the modeling capacity of long-range dependencies in text, the LAMBADA dataset [188] has been introduced, where LLMs are required to predict the last word of sentences based on a paragraph of context. Then, the accuracy and perplexity of the predicted last words are employed to evaluate LLMs. As shown in existing work, the performance on the language modeling tasks typically follows the scaling law [30], which means that scaling language models would improve the accuracy and reduce the perplexity.

Conditional Text Generation. As an important topic in language generation, conditional text generation [48] focuses on generating texts satisfying specific task demands based on the given conditions, typically including machine translation [503], text summarization [424], and question answering [434]. To measure the quality of the generated text, automatic metrics (*e.g.,* Accuracy, BLEU [504] and ROUGE [505]) and human ratings have been typically used for evaluating the performance. Due to the powerful language generation capabilities, LLMs have achieved remarkable performance on existing datasets and benchmarks. For instance, GPT-4 exhibits comparable performance as commercial translation products, even for the translation task of languages that are with significant linguistic distance [506]. On news summarization tasks (*i.e.,* CNN/DM and XSUM), LLMs also demonstrate comparable performance with human freelance writers [507]. Despite the rapid progress on model capacity, there are increasing concerns on the feasibility of existing automatic metrics to faithfully assess

TABLE 10: Representative basic and advanced abilities and corresponding representative datasets for evaluating.

Level	Ability	Task	Dataset
Basic	Language Generation	Language Modeling	Penn Treebank [414], WikiText-103 [415], the Pile [146], LAMBADA [188]
		Conditional Text Generation	WMT'14,16,19,20,21,22 [416–421], Flores-101 [422], DiaBLa [423], CNN/DailyMail [424], XSum [425], WikiLingua [426] OpenDialKG [427]
		Code Synthesis	APPS [428], HumanEval [92], MBPP [168], CodeContest [101], MTPB [77], DS-1000 [429], ODEX [430]
	Knowledge Utilization	Closed-Book QA	Natural Questions [431], ARC [432], TruthfulQA [433], Web Questions [434], TriviaQA [435], PIQA [436], LC-quad2.0 [437], GrailQA [438], KQApro [439], CWQ [440], MKQA [441], ScienceQA [442]
		Open-Book QA	Natural Questions [431], OpenBookQA [443], ARC [432], TriviaQA [435], Web Questions [434], MS MARCO [444], QASC [445], SQuAD [446], WikiMovies [447]
		Knowledge Completion	WikiFact [448], FB15k-237 [449], Freebase [450], WN18RR [451], WordNet [452], LAMA [453], YAGO3-10 [454], YAGO [455]
	Complex Reasoning	Knowledge Reasoning	CSQA [395], StrategyQA [456], HotpotQA [457], ARC [432], BoolQ [458], PIQA [436], SIQA [459], HellaSwag [460], WinoGrande [461], COPA [462], OpenBookQA [443], ScienceQA [442], proScript [463], ProPara [464], ExplaGraphs [465], ProofWriter [466], EntailmentBank [467], ProOntoQA [468]
		Symbolic Reasoning	CoinFlip [33], ReverseList [33], LastLetter [33], Boolean Assignment [469], Parity [469], Colored Object [292], Penguins in a Table [292], Repeat Copy [361], Object Counting [361]
		Mathematical Reasoning	MATH [291], GSM8k [470], SVAMP [471], MultiArith [472], ASDiv [394], MathQA [473], AQUA-RAT [474], MAWPS [475], DROP [476], NaturalProofs [477], PISA [478], miniF2F [479], ProofNet [480]
Advanced	Human Alignment	Honestness	TruthfulQA [433], HaluEval [481]
		Helpfulness	HH-RLHF [267]
		Harmlessness	HH-RLHF [267], Crows-Pairs [482] WinoGender [483], RealToxicityPrompts [484]
	Interaction with External Environment	Household	VirtualHome [485], BEHAVIOR [486], ALFRED [487],ALFWorld [488]
		Website Environment	WebShop [489], Mind2Web [490]
		Open World	MineRL [491], MineDojo [492]
	Tool Manipulation	Search Engine	HotpotQA [457], TriviaQA [435], Natural Questions [431]
		Code Executor	GSM8k [470], TabMWP [493], Date Understanding [292]
		Calculator	GSM8k [470], MATH [291], CARP [494]
		Model Interface	GPT4Tools [495], Gorilla [496]
		Data Interface	WebQSP [497], MetaQA [498], WTQ [499] WikiSQL [500], TabFact [501], Spider [502]

the performance of LLMs in conditional text generation tasks [507–509]. As the alternatives to automatic metrics, recent studies also propose to incorporate LLMs as generation evaluators to examine the quality of the generated content [124, 510, 511]. Moreover, researchers also explore more challenging language generation tasks for LLMs, such as structured data generation [512] and long text generation [46, 513, 514].

Code Synthesis. In addition to generating high-quality natural language text, existing LLMs also show strong abilities to generate formal language, especially computer programs (*i.e.,* code) that satisfy specific conditions, called *code synthesis* [515]. Unlike natural language generation, as the generated code can be directly checked by execution with corresponding compilers or interpreters, existing work mostly evaluates the quality of the generated code from LLMs by calculating the pass rate against the test cases, *i.e.,* pass@k[48]. Recently, several code benchmarks focusing on functional correctness are proposed to assess the code synthesis abilities of LLMs, such as APPS [428], HumanEval [92], and MBPP [168]. Typically, they consist of diverse programming problems, with text specification and test cases for correctness checking. To improve such an ability, it is key to fine-tuning (or pre-training) LLMs on code data, which can effectively adapt LLMs to code synthesis tasks [77]. In addition, existing work has proposed new strategies to generate code, *e.g.,* sampling multiple candidate solutions [168] and planning-guided decoding [516], which can be considered as the imitation of bug-fixing and code-planning processes by programmers. Impressively, LLMs have recently shown competitive performance with humans by achieving a rank-

48. Given k programs generated by the LLM, pass@k is computed as 1 when at least one program passes all test cases, or else 0

ing of the top 28% among users on the programming contest platform Codeforces [101]. Further, GitHub Copilot has been released to assist programming in coding IDEs (*e.g.,* Visual Studio and JetBrains IDEs), which can support a variety of languages including Python, JavaScript, and Java. A viewpoint article entitled "*The End of Programming*" [517] in Communications of the ACM has discussed the impact of AI programming in the field of computer science, emphasizing an important shift towards the highly adaptive LLM as a new atomic unit of computation.

Major Issues. Although LLMs have achieved splendid performance in generating human-like text, they are susceptible to suffering from two major issues in language generation as discussed below.

• *Unreliable generation evaluation.* With the advancement of language generation ability of LLMs, existing studies find that the generated texts from LLMs have reached a comparable quality to the reference texts on a variety of text generation tasks. However, due to the intrinsic weakness of existing evaluation benchmarks, there exists pronounced inconsistency between human evaluation and automatic reference-based metrics [507–509, 518]. For example, in OpenDialKG [427], ChatGPT underperforms a fine-tuned GPT-2 on BLEU and ROUGE-L metrics, while earning more favor from human judgment [518]. Furthermore, existing work argues that even human evaluation may not be robust enough [507, 508, 519, 520]. In some cases, it is difficult to achieve a high level of consensus among human annotators [508], and there is also a large gap between the annotation quality of crowdworkers and experts [519, 520]. Thus, how to conduct reliable evaluation for language generation tasks in the era of LLMs has become a fundamental yet challenging research topic. Recently, increasing research work proposes to leverage LLMs to improve the evaluation quality of the generated texts. Specially, LLMs can be used to improve the evaluation quality of existing metrics. For example, Para-Ref [521] augments various automatic metrics by leveraging LLMs to paraphrase existing references into semantically equivalent references with diverse expressions. Further, LLMs are widely employed as the evaluators of text generation in a reference-free manner, including evaluating a single prediction [510, 511, 522] or comparing several candidates [124, 523–525]. Nevertheless, LLMs may expose bias (*e.g.,* order bias or preference for LLM-generated texts over human-written texts) as language generation evaluators, demonstrating disparities when compared to human evaluation [511, 526, 527].

Unreliable Generation Evaluation

LLMs have been capable of generating texts with a comparable quality to human-written texts, which however might be underestimated by automatic reference-based metrics. As an alternative evaluation approach, LLMs can serve as language generation evaluators to evaluate a single text, compare multiple candidates, and improve existing metrics. However, this evaluation approach still needs more inspections and examinations in real-world tasks.

• *Underperforming specialized generation.* Although LLMs have learned general language patterns to generate coherent text, their proficiency in generation might be constrained when dealing with a specialized domain or task. For instance, a language model that has been trained on general web articles may face challenges when generating a medical report which involves many medical jargon and methods. Intuitively, domain knowledge should be critical for model specialization. However, it is not easy to inject such specialized knowledge into LLMs. As discussed in recent analyses [47, 528], when LLMs are trained to exhibit some specific ability that allows them to excel in some areas, they might struggle in others. Such an issue is related to *catastrophic forgetting* [529, 530] in training neural networks, which refers to the conflict phenomenon of integrating new and old knowledge. Similar cases also occur in human alignment of LLMs, where "*alignment tax*" [61] (*e.g.,* a potential loss in the in-context learning ability) has to be paid for aligning to human values and needs. Moreover, due to the limitations of sequence modeling architecture, LLMs still face challenges in the understanding and generation of structured data. Consequently, they often fall behind task-specific models on complex structured data tasks, such as knowledge-base question answering and semantic parsing [512, 531]. Therefore, it is important to develop effective model specialization methods that can flexibly adapt LLMs to various task scenarios, meanwhile retaining the original abilities as possible.

Underperforming Specialized Generation

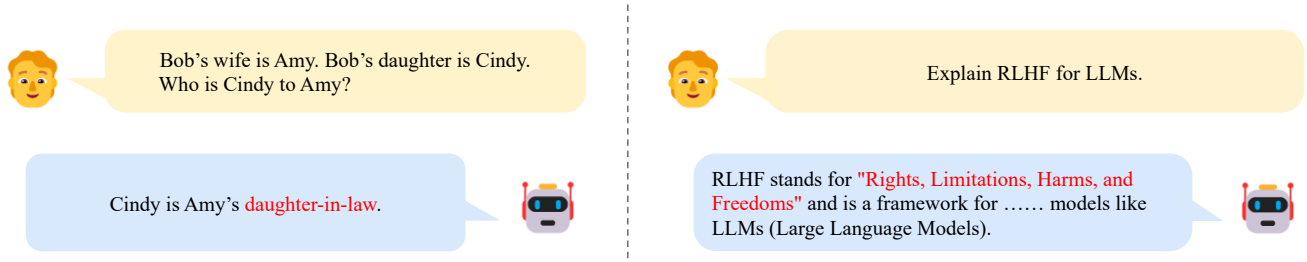
LLMs may fall short in mastering generation tasks that require domain-specific knowledge or generating structured data. It is non-trivial to inject specialized knowledge into LLMs, meanwhile maintaining the original abilities of LLMs.

7.1.2 Knowledge Utilization

Knowledge utilization is an important ability of intelligent systems to accomplish knowledge-intensive tasks (*e.g.,* commonsense question answering and fact completion) based on supporting factual evidence. Concretely, it requires LLMs to properly utilize the rich factual knowledge from the pre-training corpus or retrieve external data when necessary. In particular, question answering (QA) and knowledge completion have been two commonly used tasks for evaluating this ability. According to the test tasks (question answering or knowledge completion) and evaluation settings (*with* or *without* external resources), we categorize existing knowledge utilization tasks into three types, namely closed-book QA, open-book QA[49], and knowledge completion.

Closed-Book QA. Closed-book QA tasks [532] test the acquired factual knowledge of LLMs from the pre-training

49. In this part, open-book QA refers to the QA tasks that require to extract and utilize useful information from external knowledge resources, as the antithesis of closed-book QA (only using the encoded information from pre-training corpus). Note that there is a dataset also named OpenBookQA [443], which follows the settings of open-book QA tasks by extracting and utilizing external science facts.



(a) Intrinsic hallucination

(b) Extrinsic hallucination

Fig. 14: Examples of intrinsic and extrinsic hallucination for a public LLM (access date: March 19, 2023). As an example of intrinsic hallucination, the LLM gives a conflicting judgment about the relationship between Cindy and Amy, which contradicts the input. For extrinsic hallucination, in this example, the LLM seems to have an incorrect understanding of the meaning of RLHF (reinforcement learning from human feedback), though it can correctly understand the meaning of LLMs (in this context).

corpus, where LLMs should answer the question only based on the given context without using external resources. For evaluating this ability, there are several datasets that can be leveraged, including Natural Questions [431], Web Questions [434], and TriviaQA [435], where the accuracy metric is widely adopted. Empirical results have revealed that LLMs can perform well in this setting and even match the performance of state-of-the-art open-domain QA systems [56]. Also, the performance of LLMs on closed-book QA tasks shows a scaling law pattern in terms of both model size and data size: scaling the parameters and training tokens can increase the capacity of LLMs and help them learn (or memorize) more knowledge from the pre-training data [56]. Further, under a similar parameter scale, LLMs with more pre-training data relevant to the evaluated tasks would achieve better performance [72]. Also, the closed-book QA setting provides a testbed for probing the accuracy of the factual knowledge encoded by LLMs. However, as shown in existing work [55], LLMs might perform less well on QA tasks relying on fine-grained knowledge, even when it exists in the pre-training data.

Open-Book QA. Unlike closed-book QA, in open-book QA tasks, LLMs can extract useful evidence from the external knowledge base or document collections, and then answer the question based on the extracted evidence [533–536]. Typical open-book QA datasets (*e.g.,* Natural Questions [431], OpenBookQA [443], and SQuAD [446]) have overlap with closed-book QA datasets, but they incorporate external data sources, *e.g.,* Wikipedia. The metrics of accuracy and F1 score are widely used in open-book QA tasks for evaluation. To select relevant knowledge from external resources, LLMs are often paired with a text retriever (or even a search engine), which is trained independently or jointly with LLMs [72, 533, 537]. Also, previous work [538–540] has indicated that retrievers can assist LLMs in verifying and rectifying the reasoning path. In evaluation, existing studies mainly focus on testing how LLMs utilize the extracted knowledge to answer the question and show that the retrieved evidence can largely improve the accuracy of the generated answers, even enabling a smaller LLM to outperform $10\times$ larger ones [533, 537]. Further, open-book QA tasks can be also employed to evaluate the recency of knowledge information. Pre-training or retrieving from outdated knowledge resources may cause LLMs to generate incorrect answers for time-sensitive questions [533].

Knowledge Completion. In knowledge completion tasks, LLMs might be (to some extent) considered as a knowledge base [453], which can be leveraged to complete or predict the missing parts of knowledge units (*e.g.,* knowledge triples). Such tasks can probe and evaluate *how much* and *what kind of* knowledge LLMs have learned from the pre-training data. Existing knowledge completion tasks can be roughly divided into knowledge graph completion tasks (*e.g.,* FB15k-237 [449] and WN18RR [451]) and fact completion tasks (*e.g.,* WikiFact [448]), which aim to complete the triples from a knowledge graph and incomplete sentences about specific facts, respectively. Empirical studies have revealed that it is difficult for existing LLMs to accomplish knowledge completion tasks related to specific relation types [398]. As shown in the evaluation results on WikiFact, LLMs perform well on several frequent relations that occur in the pre-training data (*e.g.,* `currency` and `author`), while not well on rare ones (*e.g.,* `discoverer_or_inventor` and `place_of_birth`). Interestingly, under the same evaluation settings (*e.g.,* in-context learning), InstructGPT (*i.e.,* `text-davinci-002`) outperforms GPT-3 in all subsets of WikiFact.

Major Issues. Although LLMs have achieved key progress in capturing and utilizing knowledge information, they suffer from two major issues as discussed below.

• *Hallucination*. In generating factual texts, a challenging issue is *hallucination generations* [518], where the generated information is either in conflict with the existing source (*intrinsic hallucination*) or cannot be verified by the available source (*extrinsic hallucination*), which are illustrated by two examples in Figure 14. Hallucination widely occurs in existing LLMs, even the most superior LLMs such as GPT-4 [46]. Furthermore, existing work shows that LLMs encounter difficulties in recognizing the hallucinated content in text [481], even the powerful ChatGPT. Additionally, beyond language tasks, a recent study has shown that large vision-language models (LVLM) also face challenges with hallucination, *i.e.,* generating objects that are not present in the accompanying images [541]. In essence, LLMs seem

to "unconsciously" utilize the knowledge in task solving, which still lack an ability to accurately control the use of internal or external knowledge. Hallucinations would mislead LLMs to generate undesired outputs and mostly degrade the performance, leading to potential risks when deploying LLMs in real-world applications. To alleviate this problem, alignment tuning strategies (as discussed in Section 5.2) have been widely utilized in existing work [61], which rely on tuning LLMs on high-quality data or using human feedback. Moreover, the integration of external tools for the provision of credible information sources can help alleviate the hallucination issue [72, 481, 539]. Another line of research work leverages uncertainty estimation of LLMs to identify hallucinations [542, 543]. For instance, considering that hallucinated facts are prone to exhibit inconsistency across different sampled outputs, SelfCheckGPT [543] detects hallucination by measuring information inconsistency within sampled outputs. For the evaluation of the hallucination problem, a set of hallucination detection tasks have been proposed, *e.g.,* TruthfulQA [433] for detecting human falsehood mimicked by models. More recently, HaluEval [481] creates a large-scale LLM-generated and human-annotated hallucinated samples to evaluate the ability of language models to recognize hallucination in both task-specific and general scenarios.

Hallucination

LLMs are prone to generate untruthful information that either conflicts with the existing source or cannot be verified by the available source. Even the most powerful LLMs such as ChatGPT face great challenges in migrating the hallucinations the generated texts. This issue can be partially alleviated by special approaches such as alignment tuning and tool utilization.

• *Knowledge recency*. As another major challenge, LLMs would encounter difficulties when solving tasks that require the latest knowledge beyond the training data. To tackle this issue, a straightforward approach is to regularly update LLMs with new data. However, it is very costly to fine-tune LLMs, and also likely to cause the catastrophic forgetting issue when incrementally training LLMs. Therefore, it is necessary to develop efficient and effective approaches that can integrate new knowledge into existing LLMs, making them up-to-date. Existing studies have explored how to utilize the external knowledge source (*e.g.,* search engine) to complement LLMs, which can be either jointly optimized with LLMs [533] or used as a plug-and-play module [539]. For instance, ChatGPT utilizes a retrieval plugin to access up-to-date information sources [544]. By incorporating the extracted relevant information into the context [545–547], LLMs can acquire new factual knowledge and perform better on relevant tasks. However, such an approach seems to be still at a superficial level. In addition, existing studies also explore editing parameters of language models to update intrinsic knowledge [548–550]. Nevertheless, previous work [551] has shown that several parameter editing methods perform not well on LLMs, though they can improve the performance of small language models. Therefore, it is still difficult to directly amend intrinsic knowledge or inject specific knowledge into LLMs, which remains an open research problem [551]. Recently, a useful framework *EasyEdit* [552] has been released to facilitate the research of knowledge editing for LLMs.

Knowledge Recency

The parametric knowledge of LLMs is hard to be updated in a timely manner. Augmenting LLMs with external knowledge sources is a practical approach to tackling the issue. However, how to effectively update knowledge within LLMs remains an open research problem.

7.1.3 Complex Reasoning

Complex reasoning refers to the ability of understanding and utilizing supporting evidence or logic to derive conclusions or make decisions [51, 52]. According to the type of involved logic and evidence in the reasoning process, we consider dividing existing evaluation tasks into three major categories, namely knowledge reasoning, symbolic reasoning, and mathematical reasoning.

Knowledge Reasoning. The knowledge reasoning tasks rely on logical relations and evidence about factual knowledge to answer the given question. Existing work mainly uses specific datasets to evaluate the reasoning capacity of the corresponding type of knowledge, *e.g.,* CSQA [395]/StrategyQA [456] for commonsense knowledge reasoning and ScienceQA [442] for science knowledge reasoning. In addition to the accuracy of the predicted results, existing work [442] has also evaluated the quality of the generated reasoning process, via automatic metrics (*e.g.,* BLEU) or human evaluation. Typically, these tasks require LLMs to perform step-by-step reasoning based on factual knowledge, until reaching the answer to the given question. To elicit the step-by-step reasoning ability, chain-of-thought (CoT) prompting strategy [33] has been proposed for enhancing the complex reasoning capacity of LLMs. As discussed in Section 6.2, CoT involves the intermediate reasoning steps, which can be manually created [33] or automatically generated [553], into the prompts to guide LLMs to perform multi-step reasoning. Such a way largely improves the reasoning performance of LLMs, leading to new state-of-the-art results on several complex knowledge reasoning tasks [33, 56, 403]. Further, after reformulating knowledge reasoning tasks into code generation tasks, researchers have found that the performance of LLMs can be further improved [171], especially with the LLMs pre-trained on code. However, due to the complexity of knowledge reasoning tasks, the performance of current LLMs still lags behind human results on tasks such as commonsense reasoning [33, 56, 554]. As a common type of mistakes, LLMs might generate inaccurate intermediate steps, leading to a wrong final result. To address this issue, existing work has proposed special decoding or ensemble strategies to improve the accuracy of the whole reasoning chain [354, 355].

Symbolic Reasoning[50]**.** The symbolic reasoning tasks mainly focus on manipulating the symbols in a formal rule setting to fulfill some specific goal [51], where the operations and rules may have never been seen by LLMs during pre-training. Existing work [33, 357, 396] commonly evaluates LLMs on the task of last letter concatenation and coin flip, where the evaluation examples require the same reasoning steps as the in-context examples (called *in-domain test*) or more steps (called *out-of-domain test*). For an example of the out-of-domain test, LLMs could only see the examples with two words in context, but it requires LLMs to concatenate the last letters of three or more words. Typically, the accuracy of the generated symbols is adopted to evaluate the performance of LLMs on these tasks. Thus, LLMs need to understand the semantic relations among the symbolic operations and their composition in complex scenarios. However, under the out-of-domain setting, as LLMs have not seen the complex compositions of symbolic operations and rules (*e.g.*, twice the number of operations in context examples), it is hard for LLMs to capture their accurate meanings. To solve this issue, existing studies incorporate scratchpad [469, 555] and tutor [556] strategies to help LLMs better manipulate symbolic operations, for generating longer and more complex reasoning processes. Another line of research work utilizes the formal programming language to represent the symbolic operations and rules, which requires LLMs to generate code and perform the reasoning process by executing it with external interpreters. Such a way can decompose the complex reasoning process into code synthesis and program execution for LLMs and interpreters, respectively, leading to a simplified reasoning process with yet more accurate results [361].

Mathematical Reasoning. The mathematical reasoning tasks need to comprehensively utilize mathematical knowledge, logic, and computation for solving problems or generating proof statements. Existing mathematical reasoning tasks can be mainly categorized into math problem solving and automated theorem proving. For math problem solving tasks, SVAMP [471], GSM8k [470] and MATH [291] datasets are commonly used for evaluation, where LLMs need to generate accurate concrete numbers or equations to answer the mathematical problem. As these tasks also require multi-step reasoning, the CoT prompting strategy has been widely adopted for LLMs to improve the reasoning performance [33]. As another practical strategy, continually pre-training LLMs on large-scale mathematical corpora can largely boost their performance on mathematical reasoning tasks [35, 163, 557]. Further, since math problems in different languages share the same mathematical logic, researchers also propose a multilingual math word problem benchmark [401] to evaluate the multilingual mathematical reasoning capacity of LLMs. As another challenging task, automated theorem proving (ATP) [477, 479, 558] requires the reasoning model to strictly follow the reasoning logic and mathematical skills. To evaluate the performance on this task, PISA [478] and miniF2F [479] are two typical ATP datasets with the *proof success rate* as the evaluation metric. As a typical approach, existing work on ATP utilizes LLMs to aid the search for proofs using an interactive theorem prover (ITP), such as Lean, Metamath, and Isabelle [559–561]. A major limitation of ATP research is the lack of related corpora in formal language. To tackle it, several studies utilize LLMs to convert informal statements into formal proofs for augmenting new data [562] or generate drafts and proof sketches to reduce the search space of the proofs [563].

Major Issues. In spite of the advancements, LLMs still have several limitations in solving complex reasoning tasks.

- *Reasoning inconsistency*. With improved reasoning strategies (*e.g.*, CoT prompting), LLMs can solve some complex reasoning tasks, by performing step-by-step reasoning based on the supporting logic and evidence. Despite the effectiveness, the *reasoning inconsistency* issue often occurs in the decomposed reasoning process. Concretely, LLMs may generate the correct answer following an invalid reasoning path, or produce a wrong answer after a correct reasoning process [33, 360], leading to inconsistency between the derived answer and the reasoning process. To alleviate this problem, existing work has proposed to guide the whole generation process of LLMs via external tools or models [355, 369, 516], to re-check the reasoning process and final answer for correcting the potential errors [564–566] or fine-tune LLMs with process-based feedback [567, 568]. For instance, *Tree of Thoughts (ToT)* [369] empowers LLMs to engage in the decision-making process by concurrently exploring and self-evaluating various reasoning paths. To refine the reasoning processes, Self-Refine [564] elicits feedback from LLMs on self-generated solutions, enabling the iterative refinement of solutions based on the feedback. Moreover, several studies improve the consistency in the reasoning chain of LLMs through the integration of process-based supervision during training [567, 568]. As a promising solution, recent approaches reformulate the complex reasoning tasks into code generation tasks, where the strict execution of the generated code ensures the consistency between the reasoning process and the outcome. Also, it has been revealed that there might exist inconsistency between tasks with similar inputs, where small changes in the task description may cause the model to produce different results [49, 471]. To mitigate this problem, self-consistency [354] adopts the ensemble of multiple reasoning paths to enhance the decoding process of LLMs.

Reasoning Inconsistency

LLMs may generate the correct answer following an invalid reasoning path, or produce a wrong answer after a correct reasoning process, leading to inconsistency between the derived answer and the reasoning process. The issue can be alleviated by fine-tuning LLMs with process-level feedback, using an ensemble of diverse reasoning paths, and refining the reasoning process with self-reflection or external feedback.

- *Numerical computation*. For complex reasoning tasks, LLMs still face difficulties in the involved numerical com-

50. Following [33], we mainly discuss symbolic reasoning tasks specially designed for evaluating LLMs. We do not consider symbolic reasoning methods in traditional NLP tasks, such as deducing logical rules from the knowledge graphs in KBQA.

putation, especially for the symbols that are seldom encountered during pre-training, such as arithmetic with large numbers [49, 556, 569]. To tackle this issue, a direct way is to tune LLMs on synthesized arithmetic problems [287, 570]. Also, a surge of studies improve the numerical computation performance by tracing intermediate calculation steps in training and inference stages [287, 555, 571], *e.g.,* scratchpad tracing. In addition, existing work [71] has also incorporated external tools (*e.g.,* calculator), especially for handling arithmetic operations. More recently, ChatGPT has provided a plugin mechanism to use external tools [544]. In this way, LLMs need to learn how to properly manipulate the tools. For this purpose, researchers have augmented the examples using tools (even the LLM itself) for tuning the LLM [71, 572], or devised instructions and exemplars for in-context learning [361]. In addition to the aid of external tools, recent studies find that tokenizing digits into individual tokens (*e.g.,* LLaMA and Galactica tokenizers) is a useful approach to enhancing the inherent arithmetic ability of LLMs [287, 569]. One possible explanation is that subword tokenization techniques can result in inconsistent sequences when tokenizing numbers. For instance, with a subword tokenizer the integer 7481 may be tokenized as 7_481, while 74815 may be tokenized as 748_15 (the same numerical substrings with different splits) [287]. As a comparison, digit-based tokenization for numbers can avoid such an inconsistency, thus likely improving the numerical computation ability of LLMs.

Numerical Computation

LLMs face difficulties in numerical computation, especially for the symbols that are seldom encountered during pre-training. In addition to using mathematical tools, tokenizing digits into individual tokens is also an effective design choice for improving the arithmetic ability of LLMs.

7.2 Advanced Ability

In addition to the above basic evaluation tasks, LLMs also exhibit some superior abilities that require special considerations for evaluation. In this part, we discuss several representative advanced abilities and the corresponding evaluation approaches, including human alignment, interaction with the external environment, and tool manipulation. Next, we discuss these advanced abilities in detail.

7.2.1 Human Alignment

It is desired that LLMs could well conform to human values and needs, *i.e.,* human alignment, which is a key ability for the broad use of LLMs in real-world applications.

To evaluate this ability, existing studies consider multiple criteria for human alignment, such as helpfulness, honesty, and safety [46, 267, 295]. For helpfulness and honesty, adversarial question answering tasks (*e.g.,* TruthfulQA [433]) can be utilized to examine LLM's ability in detecting possible falsehood in the text [46, 72]. Furthermore, harmlessness can be also evaluated by several existing benchmarks, *e.g.,* CrowS-Pairs [482] and Winogender [483]. Despite the automatic evaluation with the above datasets, human evaluation is still a more direct way to effectively test the human alignment ability of LLMs. OpenAI invites many experts in domains related to AI risks to evaluate and improve the behaviors of GPT-4 when encountering risky contents [46]. In addition, for other aspects of human alignment (*e.g.,* truthfulness), several studies propose to use specific instructions and devise annotation rules to guide the annotation process [72]. Empirical studies have revealed that these strategies can greatly improve the human alignment ability of LLMs [267]. For instance, after alignment tuning on data collected through interactions with experts, the incorrect behavior rate of GPT-4 can be largely reduced when it deals with sensitive or disallowed prompts. In addition, high-quality pre-training data can reduce the effort required for alignment [46]. For instance, Galactica is potentially more harmless due to the less biased contents in the scientific corpus [35].

7.2.2 Interaction with External Environment

In addition to standard evaluation tasks, LLMs have the ability to receive feedback from the external environment and perform actions according to the behavior instruction, *e.g.,* generating action plans in natural language to manipulate agents [573, 574]. Such an ability is also emergent in LLMs that can generate detailed and highly realistic action plans, while smaller models (*e.g.,* GPT-2) tend to generate shorter or meaningless plans [573].

To test this ability, several embodied AI environments and benchmarks can be used for evaluation, described as follows. VirtualHome [485] builds a 3D simulator for household tasks such as cleaning and cooking, in which the agent can execute natural language actions generated by LLMs. ALFRED [487] includes more challenging tasks that require LLMs to accomplish compositional targets. BEHAVIOR [486] focuses on everyday chores in simulation environments and requires LLMs to generate complex solutions, *e.g.,* changing the internal status of objects. Apart from restricted environments such as household tasks, a line of research work investigates the proficiency of LLM-based agents to explore open-world environments, such as Minecraft and the Internet [575, 576]. Voyager [576] introduces an automatic curriculum module that enables LLMs to continuously acquire new skills based on feedback from the environment. GITM [575] focuses on solving various challenges in Minecraft based on LLM, through task decomposition, planning, and invocation of interfaces. Based on the generated action plans or task completions, existing work either adopts the regular metrics (*e.g.,* executability and correctness of the generated action plans) [573] in the benchmark or directly conducts real-world experiments and measures the success rate [577], to evaluate such ability. It has been shown that LLMs are capable in interacting with the external environment and generating accurate action plans [578]. Recently, several improvement methods have been proposed to enhance the interaction ability of LLMs, *e.g.,* designing code-like prompts [407] and providing real-world grounding [577].

In addition, recent work also explores multi-agent collaboration based on LLMs in simulated environments [410, 579, 580]. These studies simulate human social behaviors

by instantiating multiple LLM-based agents with observations, planning, and memories in a sandbox environment. In controlled evaluation, the abilities of generative agents to search, plan, and think are evaluated by humans in an interview-like manner. Further, they also conduct descriptive measurements on multiple agents within a simulated environment to examine emergent social behaviors.

7.2.3 Tool Manipulation

When solving complex problems, LLMs can turn to external tools if they determine it is necessary. By encapsulating available tools with API calls, existing work has involved a variety of external tools, *e.g.,* search engine [72], calculator [71], and compiler [361], to enhance the performance of LLMs on several specific tasks. Recently, OpenAI has supported the use of plugins in ChatGPT [544], which can equip LLMs with broader capacities beyond language modeling. For example, the web browser plugin enables ChatGPT to access fresh information. Further, incorporating third-party plugins is particularly key for creating a prosperous ecosystem of applications based on LLMs.

To examine the ability of tool manipulation, existing work mostly adopts complex reasoning tasks for evaluation, such as mathematical problem solving (*e.g.,* GSM8k [470] and SVAMP [471]) or knowledge question answering (*e.g.,* TruthfulQA [433]), where the successful utilization of tools is very important for enhancing the required skills that LLMs are incapable in (*e.g.,* numerical calculation). In this way, the evaluated performance on these tasks can reflect the ability of LLMs in tool manipulation. To teach LLMs to utilize tools, existing studies add exemplars using tools in context to elicit LLMs [361], or fine-tune LLMs on simulated data about tool utilization [71, 572]. It has been found that with the help of tools, LLMs become more capable of handling the issues that they are not good at, *e.g.,* equation calculation and answering timely questions [71, 366]. However, as the number of available tools increases, the limited context length of LLMs may pose challenges in describing and demonstrating extensive tool APIs. To address this issue, existing work retrieves the usage of relevant tools, or encoding tool information as tokens within the embedding space [581–583].

In addition to existing tools developed by humans, LLMs possess the capability to make their own tools for specific tasks autonomously [584]. This enables the models to independently explore and manipulate these self-created tools, thereby expanding their potential for autonomous exploration in solving a wide range of real-world tasks.

Summary. The above three abilities are of great value to the practical performance of LLMs: conforming to human values and preferences (human alignment), acting properly in real-world scenarios (interaction with the external environment), and expanding the ability scope (tool manipulation). In addition to the above three advanced abilities, LLMs might also show other abilities that are specially related to some tasks (*e.g.,* data annotation [378]) or learning mechanisms (*e.g.,* self-improvement [585]). It will be an open direction to discover, measure and evaluate these newly emerging abilities, so as to better utilize and improve LLMs.

7.3 Benchmarks and Evaluation Approaches

In the above, we have discussed the basic and advanced abilities of LLMs. Next, we will introduce existing evaluation benchmarks and approaches [612, 613].

7.3.1 Comprehensive Evaluation Benchmarks

Recently, several comprehensive benchmarks [291, 292, 398] have been released for the evaluation of LLMs. In this part, we introduce several widely used benchmarks, *i.e.,* MMLU, BIG-bench, HELM, and a series of human exam benchmarks.

• *MMLU* [291] is a versatile benchmark for large-scale evaluation of multi-task knowledge understanding, covering a wide range of knowledge domains from mathematics and computer science to humanities and social sciences. The difficulties of these tasks vary from basic to advanced. As shown in existing work, LLMs mostly outperform small models by a substantial margin on this benchmark [35, 56, 57, 64], which shows the scaling law in model size. More recently, GPT-4 achieves a remarkable record (86.4% in 5-shot setting) in MMLU, which is significantly better than the previous state-of-the-art models [46].

• *BIG-bench* [292] is a collaborative benchmark intended to probe existing LLMs from various aspects. It comprises 204 tasks that encompass a broad range of topics, including linguistics, childhood development, mathematics, commonsense reasoning, biology, physics, social bias, software development, and so on. By scaling the model size, LLMs can even outperform the average human performance under the few-shot setting on 65% of tasks in BIG-bench [56]. Considering the high evaluation cost of the entire benchmark, a lightweight benchmark BIG-bench-Lite has been proposed, which contains 24 small yet diverse and challenging tasks from BIG-bench. Additionally, the BIG-bench hard (BBH) benchmark [614] has been proposed to concentrate on investigating the currently unsolvable tasks of LLMs by selecting the challenging tasks in which LLMs exhibit inferior performance compared to humans. Since BBH becomes more difficult, small models mostly achieve performance close to random. As a comparison, CoT prompting can elicit the abilities of LLMs to perform step-by-step reasoning for enhancing the performance, even exceeding the average human performance in BBH.

• *HELM* [398] is a comprehensive benchmark that currently implements a core set of 16 scenarios and 7 categories of metrics. It is built on top of many prior studies, conducting a holistic evaluation of language models. As shown in the experimental results of HELM, instruction tuning can consistently boost the performance of LLMs in terms of accuracy, robustness, and fairness. Further, for reasoning tasks, the LLMs that have been pre-trained on the code corpus show superior performance.

• *Human-level test benchmarks* aim to evaluate the comprehensive ability of LLMs with questions designed for testing humans, such as AGIEval [587], MMCU [588], M3KE [589], C-Eval [590] and Xiezhi [591]. These benchmarks encompass a wide range of domains, difficulty levels, and languages to provide a comprehensive evaluation of LLMs' general capabilities. Compared to publicly available models, models offering API services (*e.g.,* GPT-4, ChatGPT, Claude) demon-

TABLE 11: A category of existing evaluation work. "General" denotes that the evaluation focuses on an overall performance of multiple abilities. The evaluated abilities are not limited to the representative basic and advanced abilities mentioned in Section 7.1 and 7.2.

Method	Evaluation	Model Types	Abilities/Domain	Data Source
Benchmark	MMLU [291]	Base/Fine-tuned/Specialized	General	Human exam/practice
	BIG-bench [292]	Base/Fine-tuned/Specialized	General	Human annotation
	HELM [398]	Base/Fine-tuned/Specialized	General	Benchmark collection
	Open LLM Leaderboard [586]	Base/Fine-tuned/Specialized	General	Benchmark collection
	AGIEval [587]	Base/Fine-tuned/Specialized	General	Human exam/practice
	MMCU [588]	Base/Fine-tuned/Specialized	General	Human exam/practice
	M3KE [589]	Base/Fine-tuned/Specialized	General	Human exam/practice
	C-Eval [590]	Base/Fine-tuned/Specialized	General	Human exam/practice
	Xiezhi [591]	Base/Fine-tuned/Specialized	General	Human exam/practice
	OpenCompass [592]	Base/Fine-tuned/Specialized	General	Benchmark collection
	Chain-of-Thought Hub [593]	Base/Fine-tuned	General	Benchmark collection
	KoLA [594]	Base/Fine-tuned	Knowledge utilization	Web
	ARB [595]	Fine-tuned	Complex reasoning	Human exam/practice
	APIBench [596]	Base/Fine-tuned	Tool manipulation	Web
	APIBank [597]	Fine-tuned	Tool manipulation	Synthesis
	ToolAlpaca [598]	Base/Fine-tuned	Tool manipulation	Synthesis
	T-Bench [599]	Fine-tuned	Tool manipulation	Synthesis
	ToolBench [600]	Fine-tuned	Tool manipulation	Synthesis
	BOLAA [601]	Base/Fine-tuned	Environment interaction	Benchmark collection
	AgentBench [602]	Base/Fine-tuned	Environment interaction	Human annotation/Synthesis
	HaluEval [481]	Base/Fine-tuned	Human alignment	Human annotation/Synthesis
	PromptBench [603]	Base/Fine-tuned	Robustness	Benchmark collection
	HumanEval [92]	Base/Fine-tuned/Specialized	Code synthesis	Human annotation
	MultiMedQA [282]	Specialized	Healthcare	Benchmark collection
	FLUE [604]	Specialized	Finance	Benchmark collection
	LegalBench [605]	Specialized	Legal	Human annotation
Human	Chatbot Arena [606]	Base/Fine-tuned/Specialized	Human Alignment	Human annotation
	SciBench [607]	Fine-tuned	Complex reasoning	Human exam/practice
Model	AlpacaEval [608]	Fine-tuned	Instruction following	Synthesis
	MT-bench [606]	Fine-tuned	Human alignment	Human annotation
	TrustGPT [609]	Base/Fine-tuned	Human alignment	Benchmark collection
	LMExamQA [610]	Base/Fine-tuned	Knowledge utilization	Synthesis
	ChatEval [611]	Base/Fine-tuned	Knowledge utilization	Benchmark collection

strate superior performance compared to publicly available models on these evaluation benchmarks. As the best-performing model in evaluations, GPT-4 surpasses average human performance in AGIEval [587]. However, it still lags behind the top human performance on these challenging benchmarks. Hence, there remains ample room for further enhancements in the overall abilities of LLMs, particularly for publicly accessible models.

The above benchmarks cover a variety of mainstream evaluation tasks and real-world human exam questions for the evaluation of LLMs. Also, there are several benchmarks that focus on evaluating specific abilities of LLMs, such as TyDiQA [615] for multilingual knowledge utilization and MGSM [401] for multilingual mathematical reasoning. To conduct the evaluation, one can select suitable benchmarks according to specific goals. In addition, there are also several open-source evaluation frameworks for researchers to evaluate LLMs on existing benchmarks or extend new tasks for customized evaluations, such as Language Model Evaluation Harness [616] and OpenAI Evals [46]. Further, some researchers also construct continuously updated leaderboards by aggregating representative benchmarks, to compare the performance of existing LLMs, such as Open LLM Leaderboard [586]. The above benchmarks and leaderboards provide important references to demonstrate the basic and advanced abilities of LLMs. We will give more deep discussions on pros and cons on evaluation approaches in Section 7.3.2.

7.3.2 Evaluation Approaches

After introducing existing benchmarks, in this part, we will review existing evaluation approaches for assessing the performance of LLMs. To organize our discussion, we categorize LLMs into three different types: *base LLMs* (pre-trained model checkpoints), *fine-tuned LLMs* (instruction or alignment fine-tuned model checkpoints), and *specialized LLMs* (adapted model checkpoints for some specific task or domain). Here, we keep both fine-tuned LLMs and specialized LLMs, to distinguish the different purposes of LLMs: general or specific task solvers. To evaluate the three types of LLMs, we can test the LLM's performance related to different abilities (*e.g.,* basic or advanced abilities as discussed in Section 7.1 and 7.2). In general, there are three main approaches to evaluating LLMs, namely benchmark-based approach [291], human-based approach [606], and model-based approach [608]. Table 11 shows an illustration of the relationship among LLM type, evaluation approach, and tested abilities. Next, we will discuss the evaluation approaches for different types of LLMs.

Evaluation of Base LLMs. Base LLMs refer to the model checkpoints obtained right after pre-training. For base LLMs, we mainly focus on examining the basic abilities (Section 7.1), such as complex reasoning and knowledge

utilization. Since most of these basic abilities can be assessed with well-defined tasks, benchmark-based approaches have been widely used to evaluate base LLMs. Next, we will introduce common evaluation benchmarks and evaluation procedures for base LLMs.

• *Common benchmarks.* To evaluate base LLMs, typical benchmarks are designed in the form of close-ended problems like multiple-choice questions. These commonly used benchmarks can be mainly divided into two categories: knowledge-oriented and reasoning-oriented benchmarks. Knowledge-oriented benchmarks (*e.g.,* MMLU [291] and C-Eval [590]) aim to evaluate the capacity of world knowledge, while reasoning-oriented benchmarks (*e.g.,* GSM8K [523], BBH [614], and MATH [291]) focus on evaluating the capability of solving complex reasoning tasks. Further, some recently proposed benchmarks (*e.g.,* OpenCompass [592]) combine these two types for a comprehensive comparison.

• *Benchmark based evaluation procedure.* To perform the benchmark evaluation, each problem will first be formatted into a prompt for LLMs to generate the result text. Then, the generated result text will be parsed with human-written rules to get the predicted answer. Finally, the performance of LLMs can be automatically calculated using standard metrics like accuracy by comparing the predicted answer with the ground-truth one. The evaluation approach can be conducted in either the few-shot or zero-shot setting, which might lead to different evaluation results or rankings. Since base LLMs have not been instruction fine-tuned (with relatively weak task generalization ability), the few-shot setting is often more suitable for evaluation. For some complex reasoning tasks, CoT prompts also need to be used to fully exhibit the capacity during evaluation. Another note is that this evaluation approach can also be applied to assess the abilities of fine-tuned LLMs. Actually, several leaderboards (*e.g.,* Open LLM Leaderboard [586]) are built upon this approach, evaluating both base and fine-tuned LLMs.

Evaluation of Fine-tuned LLMs. Fine-tuned LLMs in this part refer to the model checkpoints obtained after instruction tuning or alignment tuning based on pre-trained model weights[51]. Typically, fine-tuned LLMs will be tested on various abilities (*e.g.,* knowledge utilization and human alignment), and thus it is common that they are assessed with multiple evaluation approaches. In addition to benchmark-based evaluation, human-based and model-based approaches have also been widely used to evaluate the advanced abilities of fine-tuned LLMs. Next, we will introduce the two evaluation methods.

• *Human-based evaluation.* Unlike automatic evaluation for basic abilities, human evaluation typically considers more factors or abilities in real-world use, such as human alignment and tool manipulation. In this evaluation approach, test tasks are usually in the form of open-ended questions, and human evaluators are invited to make judgments on the quality of answers generated by LLMs. Typically, there are two main types of scoring methods for human evaluators: pairwise comparison and single-answer grading. In pairwise comparison, given the same question, humans are assigned two answers from different models to determine which one is better, while in single-answer grading, they only need to score a single answer at a time. For example, HELM [398] employs humans to perform single-answer grading on summarization and disinformation tasks, while Chatbot Arena [606] constructs a crowdsourcing platform that allows users to engage in conversations with two anonymous chat LLMs and report pairwise comparison results.

• *Model-based evaluation.* Since human-based evaluation is both expensive and time-consuming, some work has proposed leveraging powerful closed-source LLMs such as ChatGPT and GPT-4 as a surrogate for human evaluators [606, 608]. For example, AlpacaEval [608] collects a set of instructions and utilizes a capable LLM (*e.g.,* GPT-4) as the judge to perform pair-wise comparisons against the reference outputs. Furthermore, MT-bench [606] collects a set of multi-turn questions for evaluation and improves the reliability of LLM-based evaluators through methods like ICL and CoT. Compared with human evaluators, LLMs such as ChatGPT and GPT-4 can achieve high agreement with humans, in both small-scale handcrafted and large-scale crowdsourced evaluation tasks. Despite this, these closed-source LLMs are limited in access and have the potential risk of data leakage. To address this, recent work [606] has explored fine-tuning open-source LLMs (*e.g.,* Vicuna [124]) as model evaluators using scoring data from human evaluators, which has narrowed the gap with powerful closed-source LLMs (*e.g.,* GPT-4).

Evaluation of Specialized LLMs. Specialized LLMs refer to the model checkpoints specially adapted to some domains or applications like healthcare [282] and finance [617]. As special task solvers, specialized LLMs will be tested not only on general abilities (*e.g.,* basic ability like complex reasoning and advanced ability like human alignment), but also on specific abilities related to their designated domains or applications. For this purpose, one often needs to construct specific benchmarks tailored for the target domains or applications. Then, these domain-specific benchmarks can be combined with general benchmarks to conduct both comprehensive and targeted evaluation for specialized LLMs. For example, MultiMedQA [282] is a specific benchmark in healthcare, which includes medical examinations and healthcare questions. In this work [282], MultiMedQA has been combined with MMLU [291] to assess the performance of specialized LLMs for healthcare, such as Med-PaLM [282]. Similarly, FLUE [617] constructs a benchmark for finance, spanning from financial sentiment analysis to question answering. It has been used collaboratively with BBH [614] to evaluate finical LLMs like BloombergGPT [286].

Pros and Cons of Different Evaluation Approaches. In the above, we have discussed different evaluation approaches to assess the abilities of LLMs. Next, we simply analyze the pros and cons of each evaluation approach.

• *Benchmark-based approach*. This evaluation approach can leverage existing benchmarks for assessing the performance of LLMs. The tasks involved in these benchmarks often contain sufficient test samples to measure the core abilities (*e.g.,* reasoning). The whole evaluation procedure can be

51. In some cases, it is also called *chat models*.

(almost) automatic, and it is convenient to carry out test experiments for various base LLMs, especially useful for monitoring the performance of model checkpoints during pre-training. However, LLMs are often sensitive to the evaluation settings, including the question prompts, zero-shot or few-shot tests, and the answer parsing methods. Thus, one should take possible influencing factors into consideration when conducting the evaluation experiments. The evaluation results should be noted with the adopted evaluation settings. Another issue is the data contamination [56], *i.e.,* the test data itself or relevant content has been contained in the pre-training corpora. This phenomenon has become increasingly severe since more and more open data has been collected for developing LLMs.

• *Human-based approach*. Human evaluation offers several advantages when assessing the capabilities of LLMs to solve real-world tasks. One of the key benefits is its ability to directly reflect the actual abilities of LLMs. Based on feedback and experiences from real users, human evaluation provides a more direct measure of LLMs' performance in real-world scenarios. Further, it can conduct more flexible and diverse evaluation tasks based on human evaluators. For instance, users can submit various queries and test the abilities of LLMs according to their own task cognition. It allows for a deep understanding of the strengths and weaknesses of LLMs across different types of tasks and contexts. However, human evaluation also has inherent limitations that could potentially affect its accuracy and consistency. Factors such as personalized tastes and varying education levels among evaluators can introduce biases or even inconsistencies in the evaluation process. In some cases, users' judgments are likely to be subjective, which may not reflect the true capabilities of the LLMs. Moreover, conducting robust and reliable human evaluations often requires a large number of evaluators, which can be very expensive and time-consuming. In addition, human evaluation is often not reproducible, making it infeasible to extend existing evaluation results or track the progress of LLMs.

• *Model-based approach*. As a surrogate for human-based approaches, model-based approaches serve to diminish the reliance on human involvement, and enable more efficient and scalable evaluation. In addition, LLMs can provide meaningful explanations for the assigned rating scores, thereby enhancing the interpretability of evaluations. Despite their scalability and explanability, model-based approaches have been found to suffer from several issues, including position, verbosity, and self-enhancement bias [606]. Specially, position bias (*i.e.,* the order to present the responses) refers to the fact that LLMs tend to assign high scores for the answers at specific positions over others, verbosity bias means that LLMs favor verbose answers even if they are short in quality compared with shorter answers, and self-enhancement bias indicates that LLMs often overrate in their own generations. In addition, since LLMs have limited capacities in solving complex reasoning problems, they cannot serve as qualified evaluators for some difficult tasks (*e.g.,* mathematical reasoning). These limitations can be mitigated to some extent by specific prompt engineering and fine-tuning strategies [606].

To summarize, our categorization (Table 11) of existing work on LLM evaluation is mainly based on two major dimensions, namely evaluation methodology and model type, which are further extended with the test abilities. There are some recent work [612, 613] that also has discussed the categorization or taxonomies of existing work for LLM evaluation.

7.4 Empirical Evaluation

The above evaluation benchmarks and approaches are mainly employed to evaluate the overall abilities of LLMs. In this part, we conduct a fine-grained evaluation of the abilities discussed in Section 7.1 and Section 7.2. For each kind of ability, we select representative tasks and datasets for conducting evaluation experiments to examine the corresponding performance of LLMs.

7.4.1 Experimental Settings

In this part, we introduce the experimental settings for our evaluation.

Evaluation Models. To conduct the evaluation, we consider representative LLMs from open-source models to closed-source API-accessing models as follows:

• *Open-source models.* Existing open-source models can be categorized into base models and instruction-tuned models. Base models are only pre-trained on a large general-purpose corpus with the language modeling objective, but without further supervised fine-tuning. In our evaluation, we select four representative base models including LLaMA (7B) [57], LLaMA 2 (7B) [90], Pythia (7B and 12B) [87], and Falcon (7B) [626][52]. Instruction-tuned models are those fine-tuned using instructions (*i.e.,* task datasets, daily chat, or synthetic instructions). In our experiments, we select four representative instruction-tuned models including Vicuna (7B and 13B) [124], Alpaca (7B) [123], and ChatGLM (6B) [84]. In addition, we also include LLaMA 2-Chat (7B) [90] for comparison, and it is a representative model that has been aligned with human via instruction tuning and RLHF, based on LLaMA 2 (7B).

• *Closed-source models.* In addition to the open-source models, there are also closed-source models that can only be accessed via APIs, which have gained much attention from both developers and researchers. Here, we select four representative closed-source models including text-davinci-002/003 (short as *Davinci002/003*), ChatGPT, Claude, and Claude 2, where the first three models are developed by OpenAI and the other two are developed by Anthropic.

Tasks and Datasets. Next, we set up the evaluation tasks and datasets for the abilities discussed in Section 7.1 and Section 7.2. We mainly evaluate the zero-shot performance of LLMs on these datasets. For more complex tasks that are hard to be solved in the zero-shot manner (*e.g.,* mathematical reasoning and tool manipulation), we mainly report the 3-shot performance, considering the context length limit of open-source models.

• *Language generation.* As discussed before, for language generation, we consider evaluating three kinds of tasks, *i.e.,* language modeling, conditional text generation, and

52. Experiments with larger models are still in schedule due to the limit of computational resources.

TABLE 12: Evaluation on the eight abilities of LLMs with specially selected tasks. The shade of the Orange and Blue fonts denote the performance orders of the results in closed-source and open-source models, respectively. This table will be continuously updated by incorporating the results of more models.

Models	Language Generation				Knowledge Utilization				
	LBD↑	WMT↑	XSum↑	HumanEval↑	TriviaQA↑	NaturalQ↑	WebQ↑	ARC↑	WikiFact↑
ChatGPT	55.81	36.44	21.71	79.88	54.54	21.52	17.77	93.69	29.25
Claude	64.47	31.23	18.63	51.22	40.92	13.77	14.57	66.62	34.34
Claude 2	45.20	12.93	19.13	78.04	54.30	21.30	21.06	79.97	35.83
Davinci003	69.98	37.46	18.19	67.07	51.51	17.76	16.68	88.47	28.29
Davinci002	58.85	35.11	19.15	56.70	52.11	20.47	18.45	89.23	29.15
LLaMA 2-Chat (7B)	56.12	12.62	16.00	11.59	38.93	12.96	11.32	72.35	23.37
Vicuna (13B)	62.45	20.49	17.87	20.73	29.04	10.75	11.52	20.69	28.76
Vicuna (7B)	63.90	19.95	13.59	17.07	28.58	9.17	6.64	16.96	26.95
Alpaca (7B)	63.35	21.52	8.74	13.41	17.14	3.24	3.00	49.75	26.05
ChatGLM (6B)	33.34	16.58	13.48	13.42	13.42	4.40	9.20	55.39	16.01
LLaMA 2 (7B)	66.39	11.57	11.57	17.07	30.92	5.15	2.51	24.16	28.06
LLaMA (7B)	67.68	13.84	8.77	15.24	34.62	7.92	11.12	4.88	19.78
Falcon (7B)	66.89	4.05	10.00	10.37	28.74	10.78	8.46	4.08	23.91
Pythia (12B)	61.19	5.43	8.87	14.63	15.73	1.99	4.72	11.66	20.57
Pythia (7B)	56.96	3.68	8.23	9.15	10.16	1.77	3.74	11.03	15.75

Models	Knowledge Reasoning			Symbolic Reasoning		Mathematical Reasoning		Interaction with Environment	
	OBQA↑	HellaSwag↑	SocialIQA↑	C-Objects↑	Penguins↑	GSM8k↑	MATH↑	ALFW↑	WebShop↑
ChatGPT	81.20	61.43	73.23	53.20	40.27	78.47	33.78	58.96	45.12/15.60
Claude	81.80	54.95	73.23	59.95	47.65	70.81	20.18	32.09	50.02/30.40
Claude 2	71.60	50.75	58.34	66.76	74.50	82.87	32.24		34.96/19.20
Davinci003	74.40	62.65	69.70	64.60	61.07	57.16	17.66	65.67	64.08/32.40
Davinci002	69.80	47.81	57.01	62.55	67.11	49.96	14.28	76.87	29.66/15.20
LLaMA 2-Chat (7B)	45.62	74.01	43.84	43.40	38.93	9.63	2.22	11.19	24.51/5.60
Vicuna (13B)	43.65	70.51	45.97	53.55	36.91	18.50	3.72	8.96	22.74/5.00
Vicuna (7B)	43.84	69.25	46.27	44.25	36.24	14.03	3.54	1.49	6.90/1.40
Alpaca (7B)	47.82	69.81	47.55	39.35	40.27	4.93	4.16	4.48	0.00/0.00
ChatGLM (6B)	30.42	29.27	33.18	14.05	14.09	3.41	1.10	0.00	0.00/0.00
LLaMA 2 (7B)	44.81	74.25	41.72	43.95	35.75	10.99	2.64	8.96	0.00/0.00
LLaMA (7B)	42.42	73.91	41.46	39.95	34.90	10.99	3.12	2.24	0.00/0.00
Falcon (7B)	39.46	74.58	42.53	29.80	24.16	1.67	0.94	7.46	0.00/0.00
Pythia (12B)	37.02	65.45	41.53	32.40	26.17	2.88	1.96	5.22	3.68/0.60
Pythia (7B)	34.88	61.82	41.01	29.05	27.52	1.82	1.46	7.46	10.75/1.80

Models	Human Alignment					Tool Manipulation			
	TfQA↑	C-Pairs↓	WinoGender↑	RTP↓	HaluEval↑	HotpotQA↑	Gorilla-TH↑	Gorilla-TF↑	Gorilla-HF↑
ChatGPT	69.16	18.60	62.50/72.50/79.17	3.07	66.64	23.80	67.20	44.53	19.36
Claude	67.93	32.73	71.67/55.00/52.50	3.75	63.75	33.80	22.04	7.74	7.08
Claude 2	71.11	10.67	60.00/60.00/55.83	3.20	50.63	36.4	61.29	22.19	23.67
Davinci003	60.83	0.99	67.50/68.33/79.17	8.81	58.94	34.40	72.58	3.80	6.42
Davinci002	53.73	7.56	72.50/70.00/64.17	10.65	59.67	26.00	2.69	1.02	1.00
LLaMA 2-Chat (7B)	69.77	48.54	47.50/46.67/46.67	4.61	43.82	4.40	0.00	0.00	0.22
Vicuna (13B)	62.30	45.95	50.83/50.83/52.50	5.00	49.01	11.20	0.00	0.44	0.89
Vicuna (7B)	57.77	67.44	49.17/49.17/49.17	4.70	43.44	6.20	0.00	0.00	0.33
Alpaca (7B)	46.14	65.45	53.33/51.67/53.33	4.78	44.16	11.60	0.00	0.00	0.11
ChatGLM (6B)	63.53	50.53	47.50/47.50/46.67	2.89	41.82	4.00	0.00	0.00	0.00
LLaMA 2 (7B)	50.06	51.39	48.83/48.83/50.83	6.17	42.23	3.80	0.00	0.00	0.11
LLaMA (7B)	47.86	67.84	54.17/52.50/51.67	5.94	14.18	1.60	0.00	0.00	0.11
Falcon (7B)	53.24	68.04	50.00/50.83/50.00	6.71	37.41	1.00	0.00	0.00	0.00
Pythia (12B)	54.47	65.78	49.17/48.33/49.17	6.59	27.09	0.40	0.00	0.00	0.00
Pythia (7B)	50.92	64.79	51.67/49.17/50.00	13.02	25.84	0.20	0.00	0.00	0.00

code synthesis. Specially, we select four commonly-used datasets, namely LAMBADA [188] (language modeling), WMT'22 [421] (machine translation), XSum [425] (text summarization), and HumanEval [92] (code synthesis) for evaluation. In WMT'22, we construct a new evaluation set by selecting 1000 examples for each language pair from the original large-scale test set to examine the average performance of LLMs in machine translation. We evaluate the zero-shot performance of LLMs on these datasets, and compute the *accuracy* of predicting words for LAMBADA, *BLEU-4* for WMT'22, *ROUGE-L* for XSum, and *pass*@10 for HumanEval.

• *Knowledge utilization.* To evaluate the ability of knowledge utilization, we select four question answering datasets (*i.e.,* TriviaQA [435], Natural Questions [431], Web Questions [434], and ARC [432]), and a fact extraction dataset, WikiFact [448]. We also report the zero-shot performance of LLMs on these datasets, and compute *accuracy* for ARC and *exact match* for other datasets.

• *Complex reasoning.* For complex reasoning, we evaluate the comparison models on OpenbookQA [443], HellaSwag [460], and SocialIQA [459] for knowledge reason-

TABLE 13: Prompt examples and their performance of ChatGPT on representative tasks. For most tasks, we compare the performance for *simple* and *complex* prompts. We also present the reported performance of supervised methods. "LG", "KU", "CR", "SDG", "IR" are short for "language generation", "knowledge utilization", "complex reasoning", "structured data generation", "information retrieval". "-" means there is no reported supervised result previously on this dataset.

	Tasks	Datasets	Instructions	ChatGPT	Supervised
LG	Translation	WMT	`I want you to act as a translator. Please translate the English sentence into Czech.`	20.66	41.40 [618]
			`I want you to act as a translator. Translate the given English sentence into Czech, and ensure that the translated sentence is semantically consistent with the given sentence. \n Sentence: {source sentence} \n Translation:`	21.12	
	Summarization	XSum	`Please generate a one-sentence summary for the given document.`	21.71	42.08 [619]
			`{document} Try your best to summarize the main content of the given document. And generate a short summary in 1 sentence for it.\n Summary:`	23.01	
KU	Closed-Book QA	ARC	`Choose your answer to the question. {query} {options}`	85.19	92.00 [620]
			`Choose a correct answer according to the given question, and output the corresponding id, do not answer other content except the answer id.`	85.86	
	Open-Book QA	OBQA	`Choose your answer to the question: {question} {choices}. You must only output A, B, C, or D without any extra explanation. The answer is`	81.20	87.20 [620]
			`Following is a question that requires multi-step reasoning, use of additional common and commonsense knowledge, and rich text comprehension. Choose your answer to the question: \n Question: Frilled sharks and angler fish live far beneath the surface of the ocean, which is why they are known as \n Choices: \n A. Deep sea animals \n B. fish \n C. Long Sea Fish \n D. Far Sea Animals \n You must only output A, B, C, or D without any extra explanation. The answer is`	82.20	
	Fact Extraction	WikiF	`Complete the sentence with one or a few words.`	29.25	34.20 [398]
			`Complete the given sentence with one entity name in Wikipedia (MUST be a noun) as short as possible, and ensure that the completed sentence conforms to the facts.`	31.21	
CR	Symbolic Reasoning	C-Objects	`Problem: {problem}\n Answer:`	53.20	—
			`You are an expert in reasoning problem. Here are some examples about symbolic reasoning. You can use the knowledge in examples and solve the last problem. You should follow the examples and generate the final answer without external solution or words.`	66.75	
	Math Word Problems	GSM8k	`Problem: {problem}\n Solution: Let's think step by step.`	78.47	63.20 [621]
			`Let's use python to solve math problems. Here are three examples how to do it,\n Q: Olivia has $23. She bought five bagels for $3 each. How much money does she have left?\n```def solution():\n """Olivia has $23. She bought five bagels for $3 each. How much money does she have left?"""\n money_initial = 23\n bagels = 5\n bagel_cost = 3\n money_spent = bagels * bagel_cost\n money_left = money_initial - money_spent\n result = money_left\n return result```\n ..... \n How about this question?\n Q:`	79.30	
SDG	Code Synthesis	HumanEval	`I want you act as a code completer. Given a code snippet, your objective is to complete the code and ensure that it can achieve the described functionality.`	79.88	48.20 [622]
	Text-to-SQL	Spider	`### Complete sqlite SQL query only and with no explanation.\n #\n### Sqlite SQL tables, with their properties: \n#\n{table}\n# {foreign_key}\n#\n### {question}\n SELECT`	70.10	84.10 [623]
IR	Recommendation	MovieLens	`I've watched the following movies in the past in order: \n {user_his_text} \n\n Now there are {recall_budget} candidate movies that I can watch next: \n {candidate_text_order} \n Please rank these {recall_budget} movies by measuring the possibilities that I would like to watch next most, according to my watching history. Please think step by step. \n Note that my most recently watched movie is {recent_item}. Please show me your ranking results with order numbers. Split your output with line break. You MUST rank the given candidate movies. You can not generate movies that are not in the given candidate list.`	48.80	76.25 [624]
	Conversational Recommendation	ReDial	`Recommend 10 items that are consistent with user preference. The recommendation list can contain items that the dialog mentioned before. The format of the recommendation list is: no. title (year). Don't mention anything other than the title of items in your recommendation list`	17.20	25.60 [625]

ing; Colored Objects [292] and Penguins in the Table [292] for symbolic reasoning; GSM8k [470] and MATH [291] for mathematical reasoning. We compute the *accuracy* for OpenbookQA, HellaSwag, and SocialIQA; *solve rate* for Colored Objects and Penguins in the Table; and *accuracy* for GSM8k and MATH. For knowledge reasoning tasks, we evaluate the zero-shot performance, since they are all QA tasks that can be solved in a zero-shot setting. For complex symbolic reasoning and mathematical reasoning tasks, we leverage 3-shot in-context exemplars to better elicit LLMs to accomplish them. Following existing work [33, 361], we also utilize the chain-of-thought prompting strategy for better solving the mathematical reasoning tasks.

• *Human alignment.* For human alignment, we select TruthfulQA [433] to measure whether a LLM is truthful in generating answers to questions, CrowS-Pairs [482] and WinoGender [483] to assess the stereotypes in LLMs, RealToxityPrompts [484] to evaluate the extent to which LLMs generate toxic language, and HaluEval [481] to test the ability of LLMs to recognize hallucination. As the test set of Real-Toxicity-Prompts is too large, we randomly sample 10000 examples from it for evaluation. We follow LLaMA [57] to report the zero-shot performance, and compute the *accuracy* of identifying a claim as true for TruthfulQA, *accuracy* of recognizing biased sentences (high perplexity) for CrowS-Pairs, *coreference resolution accuracy (he/she/they)* for WinoGender, *toxicity score* for RealToxityPrompts, and *average accuracy* of recognizing hallucinations for HaluEval. For TruthfulQA, we follow existing work [57] that utilizes text-davinci-003 to replace humans for scoring. For Crows-Pairs and WinoGender, we follow the experimental settings of LLaMA [57] to compute the perplexity and coreference resolution score. For RealToxityPrompts, we utilize the Perspective-API[53] for toxicity evaluation.

• *Interaction with environment.* To test this ability, we select ALFWorld [488] and WebShop [489] for evaluation, which simulate real-world scenarios such as household and e-commerce environments. We follow the setting of ReAct [367] that evaluate the 1-shot and 2-shot performance of LLMs on WebShop and ALFWorld respectively, and compute *success rate* for ALFWorld and *average score/success rate* for WebShop. Further, we also follow ReAct [367] to reduce the length of the input prompt and utilize line break as the EOS token.

• *Tool manipulation.* For tool manipulation, we consider two kinds of tools including search engine and model interfaces. Therefore, we adopt two tool manipulation benchmarks, *i.e.,* HotpotQA [457] and Gorilla [496]. HotpotQA requires LLMs to use search engine to retrieve documents from the web, and Gorilla to invoke model APIs from three hubs of TorchHub, TensorHub and HuggingFace. We compute *exact match* for HotpotQA and *accuracy* for Gorilla. For HotpotQA, we follow ReAct [367] to report the 3-shot performance. For Gorilla, we follow the code released by its paper [496], and evaluate the zero-shot performance.

Implementation Details. For each task and dataset, we evaluate the compared LLMs using the same prompts and results parsing method provided by existing work (*i.e.,* TruthfulQA, HotPotQA, Gorilla, HaluEval) or designed according to our empirical experience (*i.e.,* TriviaQA, Natural Questions, Web Questions, ARC, WikiFact, GSM8k, MATH, C-Objects, Penguins, LAMBADA, WMT'22, XSum, HumanEval, CrowS-Pairs, WinoGender, RealToxityPrompt). Specifically, all the experiments about closed-source models are based on invoking their official APIs, while for open-source models, we utilize their publicly available code and model parameters, and perform the inference on 8 A800-80G GPUs. For TriviaQA, OpenbookQA, HellaSwag, and SocialIQA, we experiment on the development set since the test set is not publicly released. While for other datasets, we experiment on the test set. To reproduce our experiments, we also publicly release our experimental code and data in https://github.com/RUCAIBox/LLMSurvey/tree/main/Experiments.

53. https://perspectiveapi.com/

7.4.2 Results Analysis and Findings

We report the experimental results in Table 12, and analyze the results in the following.

Analysis of Closed-Source Models. We summarize our analysis and findings of the four closed-source models (*i.e.,* ChatGPT, Claude, Davinci003 and Davinci002) as follows:

• *These five closed-source models achieve promising results as general-purpose task solvers, in which ChatGPT mostly performs the best.* ChatGPT, Claude, Claude 2, Davinci003 and Davinci002 perform well in most of tasks, including complex tasks (*e.g.,* GSM8k), which have shown great potential to be general-purpose task solvers. Among them, ChatGPT exhibits a more superior model capacity on the evaluation tasks, winning the most across all tasks. In some evaluation tasks, the performance gap between ChatGPT and other closed-source models is very large, especially for complex tasks *e.g.,* 78.47 (ChatGPT) *v.s.* 49.96 (Davinci002) on GSM8k, and 79.88 (ChatGPT) *v.s.* 51.22 (Claude) on HumanEval.

• *Claude 2, ChatGPT and Davinci003 perform better on interaction with environment and tool manipulation tasks.* On the two evaluation tasks, Claude 2, ChatGPT and Davinci003, perform better than other models by a large margin, *e.g.,* 36.40 (Claude 2) *v.s.* 26.00 (Davinci002) on HotpotQA, 44.53 (ChatGPT) *v.s.* 7.74 (Claude) on Gorilla-TF, and 72.58 (Davinci003) *v.s.* 22.04 (Claude) on Gorilla-TH. A possible reason is that these three models have been specially optimized towards these advanced abilities, *e.g.,* supporting the use of external plugins.

• *All the comparison models perform not well on very difficult reasoning tasks.* On MATH and HotpotQA, all models (including ChatGPT) perform not well. The two tasks are very difficult to solve, requiring accurate understanding of complex mathematical knowledge and performing multi-hop reasoning across documents, respectively. Further, these models also have a relatively weak performance on machine translation task (WMT). A possible reason is that WMT also contains many evaluation examples in minor languages, which might not be well covered in the pre-training data of these LLMs.

Analysis of Open-Source Models. Next, we continue to show our analysis and findings about eight open-source

models (*i.e.,* LLaMA 2-Chat, Vicuna, Alpaca, ChatGLM, LLaMA 2, LLaMA, Pythia and Falcon) as follows:

• *Instruction-tuned models mostly perform better than the base models.* Among all the compared open-source methods, the instruction-tuned models (*i.e.,* LLaMA 2-Chat, Vicuna, Alpaca and ChatGLM) mostly perform better than non-instruction-tuned models (*i.e.,* LLaMA 2, LLaMA, Pythia and Falcon). It indicates that instruction tuning is generally capable of improving the few-shot or zero-shot ability of LLMs in solving various tasks. However, after instruction tuning, Vicuna (7B) and Alpaca (7B) suffer from performance degradations on LAMBADA, a language modeling task. The reason may be that the instruction data mainly focuses on enabling LLMs to follow human instructions, which is not always useful for the general language generation task.

• *These small-sized open-source models perform not well on mathematical reasoning, interaction with environment, and tool manipulation tasks.* On the tasks of mathematical reasoning, interaction with environment and tool manipulation, all these evaluated open-source models perform not well, including instruction-tuned ones. A possible reason is that the instruction data for fine-tuning these models is not specifically designed for these tasks. In addition, these closed-source models may have limited model capacities due to small model sizes.

• *The top-performing model varies on different human alignment tasks.* For different human alignment tasks, we can see that these models achieve inconsistent performance rankings. For example, LLaMA 2-Chat (7B) performs the best among the compared open-source models on TruthfulQA, while Vicuna (13B) performs the best on CrowS-Pairs. A possible reason is that these tasks are designed with specific purposes for evaluating different aspects of human alignment, and these models exhibit varied performance on different tasks, even for the variants of the same model (*e.g.,* Pythia (7B) and Pythia (12B)). More experiments and analysis on human alignment evaluation are needed to reveal more detailed findings.

• *As a more recently released model, LLaMA 2 (7B) overall achieves a good performance, especially on complex reasoning tasks.* For complex reasoning tasks, LLaMA 2 (7B) mostly performs better than other base models, *e.g.,* 43.95 (LLaMA 2 (7B)) *v.s.* 29.80 (Falcon (7B)) in C-Objects. For other tasks (*e.g.,* language generation and knowledge utilization), LLaMA 2 (7B) can also achieve comparable performance as the best-performing base models. It has used more data for pre-training (*i.e.,* about 2 trillion tokens), which mainly contributes to the excellent performance. Furthermore, it also conducts a more robust data cleaning process.

• *Scaling the open-source modes can improve the performance consistently.* By comparing the performance of Vicuna (7B) and Vicuna (13B), Pythia (7B) and Pythia (13B), we can see that the models with larger scales mostly perform better than smaller ones on these evaluation tasks, indicating the effectiveness of scaling up the model size. Across different tasks, scaling model is more beneficial for more complex tasks (*e.g.,* symbolic and mathematical reasoning), where the larger models mostly outperform smaller ones in a large margin.

The readers should be note that these findings about open-source language models are limited to the model sizes. We will continually update this part by including the results of larger versions of these models, and also call for the support of computational resources for more experiments.

8 A Practical Guidebook of Prompt Design

As discussed in Section 6, prompting is the major approach to utilizing LLMs for solving various tasks. Since the quality of prompts will largely influence the performance of LLMs in specific tasks, we set up a special section to discuss the prompt design in practice. In this section, we will first introduce the key components of prompts and discuss several principles for prompt design. Then, we evaluate ChatGPT with different prompts to show the results on several representative tasks. We are aware that there have been several existing papers [627, 628] and websites [629–631] that present the suggestions and guidelines to design good prompts. As a comparison, we mainly aim to discuss the key factors (ingredients and principles) that are useful for prompt creation, and provide experimental results and analysis on popular tasks as the reference to the beginners.

8.1 Prompt Creation

The process of creating a suitable prompt is also called *prompt engineering* [628, 632]. A well-designed prompt is very helpful to elicit the abilities of LLMs for accomplishing specific tasks. In this part, we briefly summarize the key ingredients of prompts and discuss several basic principles of prompt design.

Key Ingredients. Typically, there are four key ingredients that depict the functionality of a prompt for eliciting the abilities of LLMs to complete the tasks, including task description, input data, contextual information, and prompt style. To have an intuitive understanding of our discussion, we also present three prompt examples for question answering, meta-review generation, and text-to-SQL in Table 15.

• *Task description.* A task description is typically a specific instruction that LLMs are expected to follow. In general, one should clearly describe the task goal in natural language. For the tasks with special input or output format, detailed clarifications are often needed, and one can further utilize keywords to highlight the special settings for better guiding LLMs in task completion.

• *Input data.* In common cases, it is straightforward to describe input data (*e.g.,* an instance to be responded by LLMs) in natural language. For special input data, such as knowledge graph and table, it is necessary to apply an appropriate and convenient way to make them readable for LLMs. For structured data, linearization is commonly used to transform the original records (*e.g.,* knowledge triples) into sequences [512] due to the simplicity. Further, the programming language (*e.g.,* executable code) has also been utilized to formulate the structured data, which can also support using external tools (*e.g.,* program executor) to produce the precise results [633, 634].

• *Contextual information.* In addition to the task description and input data, contextual or background information is also essential for specific tasks. For example, retrieved

documents are highly useful for open-domain question answering as supporting evidence. Thus, it needs to include such information in a proper prompt pattern or expression format. Furthermore, in-context task exemplars are also helpful for eliciting LLMs to accomplish a complex task, which can better depict the task goal, the special output formats, and the mapping relation between input and output.

• *Prompt style.* For different LLMs, it is important to design a suitable prompt style for eliciting their abilities to solve specific tasks. Overall, one should express the prompt as a clear question or detailed instruction that can be well understood and answered. In some cases, it is also useful to add the prefix or suffix to better guide LLMs. For example, using the prefix "*Let us think step by step*" can help elicit LLMs perform step-by-step reasoning, and using the prefix "*You are an expert on this task (or in this domain)*" can boost the performance of LLMs in some specific tasks. Further, for chat-based LLMs (*e.g.,* ChatGPT), instead of directly feeding a long or complex task prompt, it is suggested to decompose it into multiple prompts for the sub-tasks and then feed them into LLMs via a multi-turn conversation [366].

Design Principles. Based on the key ingredients of prompts, we summarize several critical design principles that can help create more effective prompts for solving various tasks.

• *Expressing the task goal clearly.* Task descriptions should not be ambiguous or unclear, which likely lead to inaccurate or inappropriate responses. This highlights the need for clear and unambiguous directives when utilizing these models [61]. A clear and detailed description should contain various elements to explain a task, including task objective, input/output data (*e.g.,* "*Given a long document, I want you to generate a concise summary.*"), and the response constraints (*e.g.,* "*the length of the summary cannot exceed 50.*"). By providing a well-clarified task description, LLMs can more effectively understand the target task and generate the desired output.

• *Decomposing into easy, detailed sub-tasks.* To solve complex tasks, it is important to decompose the difficult task into several more easier, detailed sub-tasks for helping LLMs accomplish the goal step by step, which is closely related to the planning technique in Section 6.3. For example, following the suggestion [627], we can explicitly list the sub-tasks in the form of multiple numbered items (*e.g.,* "*Braid a coherent narrative by performing the following tasks: 1. ...; 2. ...; 3. ...*"). By decomposing a target task into sub-tasks, LLMs can focus on solving easier sub-tasks and finally achieve more accurate results for complex tasks.

• *Providing few-shot demonstrations.* As discussed in Section 6.1, LLMs can benefit from in-context learning for solving complex tasks, where the prompts contain a small number of task examples of the desired input-output pairs, *i.e.,* few-shot demonstrations. Few-shot demonstrations can help LLMs learn the semantic mapping between input and output without parameter tuning. In practice, it is suggested that one should generate a few high-quality demonstrations for the target task, which would highly benefit the final task performance.

• *Utilizing model-friendly format.* Since LLMs are pre-trained on specially constructed datasets, there are some prompt formats that can make LLMs better understand the instruction. For example, as the OpenAI documentation suggests, we can use `###` or `"""` as a stop symbol to separate the instruction and context, which can be better understood by LLMs. As a general guideline, most existing LLMs perform a task better in English, thus it is useful to employ English instructions to solve difficult tasks based on machine translation.

Useful Tips. In addition to the design principles, we also present a collection of useful prompt tips based on existing work or our empirical experiences in Table 14. Note that these tips are suggested in a general manner, it does not indicate that they are the best prompts for the corresponding tasks. This part will be continuously updated with more guidelines or tips. We welcome readers to contribute to this collection of prompt tips. We present the detailed procedure to contribute to the prompt tips, at the link: https://github.com/RUCAIBox/LLMSurvey/tree/main/Prompts.

8.2 Results and Analysis

In the above subsection, we have discussed the general principles to design the prompts. This part presents concrete examples of prompts to solve a number of common tasks. Specially, these task prompts are mostly from existing papers, and the experiments are conducted by using the prompts based on ChatGPT for the corresponding tasks.

Experimental Setup. To conduct the experiments, we select a variety of tasks that span language generation, knowledge utilization, complex reasoning, structure data generation, and information retrieval. For each task, we manually write a prompt that follows general guidelines introduced in Section 8.1. Note that the tested prompts may not be the optimal for these tasks, since they mainly aim to help readers understand how to write an effective prompt for solving different tasks. Also, we add a simplified prompt as the comparison for most tasks. Following the experimental settings in Section 7.4, we examine the 3-shot performance of ChatGPT on complex reasoning tasks (Colored Objects and GSM8k), and zero-shot performance on other tasks.

Results Analysis. We report the experimental results in Table 13, where we also include the supervised performance in existing papers as reference.

• *Carefully designed prompts can boost the zero-shot or few-shot performance of ChatGPT.* By comparing the results of using different prompts on the same task, we can see that using the carefully designed prompts can achieve better performance than the simpler ones. In the carefully designed prompts, we provide a more clearly expressed task description (*e.g.,* WMT and WikiFact), or use a model-friendly format (*e.g.,* GSM8k and OBQA). For example, for WikiFact task, the prompt with a more detailed task description leads to a performance increase from 29.25 to 31.21.

• *More complex tasks can benefit more from careful prompt engineering on ChatGPT.* In the WikiFact and Colored Objects tasks, the designed prompts have greatly improved the performance of ChatGPT, *i.e.,* from 23.61 to 28.47 on WikiFact and from 53.20 to 66.75 on Colored Objects. It indicates the necessity of prompt engineering for LLMs to perform well on complex tasks, since these tasks typically have specific output formats or require background knowledge.

TABLE 14: A collection of useful tips for designing prompts that are collected from online notes [627–630] and experiences from our authors, where we also show the related ingredients and principles (introduced in Section 8.1). We abbreviate principles as Prin. and list the IDs of the related principles for each prompt. ①: expressing the task goal clearly; ②: decomposing into easy, detailed sub-tasks; ③: providing few-shot demonstrations; ④: utilizing model-friendly format.

Ingredient	Collected Prompts	Prin.
Task Description	T1. Make your prompt **as detailed as possible**, *e.g.,* "*Summarize the article into a short paragraph within 50 words. The major storyline and conclusion should be included, and the unimportant details can be omitted.*"	①
	T2. It is helpful to let the LLM know that it is **an expert with a prefixed prompt**, *e.g.,* "*You are a sophisticated expert in the domain of compute science.*"	①
	T3. Tell the model **more what it should do**, but not what it should not do.	①
	T4. To avoid the LLM to generate too long output, you can just use the prompt: "*Question: Short Answer: *". Besides, you can also use the following suffixes, "*in a or a few words*", "*in one of two sentences*".	①
Input Data	I1. For the question required factual knowledge, it is useful to first **retrieve relevant documents** via the search engine, and then **concatenate them into the prompt** as reference.	④
	I2. To highlight some important parts in your prompt, please **use special marks**, *e.g., quotation* ("") and *line break* (\n). You can also use both of them for emphasizing.	④
Contextual Information	C1. For complex tasks, you can **clearly describe the required intermediate steps** to accomplish it, *e.g.,* "*Please answer the question step by step as: Step 1 - Decompose the question into several sub-questions, · · ·* "	②
	C2. If you want LLMs to provide the score for a text, it is necessary to provide a **detailed description about the scoring standard** with examples as reference.	①
	C3. When LLMs generate text according to some context (*e.g.,* making recommendations according to purchase history), instructing them with **the explanation about the generated result** conditioned on context is helpful to improve the quality of the generated text.	②
	C4. An approach similar to **tree-of-thoughts** but can be **done in one prompt**: *e.g., Imagine three different experts are answering this question. All experts will write down one step of their thinking, then share it with the group of experts. Then all experts will go on to the next step, etc. If any expert realizes they're wrong at any point then they leave. The question is*	②
Demonstration	D1. **Well-formatted in-context exemplars** are very useful, especially for producing the outputs with complex formats.	③
	D2. For few-shot chain-of-thought prompting, you can also use the prompt "*Let's think step-by-step*", and the few-shot examples should be **separated by "\n"** instead of full stop.	①③
	D3. You can also **retrieve similar examples** in context to supply the useful task-specific knowledge for LLMs. To retrieve more relevant examples, it is useful to **first obtain the answer** of the question, and then concatenate it with the question for retrieval.	③④
	D4. The **diversity of the in-context exemplars** within the prompt is also useful. If it is not easy to obtain diverse questions, you can also seek to keep the **diversity of the solutions** for the questions.	③
	D5. When using chat-based LLMs, you can **decompose in-context exemplars into multi-turn messages**, to better match the human-chatbot conversation format. Similarly, you can also decompose the reasoning process of an exemplars into multi-turn conversation.	③
	D6. **Complex and informative** in-context exemplars can help LLMs answer complex questions.	③
	D7. As a symbol sequence can typically be divided into multiple segments (*e.g.,* $i_1, i_2, i_3 \longrightarrow i_1, i_2$ and i_2, i_3), the preceding ones can be used **as in-context exemplars** to guide LLMs to predict the subsequent ones, meanwhile providing historical information.	②③
	D8. **Order matters** for in-context exemplars and prompts components. For very long input data, the position of the question (first or last) may also affect the performance.	③
	D9. If you can not obtain the in-context exemplars from existing datasets, an alternative way is to use the **zero-shot generated ones** from the LLM itself.	③
Other Designs	O1. Let the **LLM check its outputs** before draw the conclusion, *e.g.,* "*Check whether the above solution is correct or not.*"	②
	O2. If the LLM can not well solve the task, you can **seek help from external tools** by prompting the LLM to manipulate them. In this way, the tools should be encapsulated into callable APIs with detailed description about their functions, to better guide the LLM to utilize the tools.	④
	O3. The prompt should be **self-contained**, and better not include pronouns (*e.g.,* it and they) in the context.	①
	O4. When using LLMs for **comparing** two or more examples, the order affects the performance a lot.	①
	O5. Before the prompt, **assigning a role for the LLM** is useful to help it better fulfill the following task instruction, *e.g., "I want you to act as a lawyer"*.	①
	O6. OpenAI models can perform a task better in English than other languages. Thus, it is useful to first **translate the input into English** and then feed it to LLMs.	④
	O7. For multi-choice questions, it is useful to **constrain the output space** of the LLM. You can use a more detailed explanation or just imposing constraints on the logits.	①
	O8. For sorting based tasks (*e.g.,* recommendation), instead of directly outputting the complete text of each item after sorting, one can **assign indicators** (*e.g., ABCD*) to the unsorted items and instruct the LLMs to directly output the sorted indicators.	①

Our example prompts provide more detailed task description (*e.g.,* output format and task goal), which can help ChatGPT better understand the complex task requirement for fulfilling it.

• *For mathematical reasoning tasks, it is more effective to design specific prompts based on the format of programming language.* For GSM8k, the designed prompt employs code-formatted few-shot demonstrations to convert this mathematical reasoning task into code generation task, which can leverage the strong code synthesis ability of ChatGPT for solving mathematical problems. Further, with the help of an external program executor, we are able to obtain more precise results instead of using LLMs for arithmetic operation. As we can see, the performance is boosted from 78.47 to 79.30 on GSM8k, indicating the usefulness of programming language in mathematical reasoning tasks.

• *In knowledge utilization and complex reasoning tasks, ChatGPT with proper prompts achieves comparable performance or even outperforms the supervised baselines methods.* In knowledge utilization and complex reasoning tasks, ChatGPT with proper zero-shot or few-shot prompts can achieve comparable performance or even outperform the super-

vised methods, *e.g.,* 31.21 (ChatGPT) *v.s.* 34.20 (supervised baseline) on WikiFact. Despite that, ChatGPT still performs worse than supervised baseline models on some specific tasks (*e.g.,* ARC and WikiFact), since these supervised models have been specially optimized with task-specific data.

• *Through suitable prompt engineering, LLMs can handle some non-traditional NLP tasks.* With the help of specific prompts, ChatGPT can also accomplish non-traditional NLP tasks, *i.e.,* the general recommendation and conversational recommendation. A key point is that these tasks can be well expressed or described in natural language. However, the performance of ChatGPT is still far from the referenced performance in these tasks, as LLMs cannot directly fit these tasks, which require specific domain knowledge and task adaptation [283, 635].

9 Applications

As LLMs are pre-trained on a mixture of source corpora, they can capture rich knowledge from large-scale pre-training data, thus having the potential to serve as domain experts or specialists for specific areas. In this section, we briefly review the recent progress on the applications of LLMs on several representative domains, including healthcare, education, law, finance, and scientific research.

Healthcare is a vital application field closely related to human life. Ever since the advent of ChatGPT, a number of studies have applied ChatGPT or other LLMs to the medical domain. It has been shown that LLMs are capable of handling a variety of healthcare tasks, *e.g.,* biology information extraction [637], medical advice consultation [638], mental health analysis [639], and report simplification [640]. As the major technical approach, researchers typically design specific prompts or instructions to guide LLMs to perform a wide range of medical tasks. To further harness the power of LLMs in the healthcare domain, researchers propose to develop healthcare-related LLMs. Specifically, the Med-PaLM models [282, 641] achieves expert-level performance on the United States Medical Licensing Examination (USMLE), and earns greater approval from physicians in answering consumer's medical questions. However, LLMs may fabricate medical misinformation [640, 642], *e.g.,* misinterpreting medical terms and suggesting advice inconsistent with medical guidelines. In addition, it would also raise privacy concerns to upload the health information of patients [637] into a commercial server that support the LLM.

Education is also an important application domain where LLMs potentially exert significant influence. Existing work has found that LLMs can achieve student-level performance on standardized tests [46] in a variety of subjects of mathematics (*e.g.,* physics, computer science) on both multiple-choice and free-response problems. In addition, empirical studies have shown that LLMs can serve as writing or reading assistant for education [643, 644]. A recent study [644] reveals that ChatGPT is capable of generating logically consistent answers across disciplines, balancing both depth and breadth. Another quantitative analysis [643] shows that students utilizing ChatGPT (either keeping or refining the results from LLMs as their own answers) perform better than average students in some courses from the computer security field. Recently, several perspective papers [645, 646] also explore various application scenarios of LLMs in classroom teaching, such as teacher-student collaboration, personalized learning, and assessment automation. However, the application of LLMs in education may lead to a series of practical issues, *e.g.,* plagiarism, potential bias in AI-generated content, overreliance on LLMs, and inequitable access for non-English speaking individuals [647].

Law is a specialized domain that is built on professional domain knowledge. Recently, a number of studies have applied LLMs to solve various legal tasks, *e.g.,* legal document analysis [648], legal judgment prediction [649], and legal document writing [650]. A recent study [651] has found that LLMs exhibit powerful abilities of legal interpretation and reasoning. Moreover, the latest GPT-4 model achieves a top 10% score in a simulated bar exam compared with human test-takers [46]. To further improve the performance of LLMs in the law domain, specially designed legal prompt engineering are employed to yield advanced performance in long legal document comprehension and complex legal reasoning [652, 653]. To summarize the progress, LLMs can act as helpful assistants to legal profession. Despite the progress, the use of LLMs in law raises concerns about legal challenges, including copyright issues [654], personal information leakage [655], or bias and discrimination [656].

Finance is an important field where LLMs have promising application prospects. LLMs have been employed on various finance related tasks, such as numerical claim detection [657], financial sentiment analysis [658], financial named entity recognition [659], and financial reasoning [660]. Despite the competitive zero-shot performance exhibited by general-purpose LLMs in the finance tasks, they still underperform domain-specific PLMs containing million-scale parameters [657]. To leverage the scaling effect of LLMs, researchers collect large-scale finance corpora for continually pre-training LLMs (*e.g.,* BloombergGPT [286], XuanYuan 2.0 [661], and FinGPT [662]). BloombergGPT has demonstrated remarkable performance across a diverse range of financial tasks while maintaining competitive performance in general-purpose tasks [286]. Nevertheless, it is imperative to consider the potential risks in the application of LLMs in finance, as the generation of inaccurate or harmful content by LLMs could have significant adverse implications for financial markets [286]. Therefore, it needs more strict reviewing and monitoring on the use of LLMs in the financial field.

Scientific research is another promising field that LLMs can empower the development progress. Prior research demonstrates the effectiveness of LLMs in handling knowledge-intensive scientific tasks (*e.g.,* PubMedQA [663], BioASQ [664]), especially for LLMs that are pre-trained on scientific-related corpora (*e.g.,* Galactica [35], Minerva [163]). Given the excellent general abilities and broad scientific knowledge, LLMs hold significant potential as helpful assistants across various stages of the scientific research pipeline [665]. First, during the literature survey stage, LLMs can help conduct a comprehensive overview of the progress in a specific research field [666, 667]. Second, during the research idea generation stage, LLMs demonstrate

TABLE 15: Example instructions collected from [627, 636]. The blue text denotes the task description, the red text denotes the contextual information, the green text denotes the demonstrations, and the gold text denotes the prompt style.

Use the provided articles delimited by triple quotes to answer questions. If the answer cannot be found in the articles, write "I could not find an answer."

Articles: """Joao Moutinho is a Portuguese footballer who last played as a central midfielder for Premier League club Wolverhampton Wanderers and the Portugal national team."""

Question: Is the following sentence plausible? 'Joao Moutinho was out at third.'

Answer: Let's think step by step. Joao Moutinho is a soccer player. Being out at third is part of baseball, not soccer. So the answer is No.

...
<Demonstrations>

Articles: <insert articles, each delimited by triple quotes>
Question: <insert question>
Answer:

Prepare a meta-review by answering the following questions from the reviewer comments (provided after the questions).

1. Based on the reviewer's comments, what are the core contributions made by this manuscript?
2. What are the common strengths of this work, as mentioned by multiple reviewers?
3. What are the common weaknesses of this work, as highlighted by multiple reviewers?
4. What suggestions would you provide for improving this paper?
5. What are the missing references mentioned by the individual reviews?

The review texts are below: <insert three comments R_1, R_2, R_3 from the reviewers>
Meta-review: <insert meta-review>
...
<Demonstrations>

Provide justification for your response in detail by explaining why you made the choices you actually made. A good output should be coherent, highlight major strengths/issues mentioned by multiple reviewers, be less than 400 words in length, and finally, the response should be in English only.

The review texts are below: <insert three comments R_1, R_2, R_3 from the reviewers>
Meta-review:

```
CREATE TABLE Highschooler (
ID int primary key,
name text,
grade int
);
/*
3 example rows:
SELECT * FROM Highschooler LIMIT 3;
ID    name    grade
1234    Janie    8
5678    Mary    8
9012    Mike    9
*/
```

Using valid SQLite, answer the following questions for the tables provided above.

Question: What is Kyle's id?
SQL: SELECT ID FROM Highschooler WHERE name="Kyle";
...
<Demonstrations>

Question: <insert question>
SQL:

---

the ability to generate intriguing scientific hypotheses [668]. Third, during the data analysis stage, LLMs can be employed to conduct automatic approaches to analyzing the data characteristics, including data exploration, visualization, and deriving analytical conclusions [669, 670]. Fourth, during the paper writing stage, researchers can also benefit from the assistance of LLMs in scientific writing [671, 672], in which LLMs can offer valuable support for scientific writing through diverse means, such as summarizing the existing content and polishing the writing [673]. In addition, LLMs can aid in the automated paper review process, encompassing tasks such as error detection, checklist verification, and candidate ranking [674]. Despite these advances, there is much room for improving the capacities of LLMs to serve as helpful, trustworthy scientific assistants, to both increase the quality of the generated scientific content and reduce the harmful hallucinations.

Summary. In addition to the aforementioned work, the applications of LLMs have been also discussed in several other domains. For instance, in the psychologic domain, some recent work has studied the human-like characteristics of LLMs, such as self-awareness, theory of mind (ToM), and affective computing [675, 676]. In particular, an empirical evaluation of ToM conducted on two classic false-belief tasks speculates that LLMs may have ToM-like abilities since the model in the GPT-3.5 series achieves comparable

performance with nine-year-old children in ToM task [675]. In addition, another line of work has investigated applying LLMs into the software development domain, *e.g.,* code suggestion [677], code summarization [678], and automated program repair [679]. To summarize, to assist humans by LLMs in real-world tasks has become a significant area of research. However, it also presents challenges. Ensuring the accuracy of LLM-generated content, addressing biases, and maintaining user privacy and data security are crucial considerations when applying LLMs to real-world scenarios.

10 Conclusion and Future Directions

In this survey, we have reviewed the recent progress of large language models (LLMs), and introduced the key concepts, findings, and techniques for understanding and utilizing LLMs. We focus on the large-sized models (*i.e.,* having a size larger than 10B) while excluding the contents of early pre-trained language models (*e.g.,* BERT and GPT-2) that have been well covered in the existing literature. In particular, our survey has discussed four important aspects of LLMs, *i.e.,* pre-training, adaptation tuning, utilization, and evaluation. For each aspect, we highlight the techniques or findings that are key to the success of LLMs. Furthermore, we also summarize the available resources for developing LLMs and discuss important implementation guidelines for reproducing LLMs. This survey tries to cover the most recent literature about LLMs and provides a good reference resource on this topic for both researchers and engineers.

Next, we summarize the discussions of this survey, and introduce the challenges and future directions for LLMs, in the following aspects.

Theory and Principle. To understand the underlying working mechanism of LLMs, one of the greatest mysteries is how information is distributed, organized, and utilized through the very large, deep neural network. It is important to reveal the basic principles or elements that establish the foundation of the abilities of LLMs. In particular, *scaling* seems to play an important role in increasing the capacity of LLMs [31, 55, 59]. It has been shown that some emergent abilities would occur in an unexpected way (a sudden performance leap) when the parameter scale of language models increases to a critical size (*e.g.,* 10B) [31, 33], typically including in-context learning, instruction following, and step-by-step reasoning. These emergent abilities are fascinating yet perplexing: *when* and *how* they are obtained by LLMs are not yet clear. Recent studies either conduct extensive experiments for investigating the effect of emergent abilities and the contributing factors to such abilities [345, 680, 681], or explain some specific abilities with existing theoretical frameworks [60, 380]. An insightful technical post also specially discusses this topic [47], taking the GPT-series models as the target. However, more formal theories and principles to understand, characterize, and explain the abilities or behaviors of LLMs are still missing. Since emergent abilities bear a close analogy to phase transitions in nature [31, 58], cross-discipline theories or principles (*e.g.,* whether LLMs can be considered as some kind of complex systems) might be useful to explain and understand the behaviors of LLMs. These fundamental questions are worth exploring for the research community, which are important for developing the next-generation LLMs.

Model Architecture. Due to the scalability and effectiveness, Transformer, consisting of stacked multi-head self-attention layers, has become the de facto architecture for building LLMs. Various strategies have been proposed to improve the performance of this architecture, such as neural network configuration and scalable parallel training (see discussions in Section 4.2.2). To enhance the model capacity (*e.g.,* the multi-turn conversation ability), existing LLMs typically maintain a long context window, *e.g.,* GPT-4-32k has an extremely large context length of 32,768 tokens. Thus, a practical consideration is to reduce the time complexity (originally to be quadratic costs) incurred by the standard self-attention mechanism. It is important to investigate the effect of more efficient Transformer variants in building LLMs [682], *e.g.,* sparse attention has been used in GPT-3 [55]. Besides, catastrophic forgetting has been a long-standing challenge for neural networks, which also has a negative impact on LLMs. When tuning LLMs with new data, the originally learned knowledge is likely to be damaged, *e.g.,* fine-tuning a LLM according to some specific tasks will affect the general ability of LLMs. A similar case occurs when LLMs are aligned with human values (called *alignment tax* [61, 295]). Thus, it is necessary to consider extending existing architectures with more flexible mechanisms or modules that can effectively support data update and task specialization.

Model Training. In practice, it is very difficult to pre-train capable LLMs, due to the huge computation consumption and the sensitivity to data quality and training tricks [69, 84]. Thus, it becomes particularly important to develop more systemic, economical pre-training approaches for optimizing LLMs, considering the factors of model effectiveness, efficiency optimization, and training stability. More model checking or performance diagnosis methods (*e.g.,* predictable scaling in GPT-4 [46]) should be developed in order to detect early abnormal issues during training. Furthermore, it also calls for more flexible mechanisms of hardware support or resource schedule, so as to better organize and utilize the resources in a computing cluster. Since it is very costly to pre-train a LLM from scratch, it is important to design suitable mechanisms for continually pre-training or fine-tuning the LLM based on publicly available model checkpoints (*e.g.,* LLaMA [57] and Flan-T5 [64]). For this purpose, a number of technical issues have to be resolved, *e.g.,* catastrophic forgetting and task specialization. However, to date, there still lack open-source model checkpoints for LLMs with complete pre-processing and training logs (*e.g.,* the scripts to prepare the pre-training data) for reproduction. We believe that it will be of great value to report more technical details in open-source models for the research of LLMs. Furthermore, it is also important to develop more improvement tuning strategies that effectively elicits the model abilities.

Model Utilization. Since fine-tuning is very costly in real applications, *prompting* has become the prominent approach to using LLMs. By combining task descriptions and demonstration examples into prompts, in-context learning (a spe-

cial form of prompting) endows LLMs with the ability to perform well on new tasks, even outperforming full-data fine-tuned models in some cases. Furthermore, to enhance the ability of complex reasoning, advanced prompting techniques have been proposed, exemplified by the chain-of-thought (CoT) strategy, which includes the intermediate reasoning steps into prompts. However, existing prompting approaches still have several deficiencies described as follows. Firstly, it involves considerable human efforts in the design of prompts. It would be quite useful to automatically generate effective prompts for solving various tasks. Secondly, some complex tasks (*e.g.,* formal proof and numerical computation) require specific knowledge or logic rules, which may not be well expressed in natural language or demonstrated by examples. Thus, it is important to develop more informative, flexible task formatting methods for prompts[54]. Thirdly, existing prompting strategies mainly focus on single-turn performance. It is useful to develop interactive prompting mechanisms (*e.g.,* through natural language conversations) for solving complex tasks, which have been demonstrated to be very useful by ChatGPT.

Safety and Alignment. Despite their capacities, LLMs pose similar safety challenges as small language models. For example, LLMs exhibit a tendency to generate hallucinations [518], which are texts that seem plausible but may be factually incorrect. What is worse, LLMs might be elicited by intentional instructions to produce harmful, biased, or toxic texts for malicious systems, leading to the potential risks of misuse [55, 61]. To have a detailed discussion of the safety issues of LLMs (*e.g.,* privacy, overreliance, disinformation, and influence operations), the readers can refer to the GPT-3/4 technical reports [46, 55]. As the major approach to averting these issues, reinforcement learning from human feedback (RLHF) [61, 103] has been widely used by incorporating humans in the training loop for developing well-aligned LLMs. To improve the model safety, it is also important to include safety-relevant prompts during RLHF, as shown by GPT-4 [46]. However, RLHF heavily relies on high-quality human feedback data from professional labelers, making it difficult to be properly implemented in practice. Therefore, it is necessary to improve the RLHF framework for reducing the efforts of human labelers and seek a more efficient annotation approach with guaranteed data quality, *e.g.,* LLMs can be employed to assist the labeling work. It is also meaningful to establish the proper learning mechanism for LLMs to obtain human feedback via chatting and directly utilize it for self-improvement. In addition, privacy concerns are also important to consider when fine-tuning LLMs with domain-specific data, and federated learning libraries [683] can be useful in privacy-restricted scenarios.

Application and Ecosystem. As LLMs have shown a strong capacity in solving various tasks, they can be applied in a broad range of real-world applications (*i.e.,* following task-specific natural language instructions). As a remarkable progress, ChatGPT has potentially changed the way how humans access information, which has been implemented in the release of *New Bing*. In the near future, it can be foreseen that LLMs would have a significant impact on information-seeking techniques, including both search engines and recommender systems. Furthermore, the development and use of intelligent information assistants would be highly promoted with the technology upgrade from LLMs. In a broader scope, this wave of technical innovation would lead to an ecosystem of LLM-empowered applications (*e.g.,* the support of plugins by ChatGPT), which has a close connection with human life. Lastly, the rise of LLMs sheds light on the exploration of artificial general intelligence (AGI). It is promising to develop more smart intelligent systems (possibly with multi-modality signals) than ever. However, in this development process, AI safety should be one of the primary concerns, *i.e.,* making AI lead to good for humanity but not bad [40].

54. It seems that an alternative approach to this issue is to invoke external tools, *e.g.,* the plugins for ChatGPT, when the task is difficult to solve via text generation.

CODA

It is not an easy job to write this long survey and update its content with timely work. First of all, we would like to sincerely thank the support from the readers and our team members. We work very hard on this survey, and hope that it can present a comprehensive, timely reference for LLMs.

Survey Writing. This survey was planned during a discussion meeting held by our research team, and we aimed to summarize the recent advances of large language models as a highly readable report for our team members. The first draft was finished on March 13, 2023, in which our team members tried their best to include the related studies about LLMs in a relatively objective, comprehensive way. Then, we have extensively revised the writing and contents in several passes. Due to the space limit, we can only include a fraction of existing LLMs in Figure 2 and Table 1, by setting the selection criterion. However, we set a more relaxed criterion for model selection on our GitHub page (https://github.com/RUCAIBox/LLMSurvey), which will be regularly maintained. We release the initial version on March 31, 2023, the major revision on June 29, 2023, and the latest version (v12) on September 10, 2023.

Seeking for Advice. Despite all our efforts, this survey is still far from perfect: we are likely to miss important references or topics, and might also have non-rigorous expressions or discussions. We will continuously update this survey, and improve the quality as much as we can. For us, survey writing is also a learning process for LLMs by ourselves. For readers with constructive suggestions to improve this survey, you are welcome to leave comments on the GitHub page of our survey or directly email our authors. We will make revisions following the received comments or suggestions in a future version, and acknowledge the readers who have contributed constructive suggestions in our survey.

Update log. In this part, we regularly maintain a update log for the submissions of this survey to arXiv:

- First release on March 31, 2023: the initial version.
- Update on April 9, 2023: add the affiliation information, revise Figure 2 and Table 1 and clarify the correspond-

ing selection criterion for LLMs, improve the writing, and correct some minor errors.
- Update on April 11, 2023: correct the errors for library resources.
- Update on April 12, 2023: revise Figure 2 and Table 1, and clarify the release date of LLMs.
- Update on April 16, 2023: add a new Section 2.2 about the technical evolution of GPT-series models.
- Update on April 24, 2023: add the discussion about scaling laws and add some explanations about the model sizes for emergent abilities (Section 2.1); add an illustrative figure for the attention patterns for different architectures in Figure 7, and add the detailed formulas in Table 4.
- Update on April 25, 2023: revise some copy errors in figures and tables.
- Update on April 27, 2023: add efficient tuning in Section 5.3.
- Update on April 28, 2023: revise Section 5.3.
- Update on May 7, 2023: revise Table 1, Table 2, and some minor points.
- Update on June 29, 2023 (major revision):
  - Section 1: add Figure 1 for the trends of published LLM papers in arXiv;
  - Section 2: add Figure 3 for GPT's evolution and the corresponding discussion;
  - Section 3: add Figure 4 for LLaMA family and the corresponding discussion;
  - Section 5: add latest discussion about the synthetic data formatting of instruction tuning in Section 5.1.1, the empirical analysis for instruction tuning in Section 5.1.4, parameter-efficient model adaptation in Section 5.3 and memory-efficient adaptation in Section 5.4;
  - Section 6: add latest discussion about the underlying mechanism of ICL 6.1.3, planning for complex task solving in Section 6.3;
  - Section 7: update Table 10 for representative datasets for evaluating advanced abilities of LLMs, and empirical ability evaluation in Section 7.4;
  - Section 8: add prompt design;
  - Section 9: add the discussions on applications of LLMs in finance and scientific research domains;
- Update on September 10, 2023 (this version):
  - Claim the copyrights of the figures and tables in this paper.
  - Add latest LLMs, techniques and their descriptions in Section 3, Section 4, Section 5, Section 6 and Section 7;
  - Section 4: add latest discussion about the decoding strategy in Section 4.2.4;
  - Section 5: add latest discussion about the practical tricks for instruction tuning in Section 5.1.2, the empirical analysis on LLaMA (13B) for instruction tuning in Section 5.1.4, practical strategies for RLHF in Section 5.2.3, alignment without RLHF in Section 5.2.4 and remarks on SFT and RLHF in Section 5.2.5;
  - Section 6: update the content about the planning for complex task solving in Section 6.3;
  - Section 7: add discussions about evaluation approaches in Section 7.3.2, Table 11 for the category of existing evaluation work, and update empirical ability evaluation in Section 7.4 and the results on Table 12;
  - Section 8: add new prompt examples in Table 14;

Planning Content. We will regularly include new content into this survey, to make it more self-contained and up-to-date. Here, we list several potential topics that might appear in the next major version(s): (1) more experiments with larger language models for both instruction tuning and ability evaluation; (2) more detailed prompting practice; (3) training recipe; (4) more theoretical analysis and discussion; (5) more discussions on applications.

Clarifications on Experiments. In this version, we have included a number experiments on instruction-tuning (Table 8), overall ability evaluation (Table 12), and prompt engineering (Table 13). Due to the limit of computational resources, our experiments are not complete, limited to small-sized models or a few comparisons. Despite that, we feel that it might be meaningful to share the partial results to the public. We will try to include the missing results of larger models or more comparisons in the future versions. **We also call for support of computing power for conducting more comprehensive experiments.**

Chinese Version. We also provide a translated Chinese version (corresponding to the first release) of this survey paper at the link: https://github.com/RUCAIBox/LLMSurvey/blob/main/assets/LLM_Survey_Chinese.pdf. Four volunteers contribute to check and revise the content, and they are Yiwen Hu, Xinming Hou, Yanbin Yin, and Zhanshuo Cao (in order of contribution). We will also continuously update the Chinese version, but it may not be as timely as the latest English version.

ACKNOWLEDGMENTS

The authors would like to thank Yankai Lin and Yutao Zhu for proofreading this paper. Since the first release of this paper, we have received a number of valuable comments from the readers. We sincerely thank the readers who have written to us with constructive suggestions and comments: Tyler Suard, Damai Dai, Liang Ding, Stella Biderman, Kevin Gray, Jay Alammar, Yubo Feng, Mark Holmstrom, Xingdong Liu, Il-Seok Oh, Yiting Liu, Shaojun Wang, Gaoyan Ou, Todd Morrill, Hao Liu, Zhenyu Zhang, and Xinlin Zhuang.

Since the v11 version (June 29, 2023), we have been adding a large number of experiments and prompt practices. These new contents are completed by a number of volunteers in our team. Here, we add a special part to thank all the students who have worked very hard on this part (also including the ones on our author list).

Contribution on Experiments. We would like to sincerely thank the following people for their hard work involved in experiments shown in Table 12.

• Xiaoxue Cheng: implement the experiments for evaluation on Language Generation and HaluEval tasks.

• Yuhao Wang: implement the experiments for evaluation on interaction with environment tasks.

• Bowen Zheng: implement the experiments for evaluation on tool manipulation tasks.

Contribution on Tips. We list the following guys for their contributions on the corresponding numbers of provided tips for designing prompts in Table 14.

• Xiaolei Wang: T3, O3
• Beichen Zhang: D2, D5
• Zhipeng Chen: D3, D4
• Junjie Zhang: D6
• Bowen Zheng: D7
• Zican Dong: D8
• Xinyu Tang: C2
• Yifan Du: T4
• Tianyi Tang: O6, O7, D9
• Yupeng Hou: O8, C3
• Salvatore Raieli: C4

REFERENCES

[1] S. Pinker, *The Language Instinct: How the Mind Creates Language*. Brilliance Audio; Unabridged edition, 2014.

[2] M. D. Hauser, N. Chomsky, and W. T. Fitch, "The faculty of language: what is it, who has it, and how did it evolve?" *science*, vol. 298, no. 5598, pp. 1569–1579, 2002.

[3] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. LIX, no. 236, pp. 433–460, 1950.

[4] F. Jelinek, *Statistical Methods for Speech Recognition*. MIT Press, 1998.

[5] J. Gao and C. Lin, "Introduction to the special issue on statistical language modeling," *ACM Trans. Asian Lang. Inf. Process.*, vol. 3, no. 2, pp. 87–93, 2004.

[6] R. Rosenfeld, "Two decades of statistical language modeling: Where do we go from here?" *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1270–1278, 2000.

[7] A. Stolcke, "Srilm-an extensible language modeling toolkit," in *Seventh international conference on spoken language processing*, 2002.

[8] X. Liu and W. B. Croft, "Statistical language modeling for information retrieval," *Annu. Rev. Inf. Sci. Technol.*, vol. 39, no. 1, pp. 1–31, 2005.

[9] C. Zhai, *Statistical Language Models for Information Retrieval*, ser. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2008.

[10] S. M. Thede and M. P. Harper, "A second-order hidden markov model for part-of-speech tagging," in *27th Annual Meeting of the Association for Computational Linguistics, University of Maryland, College Park, Maryland, USA, 20-26 June 1999*, R. Dale and K. W. Church, Eds. ACL, 1999, pp. 175–182.

[11] L. R. Bahl, P. F. Brown, P. V. de Souza, and R. L. Mercer, "A tree-based statistical language model for natural language speech recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 7, pp. 1001–1008, 1989.

[12] T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean, "Large language models in machine translation," in *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, J. Eisner, Ed. ACL, 2007, pp. 858–867.

[13] S. M. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 35, no. 3, pp. 400–401, 1987.

[14] W. A. Gale and G. Sampson, "Good-turing frequency estimation without tears," *J. Quant. Linguistics*, vol. 2, no. 3, pp. 217–237, 1995.

[15] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, 2003.

[16] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," in *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, T. Kobayashi, K. Hirose, and S. Nakamura, Eds. ISCA, 2010, pp. 1045–1048.

[17] S. Kombrink, T. Mikolov, M. Karafiát, and L. Burget, "Recurrent neural network based language modeling in meeting recognition," in *INTERSPEECH 2011, 12th Annual Conference of the International Speech Communication Association, Florence, Italy, August 27-31, 2011*. ISCA, 2011, pp. 2877–2880.

[18] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, 2011.

[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, Eds., 2013, pp. 3111–3119.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2013.

[21] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, M. A. Walker, H. Ji, and A. Stent, Eds. Association for Computational Linguistics, 2018, pp. 2227–2237.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-*

9, 2017, Long Beach, CA, USA, 2017, pp. 5998–6008.

[23] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 4171–4186.

[24] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, 2020, pp. 7871–7880.

[25] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *J. Mach. Learn. Res*, pp. 1–40, 2021.

[26] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, p. 9, 2019.

[27] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019.

[28] V. Sanh, A. Webson, C. Raffel, S. H. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, A. Raja, M. Dey, M. S. Bari, C. Xu, U. Thakker, S. S. Sharma, E. Szczechla, T. Kim, G. Chhablani, N. V. Nayak, D. Datta, J. Chang, M. T. Jiang, H. Wang, M. Manica, S. Shen, Z. X. Yong, H. Pandey, R. Bawden, T. Wang, T. Neeraj, J. Rozen, A. Sharma, A. Santilli, T. Févry, J. A. Fries, R. Teehan, T. L. Scao, S. Biderman, L. Gao, T. Wolf, and A. M. Rush, "Multitask prompted training enables zero-shot task generalization," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[29] T. Wang, A. Roberts, D. Hesslow, T. L. Scao, H. W. Chung, I. Beltagy, J. Launay, and C. Raffel, "What language model architecture and pretraining objective works best for zero-shot generalization?" in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, ser. Proceedings of Machine Learning Research, vol. 162, 2022, pp. 22 964–22 984.

[30] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," *CoRR*, vol. abs/2001.08361, 2020.

[31] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus, "Emergent abilities of large language models," *CoRR*, vol. abs/2206.07682, 2022.

[32] M. Shanahan, "Talking about large language models," *CoRR*, vol. abs/2212.03551, 2022.

[33] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. H. Chi, Q. Le, and D. Zhou, "Chain of thought prompting elicits reasoning in large language models," *CoRR*, vol. abs/2201.11903, 2022.

[34] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre, "Training compute-optimal large language models," vol. abs/2203.15556, 2022.

[35] R. Taylor, M. Kardas, G. Cucurull, T. Scialom, A. Hartshorn, E. Saravia, A. Poulton, V. Kerkez, and R. Stojnic, "Galactica: A large language model for science," *CoRR*, vol. abs/2211.09085, 2022.

[36] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Comput. Surv.*, pp. 195:1–195:35, 2023.

[37] C. Zhou, Q. Li, C. Li, J. Yu, Y. Liu, G. Wang, K. Zhang, C. Ji, Q. Yan, L. He, H. Peng, J. Li, J. Wu, Z. Liu, P. Xie, C. Xiong, J. Pei, P. S. Yu, and L. Sun, "A comprehensive survey on pretrained foundation models: A history from BERT to chatgpt," *CoRR*, vol. abs/2302.09419, 2023.

[38] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, Y. Yao, A. Zhang, L. Zhang, W. Han, M. Huang, Q. Jin, Y. Lan, Y. Liu, Z. Liu, Z. Lu, X. Qiu, R. Song, J. Tang, J. Wen, J. Yuan, W. X. Zhao, and J. Zhu, "Pre-trained models: Past, present and future," *AI Open*, vol. 2, pp. 225–250, 2021.

[39] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," *CoRR*, vol. abs/2003.08271, 2020.

[40] S. Altman, "Planning for agi and beyond," *OpenAI Blog*, February 2023.

[41] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, and Y. Zhang, "Sparks of artificial general intelligence: Early experiments with gpt-4," vol. abs/2303.12712, 2023.

[42] S. Huang, L. Dong, W. Wang, Y. Hao, S. Singhal, S. Ma, T. Lv, L. Cui, O. K. Mohammed, B. Patra, Q. Liu, K. Aggarwal, Z. Chi, J. Bjorck, V. Chaudhary, S. Som, X. Song, and F. Wei, "Language is not all you need: Aligning perception with language models," *CoRR*, vol. abs/2302.14045, 2023.

[43] Y. Cao, S. Li, Y. Liu, Z. Yan, Y. Dai, P. S. Yu, and L. Sun, "A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt," *arXiv preprint arXiv:2303.04226*, 2023.

[44] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu *et al.*, "Palm-e: An embodied multimodal language model," *arXiv preprint arXiv:2303.03378*, 2023.

[45] C. Wu, S. Yin, W. Qi, X. Wang, Z. Tang, and N. Duan, "Visual chatgpt: Talking, drawing and editing with visual foundation models," *arXiv preprint arXiv:2303.04671*, 2023.

[46] OpenAI, "Gpt-4 technical report," *OpenAI*, 2023.

[47] Y. Fu, H. Peng, and T. Khot, "How does gpt obtain its ability? tracing emergent abilities of language models to their sources," *Yao Fu's Notion*, Dec 2022.

[48] J. Li, T. Tang, W. X. Zhao, and J. Wen, "Pretrained language model for text generation: A survey," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, Z. Zhou, Ed. ijcai.org, 2021, pp. 4492–4499.

[49] P. Lu, L. Qiu, W. Yu, S. Welleck, and K. Chang, "A survey of deep learning for mathematical reasoning," *CoRR*, vol. abs/2212.10535, 2022.

[50] Q. Dong, L. Li, D. Dai, C. Zheng, Z. Wu, B. Chang, X. Sun, J. Xu, L. Li, and Z. Sui, "A survey for in-context learning," *CoRR*, vol. abs/2301.00234, 2023.

[51] J. Huang and K. C. Chang, "Towards reasoning in large language models: A survey," *CoRR*, vol. abs/2212.10403, 2022.

[52] S. Qiao, Y. Ou, N. Zhang, X. Chen, Y. Yao, S. Deng, C. Tan, F. Huang, and H. Chen, "Reasoning with language model prompting: A survey," *CoRR*, vol. abs/2212.09597, 2022.

[53] J. Zhou, P. Ke, X. Qiu, M. Huang, and J. Zhang, "Chatgpt: potential, prospects, and limitations," in *Frontiers of Information Technology & Electronic Engineering*, 2023, pp. 1–6.

[54] W. X. Zhao, J. Liu, R. Ren, and J. Wen, "Dense text retrieval based on pretrained language models: A survey," *CoRR*, vol. abs/2211.14876, 2022.

[55] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020.

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

[57] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," *CoRR*, 2023.

[58] B. A. Huberman and T. Hogg, "Phase transitions in artificial intelligence systems," *Artificial Intelligence*, vol. 33, no. 2, pp. 155–171, 1987.

[59] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, H. F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, E. Rutherford, T. Hennigan, J. Menick, A. Cassirer, R. Powell, G. van den Driessche, L. A. Hendricks, M. Rauh, P. Huang, A. Glaese, J. Welbl, S. Dathathri, S. Huang, J. Uesato, J. Mellor, I. Higgins, A. Creswell, N. McAleese, A. Wu, E. Elsen, S. M. Jayakumar, E. Buchatskaya, D. Budden, E. Sutherland, K. Simonyan, M. Paganini, L. Sifre, L. Martens, X. L. Li, A. Kuncoro, A. Nematzadeh, E. Gribovskaya, D. Donato, A. Lazaridou, A. Mensch, J. Lespiau, M. Tsimpoukelli, N. Grigorev, D. Fritz, T. Sottiaux, M. Pajarskas, T. Pohlen, Z. Gong, D. Toyama, C. de Masson d'Autume, Y. Li, T. Terzi, V. Mikulik, I. Babuschkin, A. Clark, D. de Las Casas, A. Guy, C. Jones, J. Bradbury, M. J. Johnson, B. A. Hechtman, L. Weidinger, I. Gabriel, W. S. Isaac, E. Lockhart, S. Osindero, L. Rimell, C. Dyer, O. Vinyals, K. Ayoub, J. Stanway, L. Bennett, D. Hassabis, K. Kavukcuoglu, and G. Irving, "Scaling language models: Methods, analysis & insights from training gopher," *CoRR*, vol. abs/2112.11446, 2021.

[60] D. Dai, Y. Sun, L. Dong, Y. Hao, Z. Sui, and F. Wei, "Why can GPT learn in-context? language models secretly perform gradient descent as meta-optimizers," *CoRR*, vol. abs/2212.10559, 2022.

[61] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," *CoRR*, vol. abs/2203.02155, 2022.

[62] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[63] R. Thoppilan, D. D. Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, Y. Li, H. Lee, H. S. Zheng, A. Ghafouri, M. Menegali, Y. Huang, M. Krikun, D. Lepikhin, J. Qin, D. Chen, Y. Xu, Z. Chen, A. Roberts, M. Bosma, Y. Zhou, C. Chang, I. Krivokon, W. Rusch, M. Pickett, K. S. Meier-Hellstern, M. R. Morris, T. Doshi, R. D. Santos, T. Duke, J. Soraker, B. Zevenbergen, V. Prabhakaran, M. Diaz, B. Hutchinson, K. Olson, A. Molina, E. Hoffman-John, J. Lee, L. Aroyo, R. Rajakumar, A. Butryna, M. Lamm, V. Kuzmina, J. Fenton, A. Cohen, R. Bernstein, R. Kurzweil, B. Aguera-Arcas, C. Cui, M. Croak, E. H. Chi, and Q. Le, "Lamda: Language models for dialog applications," *CoRR*, vol. abs/2201.08239, 2022.

[64] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay,

W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S. S. Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, S. Narang, G. Mishra, A. Yu, V. Y. Zhao, Y. Huang, A. M. Dai, H. Yu, S. Petrov, E. H. Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou, Q. V. Le, and J. Wei, "Scaling instruction-finetuned language models," *CoRR*, vol. abs/2210.11416, 2022.

[65] J. Rasley, S. Rajbhandari, O. Ruwase, and Y. He, "Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters," in *KDD*, 2020, pp. 3505–3506.

[66] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-lm: Training multi-billion parameter language models using model parallelism," *CoRR*, vol. abs/1909.08053, 2019.

[67] D. Narayanan, M. Shoeybi, J. Casper, P. LeGresley, M. Patwary, V. Korthikanti, D. Vainbrand, P. Kashinkunti, J. Bernauer, B. Catanzaro, A. Phanishayee, and M. Zaharia, "Efficient large-scale language model training on GPU clusters using megatron-lm," in *International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2021, St. Louis, Missouri, USA, November 14-19, 2021*. ACM, 2021, p. 58.

[68] V. Korthikanti, J. Casper, S. Lym, L. McAfee, M. Andersch, M. Shoeybi, and B. Catanzaro, "Reducing activation recomputation in large transformer models," *CoRR*, vol. abs/2205.05198, 2022.

[69] T. L. Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilic, D. Hesslow, R. Castagné, A. S. Luccioni, F. Yvon, M. Gallé, J. Tow, A. M. Rush, S. Biderman, A. Webson, P. S. Ammanamanchi, T. Wang, B. Sagot, N. Muennighoff, A. V. del Moral, O. Ruwase, R. Bawden, S. Bekman, A. McMillan-Major, I. Beltagy, H. Nguyen, L. Saulnier, S. Tan, P. O. Suarez, V. Sanh, H. Laurençon, Y. Jernite, J. Launay, M. Mitchell, C. Raffel, A. Gokaslan, A. Simhi, A. Soroa, A. F. Aji, A. Alfassy, A. Rogers, A. K. Nitzav, C. Xu, C. Mou, C. Emezue, C. Klamm, C. Leong, D. van Strien, D. I. Adelani, and et al., "BLOOM: A 176b-parameter open-access multilingual language model," *CoRR*, vol. abs/2211.05100, 2022.

[70] P. F. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 4299–4307.

[71] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," *CoRR*, vol. abs/2302.04761, 2023.

[72] R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, X. Jiang, K. Cobbe, T. Eloundou, G. Krueger, K. Button, M. Knight, B. Chess, and J. Schulman, "Webgpt: Browser-assisted question-answering with human feedback," *CoRR*, vol. abs/2112.09332, 2021.

[73] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, pp. 140:1–140:67, 2020.

[74] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel, "mt5: A massively multilingual pre-trained text-to-text transformer," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, 2021, pp. 483–498.

[75] W. Zeng, X. Ren, T. Su, H. Wang, Y. Liao, Z. Wang, X. Jiang, Z. Yang, K. Wang, X. Zhang, C. Li, Z. Gong, Y. Yao, X. Huang, J. Wang, J. Yu, Q. Guo, Y. Yu, Y. Zhang, J. Wang, H. Tao, D. Yan, Z. Yi, F. Peng, F. Jiang, H. Zhang, L. Deng, Y. Zhang, Z. Lin, C. Zhang, S. Zhang, M. Guo, S. Gu, G. Fan, Y. Wang, X. Jin, Q. Liu, and Y. Tian, "Pangu-α: Large-scale autoregressive pretrained chinese language models with auto-parallel computation," *CoRR*, vol. abs/2104.12369, 2021.

[76] Z. Zhang, Y. Gu, X. Han, S. Chen, C. Xiao, Z. Sun, Y. Yao, F. Qi, J. Guan, P. Ke, Y. Cai, G. Zeng, Z. Tan, Z. Liu, M. Huang, W. Han, Y. Liu, X. Zhu, and M. Sun, "CPM-2: large-scale cost-effective pre-trained language models," *CoRR*, vol. abs/2106.10715, 2021.

[77] E. Nijkamp, B. Pang, H. Hayashi, L. Tu, H. Wang, Y. Zhou, S. Savarese, and C. Xiong, "Codegen: An open large language model for code with mtulti-turn program synthesis," *arXiv preprint arXiv:2203.13474*, 2022.

[78] S. Black, S. Biderman, E. Hallahan, Q. Anthony, L. Gao, L. Golding, H. He, C. Leahy, K. McDonell, J. Phang, M. Pieler, U. S. Prashanth, S. Purohit, L. Reynolds, J. Tow, B. Wang, and S. Weinbach, "Gpt-neox-20b: An open-source autoregressive language model," *CoRR*, vol. abs/2204.06745, 2022.

[79] Y. Wang, S. Mishra, P. Alipoormolabashi, Y. Kordi, A. Mirzaei, A. Naik, A. Ashok, A. S. Dhanasekaran, A. Arunkumar, D. Stap, E. Pathak, G. Karamanolakis, H. G. Lai, I. Purohit, I. Mondal, J. Anderson, K. Kuznia, K. Doshi, K. K. Pal, M. Patel, M. Moradshahi, M. Parmar, M. Purohit, N. Varshney, P. R. Kaza, P. Verma, R. S. Puri, R. Karia, S. Doshi, S. K. Sampat, S. Mishra, S. R. A, S. Patro, T. Dixit, and X. Shen, "Super-naturalinstructions: Generalization via declarative instructions on 1600+ NLP tasks," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, 2022, pp. 5085–5109.

[80] Y. Tay, M. Dehghani, V. Q. Tran, X. García, J. Wei, X. Wang, H. W. Chung, D. Bahri, T. Schuster, H. Zheng, D. Zhou, N. Houlsby, and D. Metzler, "Ul2: Unifying language learning paradigms," 2022.

[81] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. T. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer,

"OPT: open pre-trained transformer language models," *CoRR*, vol. abs/2205.01068, 2022.

[82] M. R. Costa-jussà, J. Cross, O. Çelebi, M. Elbayad, K. Heafield, K. Heffernan, E. Kalbassi, J. Lam, D. Licht, J. Maillard, A. Sun, S. Wang, G. Wenzek, A. Youngblood, B. Akula, L. Barrault, G. M. Gonzalez, P. Hansanti, J. Hoffman, S. Jarrett, K. R. Sadagopan, D. Rowe, S. Spruit, C. Tran, P. Andrews, N. F. Ayan, S. Bhosale, S. Edunov, A. Fan, C. Gao, V. Goswami, F. Guzmán, P. Koehn, A. Mourachko, C. Ropers, S. Saleem, H. Schwenk, and J. Wang, "No language left behind: Scaling human-centered machine translation," *CoRR*, vol. abs/2207.04672, 2022.

[83] Q. Zheng, X. Xia, X. Zou, Y. Dong, S. Wang, Y. Xue, Z. Wang, L. Shen, A. Wang, Y. Li *et al.*, "Codegeex: A pre-trained model for code generation with multilingual evaluations on humaneval-x," *arXiv preprint arXiv:2303.17568*, 2023.

[84] A. Zeng, X. Liu, Z. Du, Z. Wang, H. Lai, M. Ding, Z. Yang, Y. Xu, W. Zheng, X. Xia, W. L. Tam, Z. Ma, Y. Xue, J. Zhai, W. Chen, P. Zhang, Y. Dong, and J. Tang, "GLM-130B: an open bilingual pre-trained model," vol. abs/2210.02414, 2022.

[85] N. Muennighoff, T. Wang, L. Sutawika, A. Roberts, S. Biderman, T. L. Scao, M. S. Bari, S. Shen, Z. X. Yong, H. Schoelkopf, X. Tang, D. Radev, A. F. Aji, K. Almubarak, S. Albanie, Z. Alyafeai, A. Webson, E. Raff, and C. Raffel, "Crosslingual generalization through multitask finetuning," *CoRR*, vol. abs/2211.01786, 2022.

[86] S. Iyer, X. V. Lin, R. Pasunuru, T. Mihaylov, D. Simig, P. Yu, K. Shuster, T. Wang, Q. Liu, P. S. Koura, X. Li, B. O'Horo, G. Pereyra, J. Wang, C. Dewan, A. Celikyilmaz, L. Zettlemoyer, and V. Stoyanov, "OPT-IML: scaling language model instruction meta learning through the lens of generalization," *CoRR*, vol. abs/2212.12017, 2022.

[87] S. Biderman, H. Schoelkopf, Q. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. A. Khan, S. Purohit, U. S. Prashanth, E. Raff *et al.*, "Pythia: A suite for analyzing large language models across training and scaling," *arXiv preprint arXiv:2304.01373*, 2023.

[88] E. Nijkamp, H. Hayashi, C. Xiong, S. Savarese, and Y. Zhou, "Codegen2: Lessons for training llms on programming and natural languages," *CoRR*, vol. abs/2305.02309, 2023.

[89] R. Li, L. B. Allal, Y. Zi, N. Muennighoff, D. Kocetkov, C. Mou, M. Marone, C. Akiki, J. Li, J. Chim, Q. Liu, E. Zheltonozhskii, T. Y. Zhuo, T. Wang, O. Dehaene, M. Davaadorj, J. Lamy-Poirier, J. Monteiro, O. Shliazhko, N. Gontier, N. Meade, A. Zebaze, M. Yee, L. K. Umapathi, J. Zhu, B. Lipkin, M. Oblokulov, Z. Wang, R. M. V, J. Stillerman, S. S. Patel, D. Abulkhanov, M. Zocca, M. Dey, Z. Zhang, N. Fahmy, U. Bhattacharyya, W. Yu, S. Singh, S. Luccioni, P. Villegas, M. Kunakov, F. Zhdanov, M. Romero, T. Lee, N. Timor, J. Ding, C. Schlesinger, H. Schoelkopf, J. Ebert, T. Dao, M. Mishra, A. Gu, J. Robinson, C. J. Anderson, B. Dolan-Gavitt, D. Contractor, S. Reddy, D. Fried, D. Bahdanau, Y. Jernite, C. M. Ferrandis, S. Hughes, T. Wolf, A. Guha, L. von Werra, and H. de Vries, "Starcoder: may the source be with you!" *CoRR*, vol. abs/2305.06161, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2305.06161

[90] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[91] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, and Z. Chen, "Gshard: Scaling giant models with conditional computation and automatic sharding," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

[92] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba, "Evaluating large language models trained on code," *CoRR*, vol. abs/2107.03374, 2021.

[93] Y. Sun, S. Wang, S. Feng, S. Ding, C. Pang, J. Shang, J. Liu, X. Chen, Y. Zhao, Y. Lu, W. Liu, Z. Wu, W. Gong, J. Liang, Z. Shang, P. Sun, W. Liu, X. Ouyang, D. Yu, H. Tian, H. Wu, and H. Wang, "ERNIE 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation," *CoRR*, vol. abs/2107.02137, 2021.

[94] O. Lieber, O. Sharir, B. Lenz, and Y. Shoham, "Jurassic-1: Technical details and evaluation," *White Paper. AI21 Labs*, vol. 1, 2021.

[95] B. Kim, H. Kim, S. Lee, G. Lee, D. Kwak, D. H. Jeon, S. Park, S. Kim, S. Kim, D. Seo, H. Lee, M. Jeong, S. Lee, M. Kim, S. Ko, S. Kim, T. Park, J. Kim, S. Kang, N. Ryu, K. M. Yoo, M. Chang, S. Suh, S. In, J. Park, K. Kim, H. Kim, J. Jeong, Y. G. Yeo, D. Ham, D. Park, M. Y. Lee, J. Kang, I. Kang, J. Ha, W. Park, and N. Sung, "What changes can large-scale language models bring? intensive study on hyperclova: Billions-scale korean generative pretrained transformers," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*. Association for Computational Linguistics, 2021.

[96] S. Wu, X. Zhao, T. Yu, R. Zhang, C. Shen, H. Liu, F. Li, H. Zhu, J. Luo, L. Xu *et al.*, "Yuan 1.0: Large-scale pre-trained language model in zero-shot and few-shot learning," *arXiv preprint arXiv:2110.04725*, 2021.

[97] A. Askell, Y. Bai, A. Chen, D. Drain, D. Ganguli, T. Henighan, A. Jones, N. Joseph, B. Mann, N. DasSarma, N. Elhage, Z. Hatfield-Dodds, D. Hernandez,

J. Kernion, K. Ndousse, C. Olsson, D. Amodei, T. B. Brown, J. Clark, S. McCandlish, C. Olah, and J. Kaplan, "A general language assistant as a laboratory for alignment," *CoRR*, vol. abs/2112.00861, 2021.

[98] S. Wang, Y. Sun, Y. Xiang, Z. Wu, S. Ding, W. Gong, S. Feng, J. Shang, Y. Zhao, C. Pang, J. Liu, X. Chen, Y. Lu, W. Liu, X. Wang, Y. Bai, Q. Chen, L. Zhao, S. Li, P. Sun, D. Yu, Y. Ma, H. Tian, H. Wu, T. Wu, W. Zeng, G. Li, W. Gao, and H. Wang, "ERNIE 3.0 titan: Exploring larger-scale knowledge enhanced pre-training for language understanding and generation," *CoRR*, vol. abs/2112.12731, 2021.

[99] N. Du, Y. Huang, A. M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou, A. W. Yu, O. Firat, B. Zoph, L. Fedus, M. P. Bosma, Z. Zhou, T. Wang, Y. E. Wang, K. Webster, M. Pellat, K. Robinson, K. S. Meier-Hellstern, T. Duke, L. Dixon, K. Zhang, Q. V. Le, Y. Wu, Z. Chen, and C. Cui, "Glam: Efficient scaling of language models with mixture-of-experts," in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, 2022, pp. 5547–5569.

[100] S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhumoye, G. Zerveas, V. Korthikanti, E. Zheng, R. Child, R. Y. Aminabadi, J. Bernauer, X. Song, M. Shoeybi, Y. He, M. Houston, S. Tiwary, and B. Catanzaro, "Using deepspeed and megatron to train megatron-turing NLG 530b, A large-scale generative language model," *CoRR*, vol. abs/2201.11990, 2022.

[101] Y. Li, D. H. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. D. Lago, T. Hubert, P. Choy, C. de Masson d'Autume, I. Babuschkin, X. Chen, P. Huang, J. Welbl, S. Gowal, A. Cherepanov, J. Molloy, D. J. Mankowitz, E. S. Robson, P. Kohli, N. de Freitas, K. Kavukcuoglu, and O. Vinyals, "Competition-level code generation with alphacode," *Science*, 2022.

[102] S. Soltan, S. Ananthakrishnan, J. FitzGerald, R. Gupta, W. Hamza, H. Khan, C. Peris, S. Rawls, A. Rosenbaum, A. Rumshisky, C. S. Prakash, M. Sridhar, F. Triefenbach, A. Verma, G. Tür, and P. Natarajan, "Alexatm 20b: Few-shot learning using a large-scale multilingual seq2seq model," *CoRR*, vol. abs/2208.01448, 2022.

[103] A. Glaese, N. McAleese, M. Trebacz, J. Aslanides, V. Firoiu, T. Ewalds, M. Rauh, L. Weidinger, M. Chadwick, P. Thacker, L. Campbell-Gillingham, J. Uesato, P. Huang, R. Comanescu, F. Yang, A. See, S. Dathathri, R. Greig, C. Chen, D. Fritz, J. S. Elias, R. Green, S. Mokrá, N. Fernando, B. Wu, R. Foley, S. Young, I. Gabriel, W. Isaac, J. Mellor, D. Hassabis, K. Kavukcuoglu, L. A. Hendricks, and G. Irving, "Improving alignment of dialogue agents via targeted human judgements," *CoRR*, vol. abs/2209.14375, 2022.

[104] H. Su, X. Zhou, H. Yu, Y. Chen, Z. Zhu, Y. Yu, and J. Zhou, "Welm: A well-read pre-trained language model for chinese," *CoRR*, vol. abs/2209.10372, 2022.

[105] Y. Tay, J. Wei, H. W. Chung, V. Q. Tran, D. R. So, S. Shakeri, X. Garcia, H. S. Zheng, J. Rao, A. Chowdhery, D. Zhou, D. Metzler, S. Petrov, N. Houlsby, Q. V. Le, and M. Dehghani, "Transcending scaling laws with 0.1% extra compute," *CoRR*, vol. abs/2210.11399, 2022.

[106] X. Ren, P. Zhou, X. Meng, X. Huang, Y. Wang, W. Wang, P. Li, X. Zhang, A. Podolskiy, G. Arshinov, A. Bout, I. Piontkovskaya, J. Wei, X. Jiang, T. Su, Q. Liu, and J. Yao, "Pangu-Σ: Towards trillion parameter language model with sparse heterogeneous computing," *CoRR*, vol. abs/2303.10845, 2023.

[107] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen *et al.*, "Palm 2 technical report," *arXiv preprint arXiv:2305.10403*, 2023.

[108] A. Radford, R. Józefowicz, and I. Sutskever, "Learning to generate reviews and discovering sentiment," *CoRR*, vol. abs/1704.01444, 2017.

[109] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[110] B. McCann, N. S. Keskar, C. Xiong, and R. Socher, "The natural language decathlon: Multitask learning as question answering," *CoRR*, vol. abs/1806.08730, 2018.

[111] Y. Zhang, S. Sun, M. Galley, Y. Chen, C. Brockett, X. Gao, J. Gao, J. Liu, and B. Dolan, "DIALOGPT : Large-scale generative pre-training for conversational response generation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ACL 2020, Online, July 5-10, 2020*, A. Celikyilmaz and T. Wen, Eds. Association for Computational Linguistics, 2020, pp. 270–278.

[112] D. Ham, J. Lee, Y. Jang, and K. Kim, "End-to-end neural pipeline for goal-oriented dialogue systems using GPT-2," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*. Association for Computational Linguistics, 2020, pp. 583–592.

[113] I. Drori, S. Tran, R. Wang, N. Cheng, K. Liu, L. Tang, E. Ke, N. Singh, T. L. Patti, J. Lynch, A. Shporer, N. Verma, E. Wu, and G. Strang, "A neural network solves and generates mathematics problems by program synthesis: Calculus, differential equations, linear algebra, and more," *CoRR*, vol. abs/2112.15594, 2021.

[114] A. Neelakantan, T. Xu, R. Puri, A. Radford, J. M. Han, J. Tworek, Q. Yuan, N. Tezak, J. W. Kim, C. Hallacy, J. Heidecke, P. Shyam, B. Power, T. E. Nekoul, G. Sastry, G. Krueger, D. Schnurr, F. P. Such, K. Hsu, M. Thompson, T. Khan, T. Sherbakov, J. Jang, P. Welinder, and L. Weng, "Text and code embeddings by contrastive pre-training," *CoRR*, vol. abs/2201.10005, 2022.

[115] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[116] N. Stiennon, L. Ouyang, J. Wu, D. M. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. F. Christiano, "Learning to summarize from human feedback," *CoRR*, vol. abs/2009.01325, 2020.

[117] OpenAI, "Our approach to alignment research," *OpenAI Blog*, August 2022.

[118] ——, "Introducing chatgpt," *OpenAI Blog*, November

2022.

[119] D. Ganguli, L. Lovitt, J. Kernion, A. Askell, Y. Bai, S. Kadavath, B. Mann, E. Perez, N. Schiefer, K. Ndousse, A. Jones, S. Bowman, A. Chen, T. Conerly, N. DasSarma, D. Drain, N. Elhage, S. E. Showk, S. Fort, Z. Hatfield-Dodds, T. Henighan, D. Hernandez, T. Hume, J. Jacobson, S. Johnston, S. Kravec, C. Olsson, S. Ringer, E. Tran-Johnson, D. Amodei, T. Brown, N. Joseph, S. McCandlish, C. Olah, J. Kaplan, and J. Clark, "Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned," *CoRR*, vol. abs/2209.07858, 2022.

[120] OpenAI, "Lessons learned on language model safety and misuse," *OpenAI Blog*, March 2022.

[121] E. Almazrouei, H. Alobeidli, A. Alshamsi, A. Cappelli, R. Cojocaru, M. Debbah, E. Goffinet, D. Heslow, J. Launay, Q. Malartic, B. Noune, B. Pannier, and G. Penedo, "Falcon-40B: an open large language model with state-of-the-art performance," 2023.

[122] L. Huawei Technologies Co., "Huawei mindspore ai development framework," in *Artificial Intelligence Technology*. Springer, 2022, pp. 137–162.

[123] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, "Stanford alpaca: An instruction-following llama model," https://github.com/tatsu-lab/stanford_alpaca, 2023.

[124] 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, "Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality," 2023. [Online]. Available: https://vicuna.lmsys.org

[125] 2023. [Online]. Available: https://github.com/nebuly-ai/nebullvm/tree/main/apps/accelerate/chatllama

[126] Y. You, "Colossalchat: An open-source solution for cloning chatgpt with a complete rlhf pipeline," 2023. [Online]. Available: https://medium.com/@yangyou_berkeley/colossalchat-an-open-source-solution-for-cloning-chatgpt-with-a-complete-rlhf-pipeline-5edf08fb538b

[127] G. Penedo, Q. Malartic, D. Hesslow, R. Cojocaru, A. Cappelli, H. Alobeidli, B. Pannier, E. Almazrouei, and J. Launay, "The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only," *arXiv preprint arXiv:2306.01116*, 2023.

[128] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, "Stanford alpaca: An instruction-following llama model," https://github.com/tatsu-lab/stanford_alpaca, 2023.

[129] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi, "Self-instruct: Aligning language model with self generated instructions," *CoRR*, vol. abs/2212.10560, 2022.

[130] Alpaca-LoRA, "Instruct-tune llama on consumer hardware," https://github.com/tloen/alpaca-lora, 2023.

[131] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[132] X. Geng, A. Gudibande, H. Liu, E. Wallace, P. Abbeel, S. Levine, and D. Song, "Koala: A dialogue model for academic research," Blog post, April 2023.

[133] Y. Ji, Y. Deng, Y. Gong, Y. Peng, Q. Niu, B. Ma, and X. Li, "Belle: Be everyone's large language model engine," https://github.com/LianjiaTech/BELLE, 2023.

[134] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," *CoRR*, vol. abs/2304.08485, 2023.

[135] D. Zhu, J. Chen, X. Shen, X. Li, and M. Elhoseiny, "Minigpt-4: Enhancing vision-language understanding with advanced large language models," *CoRR*, vol. abs/2304.10592, 2023.

[136] W. Dai, J. Li, D. Li, A. M. H. Tiong, J. Zhao, W. Wang, B. Li, P. Fung, and S. C. H. Hoi, "Instructblip: Towards general-purpose vision-language models with instruction tuning," *CoRR*, vol. abs/2305.06500, 2023.

[137] Y. Su, T. Lan, H. Li, J. Xu, Y. Wang, and D. Cai, "Pandagpt: One model to instruction-follow them all," 2023.

[138] Y. Zhu, R. Kiros, R. S. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, 2015, pp. 19–27.

[139] "Project gutenberg." [Online]. Available: https://www.gutenberg.org/

[140] T. H. Trinh and Q. V. Le, "A simple method for commonsense reasoning," *CoRR*, vol. abs/1806.02847, 2018.

[141] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi, "Defending against neural fake news," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 9051–9062.

[142] A. Gokaslan, V. C. E. Pavlick, and S. Tellex, "Openwebtext corpus," http://Skylion007.github.io/OpenWebTextCorpus, 2019.

[143] J. Baumgartner, S. Zannettou, B. Keegan, M. Squire, and J. Blackburn, "The pushshift reddit dataset," in *Proceedings of the Fourteenth International AAAI Conference on Web and Social Media, ICWSM 2020, Held Virtually, Original Venue: Atlanta, Georgia, USA, June 8-11, 2020*. AAAI Press, 2020, pp. 830–839.

[144] "Wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Main_Page

[145] "Bigquery dataset." [Online]. Available: https://cloud.google.com/bigquery?hl=zh-cn

[146] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, S. Presser, and C. Leahy, "The pile: An 800gb dataset of diverse text for language modeling," *CoRR*, vol. abs/2101.00027, 2021.

[147] H. Laurençon, L. Saulnier, T. Wang, C. Akiki, A. V. del Moral, T. Le Scao, L. Von Werra, C. Mou, E. G.

Ponferrada, H. Nguyen *et al.*, "The bigscience roots corpus: A 1.6 tb composite multilingual dataset," in *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[148] "Common crawl." [Online]. Available: https://commoncrawl.org/

[149] "A reproduction version of cc-stories on hugging face." [Online]. Available: https://huggingface.co/datasets/spacemanidol/cc-stories

[150] B. Wang and A. Komatsuzaki, "GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model," https://github.com/kingoflolz/mesh-transformer-jax, 2021.

[151] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*. Association for Computational Linguistics, 2020, pp. 38–45.

[152] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/google/jax

[153] Z. Bian, H. Liu, B. Wang, H. Huang, Y. Li, C. Wang, F. Cui, and Y. You, "Colossal-ai: A unified deep learning system for large-scale parallel training," *CoRR*, vol. abs/2110.14883, 2021.

[154] J. Fang, Y. Yu, S. Li, Y. You, and J. Zhou, "Patrickstar: Parallel training of pre-trained models via a chunk-based memory management," *CoRR*, vol. abs/2108.05818, 2021.

[155] "Bmtrain: Effient training for big models." [Online]. Available: https://github.com/OpenBMB/BMTrain

[156] J. He, J. Qiu, A. Zeng, Z. Yang, J. Zhai, and J. Tang, "Fastmoe: A fast mixture-of-expert training system," *CoRR*, vol. abs/2103.13262, 2021.

[157] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 8024–8035.

[158] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. A. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016*, K. Keeton and T. Roscoe, Eds. USENIX Association, 2016, pp. 265–283.

[159] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," *CoRR*, vol. abs/1512.01274, 2015.

[160] Y. Ma, D. Yu, T. Wu, and H. Wang, "Paddlepaddle: An open-source deep learning platform from industrial practice," *Frontiers of Data and Domputing*, vol. 1, no. 1, p. 105, 2019.

[161] J. Yuan, X. Li, C. Cheng, J. Liu, R. Guo, S. Cai, C. Yao, F. Yang, X. Yi, C. Wu, H. Zhang, and J. Zhao, "Oneflow: Redesign the distributed deep learning framework from scratch," *CoRR*, vol. abs/2110.15032, 2021.

[162] S. Roller, E. Dinan, N. Goyal, D. Ju, M. Williamson, Y. Liu, J. Xu, M. Ott, E. M. Smith, Y. Boureau, and J. Weston, "Recipes for building an open-domain chatbot," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, 2021, pp. 300–325.

[163] A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo, Y. Wu, B. Neyshabur, G. Gur-Ari, and V. Misra, "Solving quantitative reasoning problems with language models," *CoRR*, vol. abs/2206.14858, 2022.

[164] T. Saier, J. Krause, and M. Färber, "unarxive 2022: All arxiv publications pre-processed for nlp, including structured full-text and citation network," *arXiv preprint arXiv:2303.14957*, 2023.

[165] H. A. Simon, "Experiments with a heuristic compiler," *J. ACM*, vol. 10, no. 4, pp. 493–506, 1963.

[166] Z. Manna and R. J. Waldinger, "Toward automatic program synthesis," *Commun. ACM*, vol. 14, no. 3, pp. 151–165, 1971.

[167] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, and M. Zhou, "Codebert: A pre-trained model for programming and natural languages," in *Findings of EMNLP*, 2020.

[168] 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, "Program synthesis with large language models," *CoRR*, vol. abs/2108.07732, 2021.

[169] S. Black, L. Gao, P. Wang, C. Leahy, and S. Biderman, "GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow," 2021.

[170] F. F. Xu, U. Alon, G. Neubig, and V. J. Hellendoorn, "A systematic evaluation of large language models of code," in *MAPS@PLDI*, 2022.

[171] A. Madaan, S. Zhou, U. Alon, Y. Yang, and G. Neubig, "Language models of code are few-shot commonsense learners," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds. Association for Computational Linguistics, 2022, pp. 1384–1403.

[172] S. Longpre, G. Yauney, E. Reif, K. Lee, A. Roberts, B. Zoph, D. Zhou, J. Wei, K. Robinson, D. Mimno *et al.*,

"A pretrainer's guide to training data: Measuring the effects of data age, domain coverage, quality, & toxicity," *arXiv preprint arXiv:2305.13169*, 2023.

[173] D. Chen, Y. Huang, Z. Ma, H. Chen, X. Pan, C. Ge, D. Gao, Y. Xie, Z. Liu, J. Gao, Y. Li, B. Ding, and J. Zhou, "Data-juicer: A one-stop data processing system for large language models," 2023.

[174] D. Hernandez, T. B. Brown, T. Conerly, N. DasSarma, D. Drain, S. E. Showk, N. Elhage, Z. Hatfield-Dodds, T. Henighan, T. Hume, S. Johnston, B. Mann, C. Olah, C. Olsson, D. Amodei, N. Joseph, J. Kaplan, and S. McCandlish, "Scaling laws and interpretability of learning from repeated data," *CoRR*, vol. abs/2205.10487, 2022.

[175] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[176] K. Lee, D. Ippolito, A. Nystrom, C. Zhang, D. Eck, C. Callison-Burch, and N. Carlini, "Deduplicating training data makes language models better," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, 2022, pp. 8424–8445.

[177] N. Carlini, D. Ippolito, M. Jagielski, K. Lee, F. Tramèr, and C. Zhang, "Quantifying memorization across neural language models," *CoRR*, 2022.

[178] N. Carlini, F. Tramèr, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. B. Brown, D. Song, Ú. Erlingsson, A. Oprea, and C. Raffel, "Extracting training data from large language models," in *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, 2021, pp. 2633–2650.

[179] N. Kandpal, E. Wallace, and C. Raffel, "Deduplicating training data mitigates privacy risks in language models," in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*. PMLR, 2022, pp. 10 697–10 707.

[180] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, C. E. Brodley and A. P. Danyluk, Eds. Morgan Kaufmann, 2001, pp. 282–289.

[181] P. Gage, "A new algorithm for data compression," *C Users Journal*, vol. 12, no. 2, pp. 23–38, 1994.

[182] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016.

[183] M. Schuster and K. Nakajima, "Japanese and korean voice search," in *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2012, pp. 5149–5152.

[184] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," *CoRR*, vol. abs/1609.08144, 2016.

[185] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, I. Gurevych and Y. Miyao, Eds. Association for Computational Linguistics, 2018, pp. 66–75.

[186] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*, E. Blanco and W. Lu, Eds. Association for Computational Linguistics, 2018.

[187] M. Davis and M. Dürst, "Unicode normalization forms," 2001.

[188] D. Paperno, G. Kruszewski, A. Lazaridou, Q. N. Pham, R. Bernardi, S. Pezzelle, M. Baroni, G. Boleda, and R. Fernández, "The LAMBADA dataset: Word prediction requiring a broad discourse context," in *ACL (1)*. The Association for Computer Linguistics, 2016.

[189] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever, "Deep double descent: Where bigger models and more data hurt," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[190] B. Zhang, B. Ghorbani, A. Bapna, Y. Cheng, X. Garcia, J. Shen, and O. Firat, "Examining scaling and transfer of language model architectures for machine translation," in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, 2022, pp. 26 176–26 192.

[191] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H. Hon, "Unified language model pre-training for natural language understanding and generation," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 2019, pp. 13 042–13 054.

[192] A. Clark, D. de Las Casas, A. Guy, A. Mensch, M. Paganini, J. Hoffmann, B. Damoc, B. A. Hechtman, T. Cai, S. Borgeaud, G. van den Driessche, E. Rutherford, T. Hennigan, M. J. Johnson, A. Cassirer, C. Jones, E. Buchatskaya, D. Budden, L. Sifre, S. Osindero, O. Vinyals, M. Ranzato, J. W. Rae, E. Elsen, K. Kavukcuoglu, and K. Simonyan, "Unified scaling laws for routed language models," in *International Conference on Machine Learning, ICML 2022, 17-23 July*

2022, Baltimore, Maryland, USA, 2022, pp. 4057–4086.

[193] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. [Online]. Available: https://openreview.net/forum?id=uYLFoz1vlAC

[194] H. Mehta, A. Gupta, A. Cutkosky, and B. Neyshabur, "Long range language modeling via gated state spaces," *CoRR*, vol. abs/2206.13947, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2206.13947

[195] T. Dao, D. Y. Fu, K. K. Saab, A. W. Thomas, A. Rudra, and C. Ré, "Hungry hungry hippos: Towards language modeling with state space models," *CoRR*, vol. abs/2212.14052, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2212.14052

[196] M. Poli, S. Massaroli, E. Nguyen, D. Y. Fu, T. Dao, S. Baccus, Y. Bengio, S. Ermon, and C. Ré, "Hyena hierarchy: Towards larger convolutional language models," in *ICML*, 2023.

[197] B. Peng, E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, H. Cao, X. Cheng, M. Chung, M. Grella, K. K. G. V., X. He, H. Hou, P. Kazienko, J. Kocon, J. Kong, B. Koptyra, H. Lau, K. S. I. Mantri, F. Mom, A. Saito, X. Tang, B. Wang, J. S. Wind, S. Wozniak, R. Zhang, Z. Zhang, Q. Zhao, P. Zhou, J. Zhu, and R. Zhu, "RWKV: reinventing rnns for the transformer era," *CoRR*, vol. abs/2305.13048, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2305.13048

[198] Y. Sun, L. Dong, S. Huang, S. Ma, Y. Xia, J. Xue, J. Wang, and F. Wei, "Retentive network: A successor to transformer for large language models," *arXiv preprint arXiv:2307.08621*, 2023.

[199] J. T. Smith, A. Warrington, and S. Linderman, "Simplified state space layers for sequence modeling," in *ICLR*, 2023.

[200] A. Orvieto, S. L. Smith, A. Gu, A. Fernando, C. Gulcehre, R. Pascanu, and S. De, "Resurrecting recurrent neural networks for long sequences," in *ICML*, 2023.

[201] M. Ding, Z. Yang, W. Hong, W. Zheng, C. Zhou, D. Yin, J. Lin, X. Zou, Z. Shao, H. Yang, and J. Tang, "Cogview: Mastering text-to-image generation via transformers," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 2021, pp. 19 822–19 835.

[202] L. J. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," vol. abs/1607.06450, 2016.

[203] B. Zhang and R. Sennrich, "Root mean square layer normalization," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 2019, pp. 12 360–12 371.

[204] H. Wang, S. Ma, L. Dong, S. Huang, D. Zhang, and F. Wei, "Deepnet: Scaling transformers to 1, 000 layers," vol. abs/2203.00555, 2022.

[205] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[206] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018*, T. Linzen, G. Chrupala, and A. Alishahi, Eds. Association for Computational Linguistics, 2018, pp. 353–355.

[207] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[208] N. Shazeer, "GLU variants improve transformer," vol. abs/2002.05202, 2020.

[209] J. Su, Y. Lu, S. Pan, B. Wen, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," vol. abs/2104.09864, 2021.

[210] O. Press, N. A. Smith, and M. Lewis, "Train short, test long: Attention with linear biases enables input length extrapolation," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

[211] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, ser. JMLR Workshop and Conference Proceedings, F. R. Bach and D. M. Blei, Eds., vol. 37. JMLR.org, 2015, pp. 448–456. [Online]. Available: http://proceedings.mlr.press/v37/ioffe15.html

[212] S. Narang, H. W. Chung, Y. Tay, L. Fedus, T. Févry, M. Matena, K. Malkan, N. Fiedel, N. Shazeer, Z. Lan, Y. Zhou, W. Li, N. Ding, J. Marcus, A. Roberts, and C. Raffel, "Do transformer modifications transfer across implementations and applications?" in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, 2021, pp. 5758–5773.

[213] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu, "On layer normalization in the transformer architecture," in *ICML*, 2020.

[214] A. Baevski and M. Auli, "Adaptive input representations for neural language modeling," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[215] L. Liu, X. Liu, J. Gao, W. Chen, and J. Han, "Understanding the difficulty of training transformers," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*. Association for Computational Linguistics, 2020, pp. 5747–5763.

[216] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[217] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW,*

Australia, 6-11 August 2017, 2017, pp. 933–941.

[218] T. L. Scao, T. Wang, D. Hesslow, S. Bekman, M. S. Bari, S. Biderman, H. Elsahar, N. Muennighoff, J. Phang, O. Press, C. Raffel, V. Sanh, S. Shen, L. Sutawika, J. Tae, Z. X. Yong, J. Launay, and I. Beltagy, "What language model to train if you have one million GPU hours?" in *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, 2022, pp. 765–782.

[219] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, M. A. Walker, H. Ji, and A. Stent, Eds. Association for Computational Linguistics, 2018, pp. 464–468. [Online]. Available: https://doi.org/10.18653/v1/n18-2074

[220] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, A. Korhonen, D. R. Traum, and L. Màrquez, Eds. Association for Computational Linguistics, 2019, pp. 2978–2988. [Online]. Available: https://doi.org/10.18653/v1/p19-1285

[221] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.

[222] Y. Sun, L. Dong, B. Patra, S. Ma, S. Huang, A. Benhaim, V. Chaudhary, X. Song, and F. Wei, "A length-extrapolatable transformer," *CoRR*, vol. abs/2212.10554, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2212.10554

[223] H. Peng, N. Pappas, D. Yogatama, R. Schwartz, N. A. Smith, and L. Kong, "Random feature attention," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

[224] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontañón, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed, "Big bird: Transformers for longer sequences," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[225] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *CoRR*, vol. abs/1904.10509, 2019.

[226] N. Shazeer, "Fast transformer decoding: One write-head is all you need," *CoRR*, vol. abs/1911.02150, 2019. [Online]. Available: http://arxiv.org/abs/1911.02150

[227] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai, "Gqa: Training generalized multi-query transformer models from multi-head checkpoints," *arXiv preprint arXiv:2305.13245*, 2023.

[228] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Re, "Flashattention: Fast and memory-efficient exact attention with IO-awareness," in *NeurIPS*, 2022.

[229] T. Dao, "Flashattention-2: Faster attention with better parallelism and work partitioning," *arXiv preprint arXiv:2307.08691*, 2023.

[230] "vllm: Easy, fast, and cheap llm serving with pagedattention." [Online]. Available: https://vllm.ai/

[231] K. Murray and D. Chiang, "Correcting length bias in neural machine translation," in *WMT*. Association for Computational Linguistics, 2018, pp. 212–223.

[232] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," in *ICLR*, 2020.

[233] C.-M. U. P. P. D. O. C. SCIENCE, *Speech Understanding Systems. Summary of Results of the Five-Year Research Effort at Carnegie-Mellon University*, 1977.

[234] P. Koehn and R. Knowles, "Six challenges for neural machine translation," in *NMT@ACL*. Association for Computational Linguistics, 2017, pp. 28–39.

[235] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," *CoRR*, vol. abs/1609.08144, 2016.

[236] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," in *ICLR (Poster)*. OpenReview.net, 2018.

[237] A. K. Vijayakumar, M. Cogswell, R. R. Selvaraju, Q. Sun, S. Lee, D. J. Crandall, and D. Batra, "Diverse beam search: Decoding diverse solutions from neural sequence models," *CoRR*, vol. abs/1610.02424, 2016.

[238] A. Fan, M. Lewis, and Y. N. Dauphin, "Hierarchical neural story generation," in *ACL (1)*. Association for Computational Linguistics, 2018, pp. 889–898.

[239] Y. Su, T. Lan, Y. Wang, D. Yogatama, L. Kong, and N. Collier, "A contrastive framework for neural text generation," in *NeurIPS*, 2022.

[240] C. Meister, T. Pimentel, G. Wiher, and R. Cotterell, "Locally typical sampling," *Trans. Assoc. Comput. Linguistics*, 2023.

[241] Y. Leviathan, M. Kalman, and Y. Matias, "Fast inference from transformers via speculative decoding," in *International Conference on Machine Learning*, 2023.

[242] C. Chen, S. Borgeaud, G. Irving, J. Lespiau, L. Sifre, and J. Jumper, "Accelerating large language model decoding with speculative sampling," *CoRR*, vol. abs/2302.01318, 2023.

[243] X. Miao, G. Oliaro, Z. Zhang, X. Cheng, Z. Wang, R. Y. Y. Wong, Z. Chen, D. Arfeen, R. Abhyankar, and Z. Jia, "Specinfer: Accelerating generative LLM serving with speculative inference and token tree verification," *CoRR*, vol. abs/2305.09781, 2023.

[244] L. D. Corro, A. D. Giorno, S. Agarwal, B. Yu, A. H. Awadallah, and S. Mukherjee, "Skipdecode: Autoregressive skip decoding with batching and caching for efficient LLM inference," *CoRR*, vol. abs/2307.02628,

2023.

[245] A. Yuan, A. Coenen, E. Reif, and D. Ippolito, "Wordcraft: story writing with large language models," in *27th International Conference on Intelligent User Interfaces*, 2022, pp. 841–852.

[246] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 5156–5165. [Online]. Available: http://proceedings.mlr.press/v119/katharopoulos20a.html

[247] C. Zhu, W. Ping, C. Xiao, M. Shoeybi, T. Goldstein, A. Anandkumar, and B. Catanzaro, "Long-short transformer: Efficient transformers for language and vision," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021, pp. 17 723–17 736. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/9425be43ba92c2b4454ca7bf602efad8-Abstract.html

[248] K. M. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlós, P. Hawkins, J. Q. Davis, A. Mohiuddin, L. Kaiser, D. B. Belanger, L. J. Colwell, and A. Weller, "Rethinking attention with performers," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. [Online]. Available: https://openreview.net/forum?id=Ua6zuk0WRH

[249] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. [Online]. Available: https://openreview.net/forum?id=rkgNKkHtvB

[250] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[251] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in adam," *CoRR*, vol. abs/1711.05101, 2017.

[252] N. Shazeer and M. Stern, "Adafactor: Adaptive learning rates with sublinear memory cost," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, ser. Proceedings of Machine Learning Research, J. G. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 4603–4611.

[253] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. X. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu, and Z. Chen, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 103–112.

[254] A. Harlap, D. Narayanan, A. Phanishayee, V. Seshadri, N. R. Devanur, G. R. Ganger, and P. B. Gibbons, "Pipedream: Fast and efficient pipeline parallel DNN training," *CoRR*, vol. abs/1806.03377, 2018.

[255] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "Zero: memory optimizations toward training trillion parameter models," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2020, Virtual Event / Atlanta, Georgia, USA, November 9-19, 2020*, C. Cuicchi, I. Qualters, and W. T. Kramer, Eds. IEEE/ACM, 2020, p. 20.

[256] P. Micikevicius, S. Narang, J. Alben, G. F. Diamos, E. Elsen, D. García, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, "Mixed precision training," *CoRR*, vol. abs/1710.03740, 2017.

[257] Q. Xu, S. Li, C. Gong, and Y. You, "An efficient 2d method for training super-large deep learning models," *CoRR*, vol. abs/2104.05343, 2021.

[258] B. Wang, Q. Xu, Z. Bian, and Y. You, "Tesseract: Parallelize the tensor parallelism efficiently," in *Proceedings of the 51st International Conference on Parallel Processing, ICPP 2022, Bordeaux, France, 29 August 2022 - 1 September 2022*. ACM, 2022.

[259] Z. Bian, Q. Xu, B. Wang, and Y. You, "Maximizing parallelism in distributed training for huge neural networks," *CoRR*, vol. abs/2105.14450, 2021.

[260] S. Li, F. Xue, C. Baranwal, Y. Li, and Y. You, "Sequence parallelism: Long sequence training from system perspective," *arXiv e-prints*, pp. arXiv–2105, 2021.

[261] FairScale authors, "Fairscale: A general purpose modular pytorch library for high performance and large scale training," https://github.com/facebookresearch/fairscale, 2021.

[262] L. Zheng, Z. Li, H. Zhang, Y. Zhuang, Z. Chen, Y. Huang, Y. Wang, Y. Xu, D. Zhuo, E. P. Xing *et al.*, "Alpa: Automating inter-and {Intra-Operator} parallelism for distributed deep learning," in *OSDI*, 2022, pp. 559–578.

[263] T. Chen, B. Xu, C. Zhang, and C. Guestrin, "Training deep nets with sublinear memory cost," *CoRR*, vol. abs/1604.06174, 2016.

[264] S. Mishra, D. Khashabi, C. Baral, and H. Hajishirzi, "Cross-task generalization via natural language crowdsourcing instructions," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, S. Muresan, P. Nakov, and A. Villavicencio, Eds., 2022, pp. 3470–3487.

[265] S. H. Bach, V. Sanh, Z. X. Yong, A. Webson, C. Raffel, N. V. Nayak, A. Sharma, T. Kim, M. S. Bari, T. Févry, Z. Alyafeai, M. Dey, A. Santilli, Z. Sun, S. Ben-David, C. Xu, G. Chhablani, H. Wang, J. A. Fries, M. S. AlShaibani, S. Sharma, U. Thakker, K. Almubarak, X. Tang, D. R. Radev, M. T. Jiang, and A. M. Rush, "Promptsource: An integrated development environment and repository for natural language prompts," in *ACL (demo)*. Association for Computational Linguistics, 2022, pp. 93–104.

[266] T. Tang, J. Li, W. X. Zhao, and J. Wen, "MVP: multi-task supervised pre-training for natural language generation," *CoRR*, vol. abs/2206.12131, 2022.

[267] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, N. Joseph, S. Kadavath, J. Kernion, T. Conerly, S. E. Showk, N. Elhage, Z. Hatfield-Dodds, D. Hernandez, T. Hume, S. Johnston, S. Kravec, L. Lovitt, N. Nanda, C. Olsson, D. Amodei, T. B. Brown, J. Clark, S. McCandlish, C. Olah, B. Mann, and J. Kaplan, "Training a helpful and harmless assistant with reinforcement learning from human feedback," *CoRR*, vol. abs/2204.05862, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2204.05862

[268] B. Guo, X. Zhang, Z. Wang, M. Jiang, J. Nie, Y. Ding, J. Yue, and Y. Wu, "How close is chatgpt to human experts? comparison corpus, evaluation, and detection," *arXiv preprint arXiv:2301.07597*, 2023.

[269] A. Köpf, Y. Kilcher, D. von Rütte, S. Anagnostidis, Z.-R. Tam, K. Stevens, A. Barhoum, N. M. Duc, O. Stanley, R. Nagyfi *et al.*, "Openassistant conversations–democratizing large language model alignment," *arXiv preprint arXiv:2304.07327*, 2023.

[270] C. Xu, D. Guo, N. Duan, and J. McAuley, "Baize: An open-source chat model with parameter-efficient tuning on self-chat data," *arXiv preprint arXiv:2304.01196*, 2023.

[271] Y. Ji, Y. Gong, Y. Deng, Y. Peng, Q. Niu, B. Ma, and X. Li, "Towards better instruction following language models for chinese: Investigating the impact of training data and evaluation," *arXiv preprint arXiv:2304.07854*, 2023.

[272] R. Lou, K. Zhang, and W. Yin, "Is prompt all you need? no. A comprehensive and broader view of instruction learning," *CoRR*, vol. abs/2303.10475, 2023.

[273] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding," in *ACL (1)*. Association for Computational Linguistics, 2019, pp. 4487–4496.

[274] A. Aghajanyan, A. Gupta, A. Shrivastava, X. Chen, L. Zettlemoyer, and S. Gupta, "Muppet: Massive multi-task representations with pre-finetuning," in *EMNLP (1)*. Association for Computational Linguistics, 2021, pp. 5799–5811.

[275] S. Longpre, L. Hou, T. Vu, A. Webson, H. W. Chung, Y. Tay, D. Zhou, Q. V. Le, B. Zoph, J. Wei, and A. Roberts, "The flan collection: Designing data and methods for effective instruction tuning," *CoRR*, vol. abs/2301.13688, 2023.

[276] H. Chen, Y. Zhang, Q. Zhang, H. Yang, X. Hu, X. Ma, Y. Yanggong, and J. Zhao, "Maybe only 0.5% data is needed: A preliminary exploration of low training data instruction tuning," *arXiv preprint arXiv:2305.09246*, 2023.

[277] C. Zhou, P. Liu, P. Xu, S. Iyer, J. Sun, Y. Mao, X. Ma, A. Efrat, P. Yu, L. Yu *et al.*, "Lima: Less is more for alignment," *arXiv preprint arXiv:2305.11206*, 2023.

[278] S. Gunasekar, Y. Zhang, J. Aneja, C. C. T. Mendes, A. Del Giorno, S. Gopi, M. Javaheripi, P. Kauffmann, G. de Rosa, O. Saarikivi *et al.*, "Textbooks are all you need," *arXiv preprint arXiv:2306.11644*, 2023.

[279] M. M. Krell, M. Kosec, S. P. Perez, and A. Fitzgibbon, "Efficient sequence packing without cross-contamination: Accelerating large language models without impacting performance," *arXiv preprint arXiv:2107.02027*, 2021.

[280] Y. Cao, Y. Kang, and L. Sun, "Instruction mining: High-quality instruction data selection for large language models," *arXiv preprint arXiv:2307.06290*, 2023.

[281] C. Xu, Q. Sun, K. Zheng, X. Geng, P. Zhao, J. Feng, C. Tao, and D. Jiang, "Wizardlm: Empowering large language models to follow complex instructions," *CoRR*, vol. abs/2304.12244, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2304.12244

[282] K. Singhal, S. Azizi, T. Tu, S. S. Mahdavi, J. Wei, H. W. Chung, N. Scales, A. Tanwani, H. Cole-Lewis, S. Pfohl *et al.*, "Large language models encode clinical knowledge," *arXiv preprint arXiv:2212.13138*, 2022.

[283] J. Zhang, R. Xie, Y. Hou, W. X. Zhao, L. Lin, and J. Wen, "Recommendation as instruction following: A large language model empowered recommendation approach," *CoRR*, vol. abs/2305.07001, 2023.

[284] H. Wang, C. Liu, N. Xi, Z. Qiang, S. Zhao, B. Qin, and T. Liu, "Huatuo: Tuning llama model with chinese medical knowledge," *arXiv preprint arXiv:2304.06975*, 2023.

[285] Q. Huang, M. Tao, Z. An, C. Zhang, C. Jiang, Z. Chen, Z. Wu, and Y. Feng, "Lawyer llama technical report," *arXiv preprint arXiv:2305.15062*, 2023.

[286] S. Wu, O. Irsoy, S. Lu, V. Dabravolski, M. Dredze, S. Gehrmann, P. Kambadur, D. Rosenberg, and G. Mann, "Bloomberggpt: A large language model for finance," *arXiv preprint arXiv:2303.17564*, 2023.

[287] T. Liu and B. K. H. Low, "Goat: Fine-tuned llama outperforms gpt-4 on arithmetic tasks," *arXiv preprint arXiv:2305.14201*, 2023.

[288] Z. Sun, Y. Shen, Q. Zhou, H. Zhang, Z. Chen, D. Cox, Y. Yang, and C. Gan, "Principle-driven self-alignment of language models from scratch with minimal human supervision," *arXiv preprint arXiv:2305.03047*, 2023.

[289] YuLan-Chat-Team, "Yulan-chat: An open-source bilingual chatbot," https://github.com/RUC-GSAI/YuLan-Chat, 2023.

[290] Y. Dubois, X. Li, R. Taori, T. Zhang, I. Gulrajani, J. Ba, C. Guestrin, P. Liang, and T. B. Hashimoto, "Alpacafarm: A simulation framework for methods that learn from human feedback," *CoRR*, vol. abs/2305.14387, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2305.14387

[291] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," in *ICLR*. OpenReview.net, 2021.

[292] A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso, A. Kluska, A. Lewkowycz, A. Agarwal, A. Power, A. Ray, A. Warstadt, A. W. Kocurek, A. Safaya, A. Tazarv, A. Xiang, A. Parrish, A. Nie, A. Hussain, A. Askell, A. Dsouza, A. Rahane, A. S. Iyer, A. Andreassen, A. Santilli, A. Stuhlmüller, A. M. Dai, A. La, A. K. Lampinen, A. Zou, A. Jiang, A. Chen, A. Vuong, A. Gupta, A. Gottardi, A. Norelli,

A. Venkatesh, A. Gholamidavoodi, A. Tabassum, A. Menezes, A. Kirubarajan, A. Mullokandov, A. Sabharwal, A. Herrick, A. Efrat, A. Erdem, A. Karakas, and et al., "Beyond the imitation game: Quantifying and extrapolating the capabilities of language models," *CoRR*, vol. abs/2206.04615, 2022.

[293] Z. Kenton, T. Everitt, L. Weidinger, I. Gabriel, V. Mikulik, and G. Irving, "Alignment of language agents," *CoRR*, vol. abs/2103.14659, 2021.

[294] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. F. Christiano, and G. Irving, "Fine-tuning language models from human preferences," *CoRR*, vol. abs/1909.08593, 2019.

[295] A. Askell, Y. Bai, A. Chen, D. Drain, D. Ganguli, T. Henighan, A. Jones, N. Joseph, B. Mann, N. DasSarma, N. Elhage, Z. Hatfield-Dodds, D. Hernandez, J. Kernion, K. Ndousse, C. Olsson, D. Amodei, T. B. Brown, J. Clark, S. McCandlish, C. Olah, and J. Kaplan, "A general language assistant as a laboratory for alignment," *CoRR*, vol. abs/2112.00861, 2021.

[296] E. Perez, S. Huang, H. F. Song, T. Cai, R. Ring, J. Aslanides, A. Glaese, N. McAleese, and G. Irving, "Red teaming language models with language models," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds. Association for Computational Linguistics, 2022, pp. 3419–3448.

[297] J. Menick, M. Trebacz, V. Mikulik, J. Aslanides, H. F. Song, M. Chadwick, M. Glaese, S. Young, L. Campbell-Gillingham, G. Irving, and N. McAleese, "Teaching language models to support answers with verified quotes," *CoRR*, vol. abs/2203.11147, 2022.

[298] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, C. Chen, C. Olsson, C. Olah, D. Hernandez, D. Drain, D. Ganguli, D. Li, E. Tran-Johnson, E. Perez, J. Kerr, J. Mueller, J. Ladish, J. Landau, K. Ndousse, K. Lukosiute, L. Lovitt, M. Sellitto, N. Elhage, N. Schiefer, N. Mercado, N. DasSarma, R. Lasenby, R. Larson, S. Ringer, S. Johnston, S. Kravec, S. E. Showk, S. Fort, T. Lanham, T. Telleen-Lawton, T. Conerly, T. Henighan, T. Hume, S. R. Bowman, Z. Hatfield-Dodds, B. Mann, D. Amodei, N. Joseph, S. McCandlish, T. Brown, and J. Kaplan, "Constitutional AI: harmlessness from AI feedback," *CoRR*, vol. abs/2212.08073, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2212.08073

[299] H. Dong, W. Xiong, D. Goyal, R. Pan, S. Diao, J. Zhang, K. Shum, and T. Zhang, "RAFT: reward ranked finetuning for generative foundation model alignment," *CoRR*, vol. abs/2304.06767, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2304.06767

[300] A. Askell, Y. Bai, A. Chen, D. Drain, D. Ganguli, T. Henighan, A. Jones, N. Joseph, B. Mann, N. DasSarma *et al.*, "A general language assistant as a laboratory for alignment," *arXiv preprint arXiv:2112.00861*, 2021.

[301] R. Zheng, S. Dou, S. Gao, W. Shen, B. Wang, Y. Liu, S. Jin, Q. Liu, L. Xiong, L. Chen *et al.*, "Secrets of rlhf in large language models part i: Ppo," *arXiv preprint arXiv:2307.04964*, 2023.

[302] R. Liu, C. Jia, G. Zhang, Z. Zhuang, T. X. Liu, and S. Vosoughi, "Second thoughts are best: Learning to re-align with human values from text edits," in *NeurIPS*, 2022.

[303] X. Lu, S. Welleck, J. Hessel, L. Jiang, L. Qin, P. West, P. Ammanabrolu, and Y. Choi, "QUARK: controllable text generation with reinforced unlearning," in *NeurIPS*, 2022.

[304] R. Krishna, D. Lee, L. Fei-Fei, and M. S. Bernstein, "Socially situated artificial intelligence enables learning from human interaction," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 119, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:252381954

[305] R. Liu, R. Yang, C. Jia, G. Zhang, D. Zhou, A. M. Dai, D. Yang, and S. Vosoughi, "Training socially aligned language models in simulated human society," *CoRR*, vol. abs/2305.16960, 2023.

[306] H. Liu, C. Sferrazza, and P. Abbeel, "Chain of hindsight aligns language models with feedback," *CoRR*, vol. abs/2302.02676, 2023.

[307] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," *CoRR*, vol. abs/2305.18290, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2305.18290

[308] H. Dong, W. Xiong, D. Goyal, R. Pan, S. Diao, J. Zhang, K. Shum, and T. Zhang, "RAFT: reward ranked finetuning for generative foundation model alignment," *CoRR*, vol. abs/2304.06767, 2023.

[309] T. Zhang, F. Liu, J. Wong, P. Abbeel, and J. E. Gonzalez, "The wisdom of hindsight makes language models better instruction followers," *CoRR*, vol. abs/2302.05206, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2302.05206

[310] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Comput. Surv.*, vol. 50, no. 2, apr 2017. [Online]. Available: https://doi.org/10.1145/3054912

[311] S. Levine, "Should i imitate or reinforce," 2022. [Online]. Available: https://www.youtube.com/watch?v=sVPm7zOrBxM

[312] J. Schulman, "Reinforcement learning from human feedback: Progress and challenges," 2023. [Online]. Available: https://www.youtube.com/watch?v=hhiLw5Q_UFg

[313] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Association for Computational Linguistics, 2021, pp. 4582–4597.

[314] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Proceedings of the 2021 Conference on Empirical Methods*

in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021, M. Moens, X. Huang, L. Specia, and S. W. Yih, Eds. Association for Computational Linguistics, 2021, pp. 3045–3059.

[315] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for NLP," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, 2019, pp. 2790–2799.

[316] Z. Hu, Y. Lan, L. Wang, W. Xu, E. Lim, R. K. Lee, L. Bing, and S. Poria, "Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models," *CoRR*, vol. abs/2304.01933, 2023.

[317] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig, "Towards a unified view of parameter-efficient transfer learning," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[318] X. Liu, K. Ji, Y. Fu, Z. Du, Z. Yang, and J. Tang, "P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks," *CoRR*, vol. abs/2110.07602, 2021.

[319] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, "GPT understands, too," *CoRR*, vol. abs/2103.10385, 2021.

[320] Y. Gu, X. Han, Z. Liu, and M. Huang, "Ppt: Pre-trained prompt tuning for few-shot learning," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022, pp. 8410–8423.

[321] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig, "How can we know what language models know?" *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 423–438, 2020.

[322] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh, "Autoprompt: Eliciting knowledge from language models with automatically generated prompts," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 4222–4235.

[323] Q. Zhang, M. Chen, A. Bukharin, P. He, Y. Cheng, W. Chen, and T. Zhao, "Adaptive budget allocation for parameter-efficient fine-tuning," *CoRR*, vol. abs/2303.10512, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2303.10512

[324] M. Valipour, M. Rezagholizadeh, I. Kobyzev, and A. Ghodsi, "Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation," *CoRR*, vol. abs/2210.07558, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2210.07558

[325] N. Ding, Y. Qin, G. Yang, F. Wei, Y. Zonghan, Y. Su, S. Hu, Y. Chen, C.-M. Chan, W. Chen, J. Yi, W. Zhao, X. Wang, Z. Liu, H.-T. Zheng, J. Chen, Y. Liu, J. Tang, J. Li, and M. Sun, "Parameter-efficient fine-tuning of large-scale pre-trained language models," *Nature Machine Intelligence*, vol. 5, pp. 1–16, 03 2023.

[326] R. Zhang, J. Han, A. Zhou, X. Hu, S. Yan, P. Lu, H. Li, P. Gao, and Y. Qiao, "Llama-adapter: Efficient fine-tuning of language models with zero-init attention," *CoRR*, vol. abs/2303.16199, 2023.

[327] J. Pfeiffer, I. Vulic, I. Gurevych, and S. Ruder, "MAD-X: an adapter-based framework for multi-task cross-lingual transfer," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Association for Computational Linguistics, 2020, pp. 7654–7673.

[328] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, and S. Paul, "Peft: State-of-the-art parameter-efficient fine-tuning methods," https://github.com/huggingface/peft, 2022.

[329] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," *CoRR*, vol. abs/2103.13630, 2021. [Online]. Available: https://arxiv.org/abs/2103.13630

[330] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Llm.int8(): 8-bit matrix multiplication for transformers at scale," *CoRR*, vol. abs/2208.07339, 2022.

[331] G. Xiao, J. Lin, M. Seznec, J. Demouth, and S. Han, "Smoothquant: Accurate and efficient post-training quantization for large language models," *CoRR*, vol. abs/2211.10438, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2211.10438

[332] Z. Yao, R. Y. Aminabadi, M. Zhang, X. Wu, C. Li, and Y. He, "Zeroquant: Efficient and affordable post-training quantization for large-scale transformers," in *NeurIPS*, 2022.

[333] J. Lin, J. Tang, H. Tang, S. Yang, X. Dang, and S. Han, "Awq: Activation-aware weight quantization for llm compression and acceleration," 2023.

[334] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "Gptq: Accurate post-training quantization for generative pre-trained transformers," *arXiv preprint arXiv:2210.17323*, 2022.

[335] E. Frantar and D. Alistarh, "Optimal brain compression: A framework for accurate post-training quantization and pruning," in *NeurIPS*, 2022.

[336] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," *arXiv preprint arXiv:2305.14314*, 2023.

[337] Z. Liu, B. Oguz, C. Zhao, E. Chang, P. Stock, Y. Mehdad, Y. Shi, R. Krishnamoorthi, and V. Chandra, "Llm-qat: Data-free quantization aware training for large language models," 2023.

[338] Z. Yao, X. Wu, C. Li, S. Youn, and Y. He, "Zeroquant-v2: Exploring post-training quantization in llms from comprehensive study to low rank compensation," 2023.

[339] T. Dettmers and L. Zettlemoyer, "The case for 4-bit precision: k-bit inference scaling laws," *CoRR*, vol. abs/2212.09720, 2022.

[340] L. Peiyu, L. Zikang, G. Ze-Feng, G. Dawei, Z. W. Xin, L. Yaliang, D. Bolin, and W. Ji-Rong, "Do emergent abilities exist in quantized large language models: An empirical study," *arXiv preprint arXiv:2307.08072*, 2023.

[341] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Llm.int8(): 8-bit matrix mul-

tiplication for transformers at scale," *CoRR*, vol. abs/2208.07339, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2208.07339

[342] X. Wei, X. Cui, N. Cheng, X. Wang, X. Zhang, S. Huang, P. Xie, J. Xu, Y. Chen, M. Zhang *et al.*, "Zero-shot information extraction via chatting with chatgpt," *arXiv preprint arXiv:2302.10205*, 2023.

[343] T. Dettmers, M. Lewis, S. Shleifer, and L. Zettlemoyer, "8-bit optimizers via block-wise quantization," *9th International Conference on Learning Representations, ICLR*, 2022.

[344] C. Tao, L. Hou, W. Zhang, L. Shang, X. Jiang, Q. Liu, P. Luo, and N. Wong, "Compression of generative pre-trained language models via quantization," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Association for Computational Linguistics, 2022, pp. 4821–4836.

[345] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, and W. Chen, "What makes good in-context examples for gpt-3?" in *Proceedings of Deep Learning Inside Out: The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures, DeeLIO@ACL 2022, Dublin, Ireland and Online, May 27, 2022*, 2022, pp. 100–114.

[346] O. Rubin, J. Herzig, and J. Berant, "Learning to retrieve prompts for in-context learning," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, 2022, pp. 2655–2671.

[347] H. J. Kim, H. Cho, J. Kim, T. Kim, K. M. Yoo, and S. Lee, "Self-generated in-context learning: Leveraging auto-regressive language models as a demonstration generator," *CoRR*, vol. abs/2206.08082, 2022.

[348] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba, "Large language models are human-level prompt engineers," in *Proc. of ICLR*, 2023.

[349] Y. Hao, Y. Sun, L. Dong, Z. Han, Y. Gu, and F. Wei, "Structured prompting: Scaling in-context learning to 1, 000 examples," *CoRR*, 2022.

[350] Y. Lu, M. Bartolo, A. Moore, S. Riedel, and P. Stenetorp, "Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, S. Muresan, P. Nakov, and A. Villavicencio, Eds., 2022, pp. 8086–8098.

[351] Y. Fu, H. Peng, A. Sabharwal, P. Clark, and T. Khot, "Complexity-based prompting for multi-step reasoning," *CoRR*, vol. abs/2210.00720, 2022.

[352] Z. Zhang, A. Zhang, M. Li, and A. Smola, "Automatic chain of thought prompting in large language models," *CoRR*, vol. abs/2210.03493, 2022.

[353] A. Creswell, M. Shanahan, and I. Higgins, "Selection-inference: Exploiting large language models for interpretable logical reasoning," *CoRR*, vol. abs/2205.09712, 2022.

[354] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," *CoRR*, vol. abs/2203.11171, 2022.

[355] Y. Li, Z. Lin, S. Zhang, Q. Fu, B. Chen, J. Lou, and W. Chen, "On the advance of making language models better reasoners," *CoRR*, vol. abs/2206.02336, 2022.

[356] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, and D. Zhou, "Rationale-augmented ensembles in language models," *CoRR*, 2022.

[357] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, O. Bousquet, Q. Le, and E. H. Chi, "Least-to-most prompting enables complex reasoning in large language models," *CoRR*, vol. abs/2205.10625, 2022.

[358] T. Khot, H. Trivedi, M. Finlayson, Y. Fu, K. Richardson, P. Clark, and A. Sabharwal, "Decomposed prompting: A modular approach for solving complex tasks," *CoRR*, vol. abs/2210.02406, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2210.02406

[359] L. Wang, W. Xu, Y. Lan, Z. Hu, Y. Lan, R. K. Lee, and E. Lim, "Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models," *CoRR*, vol. abs/2305.04091, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2305.04091

[360] Q. Lyu, S. Havaldar, A. Stein, L. Zhang, D. Rao, E. Wong, M. Apidianaki, and C. Callison-Burch, "Faithful chain-of-thought reasoning," *CoRR*, vol. abs/2301.13379, 2023.

[361] L. Gao, A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, J. Callan, and G. Neubig, "PAL: program-aided language models," *CoRR*, vol. abs/2211.10435, 2022.

[362] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang, "Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface," *arXiv preprint arXiv:2303.17580*, 2023.

[363] H. Sun, Y. Zhuang, L. Kong, B. Dai, and C. Zhang, "Adaplanner: Adaptive planning from feedback with language models," *arXiv preprint arXiv:2305.16653*, 2023.

[364] Y. Lu, P. Lu, Z. Chen, W. Zhu, X. E. Wang, and W. Y. Wang, "Multimodal procedural planning via dual text-image prompting," *CoRR*, vol. abs/2305.01795, 2023.

[365] S. Hao, Y. Gu, H. Ma, J. J. Hong, Z. Wang, D. Z. Wang, and Z. Hu, "Reasoning with language model is planning with world model," *CoRR*, vol. abs/2305.14992, 2023.

[366] Z. Chen, K. Zhou, B. Zhang, Z. Gong, W. X. Zhao, and J. Wen, "Chatcot: Tool-augmented chain-of-thought reasoning on chat-based large language models," *CoRR*, vol. abs/2305.14323, 2023.

[367] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," *CoRR*, vol. abs/2210.03629, 2022.

[368] N. Shinn, F. Cassano, B. Labash, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," 2023.

[369] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," *CoRR*, vol. abs/2305.10601, 2023.

[370] S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer, "Rethinking the role of demonstrations: What makes in-context learning work?" in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*. Association for Computational Linguistics, 2022, pp. 11 048–11 064.

[371] Z. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh, "Calibrate before use: Improving few-shot performance of language models," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, M. Meila and T. Zhang, Eds., 2021, pp. 12 697–12 706.

[372] Y. Lee, C. Lim, and H. Choi, "Does GPT-3 generate empathetic dialogues? A novel in-context example selection method and automatic evaluation metric for empathetic dialogue generation," in *Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022*, N. Calzolari, C. Huang, H. Kim, J. Pustejovsky, L. Wanner, K. Choi, P. Ryu, H. Chen, L. Donatelli, H. Ji, S. Kurohashi, P. Paggio, N. Xue, S. Kim, Y. Hahm, Z. He, T. K. Lee, E. Santus, F. Bond, and S. Na, Eds. International Committee on Computational Linguistics, 2022, pp. 669–683.

[373] I. Levy, B. Bogin, and J. Berant, "Diverse demonstrations improve in-context compositional generalization," *CoRR*, vol. abs/2212.06800, 2022.

[374] H. Su, J. Kasai, C. H. Wu, W. Shi, T. Wang, J. Xin, R. Zhang, M. Ostendorf, L. Zettlemoyer, N. A. Smith, and T. Yu, "Selective annotation makes language models better few-shot learners," *CoRR*, 2022.

[375] X. Ye, S. Iyer, A. Celikyilmaz, V. Stoyanov, G. Durrett, and R. Pasunuru, "Complementary explanations for effective in-context learning," *CoRR*, 2022.

[376] X. Li and X. Qiu, "Finding supporting examples for in-context learning," *CoRR*, 2023.

[377] Y. Zhang, S. Feng, and C. Tan, "Active example selection for in-context learning," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, 2022, pp. 9134–9148.

[378] F. Gilardi, M. Alizadeh, and M. Kubli, "Chatgpt outperforms crowd-workers for text-annotation tasks," 2023.

[379] H. J. Kim, H. Cho, J. Kim, T. Kim, K. M. Yoo, and S. Lee, "Self-generated in-context learning: Leveraging auto-regressive language models as a demonstration generator," *CoRR*, vol. abs/2206.08082, 2022.

[380] S. M. Xie, A. Raghunathan, P. Liang, and T. Ma, "An explanation of in-context learning as implicit bayesian inference," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

[381] Z. Wu, Y. Wang, J. Ye, and L. Kong, "Self-adaptive in-context learning," *CoRR*, vol. abs/2212.10375, 2022.

[382] Y. Gu, L. Dong, F. Wei, and M. Huang, "Pre-training to learn in context," *CoRR*, vol. abs/2305.09137, 2023.

[383] S. Min, M. Lewis, L. Zettlemoyer, and H. Hajishirzi, "Metaicl: Learning to learn in context," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, M. Carpuat, M. de Marneffe, and I. V. M. Ruíz, Eds., 2022, pp. 2791–2809.

[384] M. Hahn and N. Goyal, "A theory of emergent in-context learning as implicit structure induction," *CoRR*, vol. abs/2303.07971, 2023.

[385] J. Pan, T. Gao, H. Chen, and D. Chen, "What in-context learning "learns" in-context: Disentangling task recognition and task learning," *CoRR*, vol. abs/2305.09731, 2023.

[386] N. Wies, Y. Levine, and A. Shashua, "The learnability of in-context learning," *CoRR*, vol. abs/2303.07895, 2023.

[387] A. Webson and E. Pavlick, "Do prompt-based models really understand the meaning of their prompts?" in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, 2022, pp. 2300–2344.

[388] J. von Oswald, E. Niklasson, E. Randazzo, J. Sacramento, A. Mordvintsev, A. Zhmoginov, and M. Vladymyrov, "Transformers learn in-context by gradient descent," *CoRR*, vol. abs/2212.07677, 2022.

[389] C. Olsson, N. Elhage, N. Nanda, N. Joseph, N. DasSarma, T. Henighan, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, D. Drain, D. Ganguli, Z. Hatfield-Dodds, D. Hernandez, S. Johnston, A. Jones, J. Kernion, L. Lovitt, K. Ndousse, D. Amodei, T. Brown, J. Clark, J. Kaplan, S. McCandlish, and C. Olah, "In-context learning and induction heads," *CoRR*, vol. abs/2209.11895, 2022.

[390] E. Akyürek, D. Schuurmans, J. Andreas, T. Ma, and D. Zhou, "What learning algorithm is in-context learning? investigations with linear models," *CoRR*, vol. abs/2211.15661, 2022.

[391] J. Wei, J. Wei, Y. Tay, D. Tran, A. Webson, Y. Lu, X. Chen, H. Liu, D. Huang, D. Zhou *et al.*, "Larger language models do in-context learning differently," *arXiv preprint arXiv:2303.03846*, 2023.

[392] J. Coda-Forno, M. Binz, Z. Akata, M. M. Botvinick, J. X. Wang, and E. Schulz, "Meta-in-context learning in large language models," *CoRR*, vol. abs/2305.12907, 2023.

[393] J. W. Wei, L. Hou, A. K. Lampinen, X. Chen, D. Huang, Y. Tay, X. Chen, Y. Lu, D. Zhou, T. Ma, and Q. V. Le, "Symbol tuning improves in-context learning in language models," *CoRR*, vol. abs/2305.08298, 2023.

[394] S. Miao, C. Liang, and K. Su, "A diverse corpus for evaluating and developing english math word problem solvers," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, Eds. Association for Computational Linguistics, 2020, pp. 975–984.

[395] A. Talmor, J. Herzig, N. Lourie, and J. Berant, "Commonsenseqa: A question answering challenge targeting commonsense knowledge," in *Proceedings of the 2019 Conference of the North American Chapter of*

the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 4149–4158.

[396] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," *CoRR*, vol. abs/2205.11916, 2022.

[397] E. Zelikman, J. Mu, N. D. Goodman, and Y. T. Wu, "Star: Self-taught reasoner bootstrapping reasoning with reasoning," 2022.

[398] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, B. Newman, B. Yuan, B. Yan, C. Zhang, C. Cosgrove, C. D. Manning, C. Ré, D. Acosta-Navas, D. A. Hudson, E. Zelikman, E. Durmus, F. Ladhak, F. Rong, H. Ren, H. Yao, J. Wang, K. Santhanam, L. J. Orr, L. Zheng, M. Yüksekgönül, M. Suzgun, N. Kim, N. Guha, N. S. Chatterji, O. Khattab, P. Henderson, Q. Huang, R. Chi, S. M. Xie, S. Santurkar, S. Ganguli, T. Hashimoto, T. Icard, T. Zhang, V. Chaudhary, W. Wang, X. Li, Y. Mai, Y. Zhang, and Y. Koreeda, "Holistic evaluation of language models," *CoRR*, vol. abs/2211.09110, 2022.

[399] A. Madaan and A. Yazdanbakhsh, "Text and patterns: For effective chain of thought, it takes two to tango," *CoRR*, vol. abs/2209.07686, 2022.

[400] Z. Zhang, A. Zhang, M. Li, H. Zhao, G. Karypis, and A. Smola, "Multimodal chain-of-thought reasoning in language models," *CoRR*, vol. abs/2302.00923, 2023.

[401] F. Shi, M. Suzgun, M. Freitag, X. Wang, S. Srivats, S. Vosoughi, H. W. Chung, Y. Tay, S. Ruder, D. Zhou, D. Das, and J. Wei, "Language models are multilingual chain-of-thought reasoners," *CoRR*, vol. abs/2210.03057, 2022.

[402] J. Qian, H. Wang, Z. Li, S. Li, and X. Yan, "Limitations of language models in arithmetic and symbolic induction," *CoRR*, vol. abs/2208.05051, 2022.

[403] N. Bian, X. Han, L. Sun, H. Lin, Y. Lu, and B. He, "ChatGPT is a Knowledgeable but Inexperienced Solver: An Investigation of Commonsense Problem in Large Language Models," *CoRR*, 2023.

[404] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," *CoRR*, vol. abs/2305.10601, 2023.

[405] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, "Voyager: An open-ended embodied agent with large language models," *arXiv preprint arXiv:2305.16291*, 2023.

[406] X. Jiang, Y. Dong, L. Wang, Q. Shang, and G. Li, "Self-planning code generation with large language model," *CoRR*, vol. abs/2303.06689, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2303.06689

[407] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," *CoRR*, vol. abs/2209.11302, 2022.

[408] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone, "LLM+P: empowering large language models with optimal planning proficiency," *CoRR*, vol. abs/2304.11477, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2304.11477

[409] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, 2022, pp. 10 674–10 685.

[410] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," *CoRR*, vol. abs/2304.03442, 2023.

[411] Z. Wang, S. Cai, A. Liu, X. Ma, and Y. Liang, "Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents," *CoRR*, vol. abs/2302.01560, 2023.

[412] J. Wang, X. Yi, R. Guo, H. Jin, P. Xu, S. Li, X. Wang, X. Guo, C. Li, X. Xu *et al.*, "Milvus: A purpose-built vector data management system," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 2614–2627.

[413] W. Zhong, L. Guo, Q. Gao, H. Ye, and Y. Wang, "Memorybank: Enhancing large language models with long-term memory," *CoRR*, vol. abs/2305.10250, 2023.

[414] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of english: The penn treebank," *Comput. Linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[415] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," in *ICLR (Poster)*. OpenReview.net, 2017.

[416] O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, L. Specia, and A. Tamchyna, "Findings of the 2014 workshop on statistical machine translation," in *WMT@ACL*. The Association for Computer Linguistics, 2014, pp. 12–58.

[417] O. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. Jimeno-Yepes, P. Koehn, V. Logacheva, C. Monz, M. Negri, A. Névéol, M. L. Neves, M. Popel, M. Post, R. Rubino, C. Scarton, L. Specia, M. Turchi, K. Verspoor, and M. Zampieri, "Findings of the 2016 conference on machine translation," in *WMT*. The Association for Computer Linguistics, 2016, pp. 131–198.

[418] L. Barrault, O. Bojar, M. R. Costa-jussà, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, P. Koehn, S. Malmasi, C. Monz, M. Müller, S. Pal, M. Post, and M. Zampieri, "Findings of the 2019 conference on machine translation (WMT19)," in *Proceedings of the Fourth Conference on Machine Translation, WMT 2019, Florence, Italy, August 1-2, 2019 - Volume 2: Shared Task Papers, Day 1*, O. Bojar, R. Chatterjee, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, A. Jimeno-Yepes, P. Koehn, A. Martins, C. Monz, M. Negri, A. Névéol, M. L. Neves, M. Post, M. Turchi, and K. Verspoor, Eds. Association for Computational Linguistics, 2019, pp. 1–61.

[419] L. Barrault, M. Biesialska, O. Bojar, M. R. Costa-

jussà, C. Federmann, Y. Graham, R. Grundkiewicz, B. Haddow, M. Huck, E. Joanis, T. Kocmi, P. Koehn, C. Lo, N. Ljubesic, C. Monz, M. Morishita, M. Nagata, T. Nakazawa, S. Pal, M. Post, and M. Zampieri, "Findings of the 2020 conference on machine translation (WMT20)," in *Proceedings of the Fifth Conference on Machine Translation, WMT@EMNLP 2020, Online, November 19-20, 2020*, L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussà, C. Federmann, M. Fishel, A. Fraser, Y. Graham, P. Guzman, B. Haddow, M. Huck, A. Jimeno-Yepes, P. Koehn, A. Martins, M. Morishita, C. Monz, M. Nagata, T. Nakazawa, and M. Negri, Eds. Association for Computational Linguistics, 2020, pp. 1–55.

[420] F. Akhbardeh, A. Arkhangorodsky, M. Biesialska, O. Bojar, R. Chatterjee, V. Chaudhary, M. R. Costa-jussà, C. España-Bonet, A. Fan, C. Federmann, M. Freitag, Y. Graham, R. Grundkiewicz, B. Haddow, L. Harter, K. Heafield, C. Homan, M. Huck, K. Amponsah-Kaakyire, J. Kasai, D. Khashabi, K. Knight, T. Kocmi, P. Koehn, N. Lourie, C. Monz, M. Morishita, M. Nagata, A. Nagesh, T. Nakazawa, M. Negri, S. Pal, A. A. Tapo, M. Turchi, V. Vydrin, and M. Zampieri, "Findings of the 2021 conference on machine translation (WMT21)," in *Proceedings of the Sixth Conference on Machine Translation, WMT@EMNLP 2021, Online Event, November 10-11, 2021*, L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussà, C. Federmann, M. Fishel, A. Fraser, M. Freitag, Y. Graham, R. Grundkiewicz, P. Guzman, B. Haddow, M. Huck, A. Jimeno-Yepes, P. Koehn, T. Kocmi, A. Martins, M. Morishita, and C. Monz, Eds. Association for Computational Linguistics, 2021, pp. 1–88.

[421] T. Kocmi, R. Bawden, O. Bojar, A. Dvorkovich, C. Federmann, M. Fishel, T. Gowda, Y. Graham, R. Grundkiewicz, B. Haddow, R. Knowles, P. Koehn, C. Monz, M. Morishita, M. Nagata, T. Nakazawa, M. Novák, M. Popel, and M. Popovic, "Findings of the 2022 conference on machine translation (WMT22)," in *Proceedings of the Seventh Conference on Machine Translation, WMT 2022, Abu Dhabi, United Arab Emirates (Hybrid), December 7-8, 2022*, P. Koehn, L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussà, C. Federmann, M. Fishel, A. Fraser, M. Freitag, Y. Graham, R. Grundkiewicz, P. Guzman, B. Haddow, M. Huck, A. Jimeno-Yepes, T. Kocmi, A. Martins, M. Morishita, C. Monz, M. Nagata, T. Nakazawa, M. Negri, A. Névéol, M. Neves, M. Popel, M. Turchi, and M. Zampieri, Eds. Association for Computational Linguistics, 2022, pp. 1–45.

[422] N. Goyal, C. Gao, V. Chaudhary, P. Chen, G. Wenzek, D. Ju, S. Krishnan, M. Ranzato, F. Guzmán, and A. Fan, "The flores-101 evaluation benchmark for low-resource and multilingual machine translation," *Trans. Assoc. Comput. Linguistics*, vol. 10, pp. 522–538, 2022.

[423] R. Bawden, E. Bilinski, T. Lavergne, and S. Rosset, "Diabla: a corpus of bilingual spontaneous written dialogues for machine translation," *Lang. Resour. Evaluation*, vol. 55, no. 3, pp. 635–660, 2021.

[424] R. Nallapati, B. Zhou, C. N. dos Santos, Ç. Gülçehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence rnns and beyond," in *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, Y. Goldberg and S. Riezler, Eds. ACL, 2016, pp. 280–290.

[425] S. Narayan, S. B. Cohen, and M. Lapata, "Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization," in *EMNLP*. Association for Computational Linguistics, 2018, pp. 1797–1807.

[426] F. Ladhak, E. Durmus, C. Cardie, and K. Mckeown, "Wikilingua: A new benchmark dataset for cross-lingual abstractive summarization," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp. 4034–4048.

[427] S. Moon, P. Shah, A. Kumar, and R. Subba, "Opendialkg: Explainable conversational reasoning with attention-based walks over knowledge graphs," in *ACL (1)*. Association for Computational Linguistics, 2019, pp. 845–854.

[428] D. Hendrycks, S. Basart, S. Kadavath, M. Mazeika, A. Arora, E. Guo, C. Burns, S. Puranik, H. He, D. Song, and J. Steinhardt, "Measuring coding challenge competence with APPS," in *NeurIPS Datasets and Benchmarks*, 2021.

[429] Y. Lai, C. Li, Y. Wang, T. Zhang, R. Zhong, L. Zettlemoyer, S. W. Yih, D. Fried, S. I. Wang, and T. Yu, "DS-1000: A natural and reliable benchmark for data science code generation," *CoRR*, vol. abs/2211.11501, 2022.

[430] Z. Wang, S. Zhou, D. Fried, and G. Neubig, "Execution-based evaluation for open-domain code generation," *CoRR*, vol. abs/2212.10481, 2022.

[431] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. P. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov, "Natural questions: a benchmark for question answering research," *Trans. Assoc. Comput. Linguistics*, pp. 452–466, 2019.

[432] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, "Think you have solved question answering? try arc, the AI2 reasoning challenge," *CoRR*, vol. abs/1803.05457, 2018.

[433] S. Lin, J. Hilton, and O. Evans, "Truthfulqa: Measuring how models mimic human falsehoods," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, 2022, pp. 3214–3252.

[434] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on freebase from question-answer pairs," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, 2013, pp. 1533–1544.

[435] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer, "Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30*

- *August 4, Volume 1: Long Papers*, 2017, pp. 1601–1611.

[436] Y. Bisk, R. Zellers, R. L. Bras, J. Gao, and Y. Choi, "PIQA: reasoning about physical commonsense in natural language," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 2020, pp. 7432–7439.

[437] M. Dubey, D. Banerjee, A. Abdelkawi, and J. Lehmann, "Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia," in *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*, 2019, pp. 69–78.

[438] Y. Gu, S. Kase, M. Vanni, B. M. Sadler, P. Liang, X. Yan, and Y. Su, "Beyond I.I.D.: three levels of generalization for question answering on knowledge bases," in *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, 2021, pp. 3477–3488.

[439] S. Cao, J. Shi, L. Pan, L. Nie, Y. Xiang, L. Hou, J. Li, B. He, and H. Zhang, "KQA pro: A dataset with explicit compositional programs for complex question answering over knowledge base," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, 2022, pp. 6101–6119.

[440] X. Hu, X. Wu, Y. Shu, and Y. Qu, "Logical form generation via multi-task learning for complex question answering over knowledge bases," in *Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022*, 2022, pp. 1687–1696.

[441] S. Longpre, Y. Lu, and J. Daiber, "MKQA: A linguistically diverse benchmark for multilingual open domain question answering," *Trans. Assoc. Comput. Linguistics*, vol. 9, pp. 1389–1406, 2021.

[442] T. Saikh, T. Ghosal, A. Mittal, A. Ekbal, and P. Bhattacharyya, "Scienceqa: a novel resource for question answering on scholarly articles," *Int. J. Digit. Libr.*, vol. 23, no. 3, pp. 289–301, 2022.

[443] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal, "Can a suit of armor conduct electricity? A new dataset for open book question answering," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, 2018, pp. 2381–2391.

[444] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng, "MS MARCO: A human generated machine reading comprehension dataset," in *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, 2016.

[445] T. Khot, P. Clark, M. Guerquin, P. Jansen, and A. Sabharwal, "QASC: A dataset for question answering via sentence composition," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 2020, pp. 8082–8090.

[446] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100, 000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, 2016, pp. 2383–2392.

[447] A. H. Miller, A. Fisch, J. Dodge, A. Karimi, A. Bordes, and J. Weston, "Key-value memory networks for directly reading documents," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, 2016, pp. 1400–1409.

[448] B. Goodrich, V. Rao, P. J. Liu, and M. Saleh, "Assessing the factual accuracy of generated text," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, 2019, pp. 166–175.

[449] K. Toutanova and D. Chen, "Observed versus latent features for knowledge base and text inference," in *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, CVSC 2015, Beijing, China, July 26-31, 2015*, 2015, pp. 57–66.

[450] K. D. Bollacker, C. Evans, P. K. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, 2008, pp. 1247–1250.

[451] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 2018, pp. 1811–1818.

[452] G. A. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, pp. 39–41, 1995.

[453] F. Petroni, T. Rocktäschel, S. Riedel, P. S. H. Lewis, A. Bakhtin, Y. Wu, and A. H. Miller, "Language models as knowledge bases?" in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, 2019, pp. 2463–2473.

[454] F. Mahdisoltani, J. Biega, and F. M. Suchanek, "YAGO3: A knowledge base from multilingual wikipedias," in *Seventh Biennial Conference on Innovative Data Systems Research, CIDR 2015, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*, 2015.

[455] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, 2007, pp.

697–706.

[456] M. Geva, D. Khashabi, E. Segal, T. Khot, D. Roth, and J. Berant, "Did aristotle use a laptop? A question answering benchmark with implicit reasoning strategies," *Trans. Assoc. Comput. Linguistics*, vol. 9, pp. 346–361, 2021.

[457] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning, "Hotpotqa: A dataset for diverse, explainable multi-hop question answering," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*. Association for Computational Linguistics, 2018, pp. 2369–2380.

[458] C. Clark, K. Lee, M. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova, "Boolq: Exploring the surprising difficulty of natural yes/no questions," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 2924–2936.

[459] M. Sap, H. Rashkin, D. Chen, R. L. Bras, and Y. Choi, "Socialiqa: Commonsense reasoning about social interactions," *CoRR*, vol. abs/1904.09728, 2019.

[460] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, "Hellaswag: Can a machine really finish your sentence?" in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, A. Korhonen, D. R. Traum, and L. Màrquez, Eds. Association for Computational Linguistics, 2019, pp. 4791–4800.

[461] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi, "Winogrande: An adversarial winograd schema challenge at scale," in *AAAI*. AAAI Press, 2020, pp. 8732–8740.

[462] M. Roemmele, C. A. Bejan, and A. S. Gordon, "Choice of plausible alternatives: An evaluation of commonsense causal reasoning," in *Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-06, Stanford, California, USA, March 21-23, 2011*. AAAI, 2011.

[463] K. Sakaguchi, C. Bhagavatula, R. L. Bras, N. Tandon, P. Clark, and Y. Choi, "proscript: Partially ordered scripts generation," in *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, M. Moens, X. Huang, L. Specia, and S. W. Yih, Eds. Association for Computational Linguistics, 2021, pp. 2138–2149.

[464] B. Dalvi, L. Huang, N. Tandon, W. Yih, and P. Clark, "Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, M. A. Walker, H. Ji, and A. Stent, Eds. Association for Computational Linguistics, 2018, pp. 1595–1604.

[465] S. Saha, P. Yadav, L. Bauer, and M. Bansal, "Explagraphs: An explanation graph generation task for structured commonsense reasoning," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, M. Moens, X. Huang, L. Specia, and S. W. Yih, Eds. Association for Computational Linguistics, 2021, pp. 7716–7740.

[466] O. Tafjord, B. Dalvi, and P. Clark, "Proofwriter: Generating implications, proofs, and abductive statements over natural language," in *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, ser. Findings of ACL, C. Zong, F. Xia, W. Li, and R. Navigli, Eds., vol. ACL/IJCNLP 2021. Association for Computational Linguistics, 2021, pp. 3621–3634.

[467] B. Dalvi, P. Jansen, O. Tafjord, Z. Xie, H. Smith, L. Pipatanangkura, and P. Clark, "Explaining answers with entailment trees," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, M. Moens, X. Huang, L. Specia, and S. W. Yih, Eds. Association for Computational Linguistics, 2021, pp. 7358–7370.

[468] A. Saparov and H. He, "Language models are greedy reasoners: A systematic formal analysis of chain-of-thought," *CoRR*, vol. abs/2210.01240, 2022.

[469] C. Anil, Y. Wu, A. Andreassen, A. Lewkowycz, V. Misra, V. V. Ramasesh, A. Slone, G. Gur-Ari, E. Dyer, and B. Neyshabur, "Exploring length generalization in large language models," *CoRR*, vol. abs/2207.04901, 2022.

[470] K. Cobbe, V. Kosaraju, M. Bavarian, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, "Training verifiers to solve math word problems," *CoRR*, vol. abs/2110.14168, 2021.

[471] A. Patel, S. Bhattamishra, and N. Goyal, "Are NLP models really able to solve simple math word problems?" in *NAACL-HLT*. Association for Computational Linguistics, 2021, pp. 2080–2094.

[472] S. Roy and D. Roth, "Solving general arithmetic word problems," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, L. Màrquez, C. Callison-Burch, J. Su, D. Pighin, and Y. Marton, Eds. The Association for Computational Linguistics, 2015, pp. 1743–1752.

[473] A. Amini, S. Gabriel, S. Lin, R. Koncel-Kedziorski, Y. Choi, and H. Hajishirzi, "Mathqa: Towards interpretable math word problem solving with operation-based formalisms," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 2357–2367.

[474] W. Ling, D. Yogatama, C. Dyer, and P. Blunsom,

"Program induction by rationale generation: Learning to solve and explain algebraic word problems," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, R. Barzilay and M. Kan, Eds. Association for Computational Linguistics, 2017, pp. 158–167.

[475] R. Koncel-Kedziorski, S. Roy, A. Amini, N. Kushman, and H. Hajishirzi, "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*, 2016, pp. 1152–1157.

[476] D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner, "DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, 2019, pp. 2368–2378.

[477] S. Welleck, J. Liu, R. L. Bras, H. Hajishirzi, Y. Choi, and K. Cho, "Naturalproofs: Mathematical theorem proving in natural language," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, J. Vanschoren and S. Yeung, Eds., 2021.

[478] A. Q. Jiang, W. Li, J. M. Han, and Y. Wu, "Lisa: Language models of isabelle proofs," in *6th Conference on Artificial Intelligence and Theorem Proving*, 2021, pp. 378–392.

[479] K. Zheng, J. M. Han, and S. Polu, "minif2f: a cross-system benchmark for formal olympiad-level mathematics," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[480] Z. Azerbayev, B. Piotrowski, H. Schoelkopf, E. W. Ayers, D. Radev, and J. Avigad, "Proofnet: Autoformalizing and formally proving undergraduate-level mathematics," *CoRR*, vol. abs/2302.12433, 2023.

[481] J. Li, X. Cheng, W. X. Zhao, J. Nie, and J. Wen, "Halueval: A large-scale hallucination evaluation benchmark for large language models," *CoRR*, vol. abs/2305.11747, 2023.

[482] N. Nangia, C. Vania, R. Bhalerao, and S. R. Bowman, "Crows-pairs: A challenge dataset for measuring social biases in masked language models," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, 2020, pp. 1953–1967.

[483] R. Rudinger, J. Naradowsky, B. Leonard, and B. V. Durme, "Gender bias in coreference resolution," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, 2018, pp. 8–14.

[484] S. Gehman, S. Gururangan, M. Sap, Y. Choi, and N. A. Smith, "Realtoxicityprompts: Evaluating neural toxic degeneration in language models," in *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, ser. Findings of ACL, T. Cohn, Y. He, and Y. Liu, Eds., vol. EMNLP 2020. Association for Computational Linguistics, 2020, pp. 3356–3369.

[485] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, "Virtualhome: Simulating household activities via programs," in *CVPR*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 8494–8502.

[486] S. Srivastava, C. Li, M. Lingelbach, R. Martín-Martín, F. Xia, K. E. Vainio, Z. Lian, C. Gokmen, S. Buch, C. K. Liu, S. Savarese, H. Gweon, J. Wu, and L. Fei-Fei, "BEHAVIOR: benchmark for everyday household activities in virtual, interactive, and ecological environments," in *CoRL*, ser. Proceedings of Machine Learning Research, vol. 164. PMLR, 2021, pp. 477–490.

[487] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox, "ALFRED: A benchmark for interpreting grounded instructions for everyday tasks," in *CVPR*. Computer Vision Foundation / IEEE, 2020, pp. 10 737–10 746.

[488] M. Shridhar, X. Yuan, M. Côté, Y. Bisk, A. Trischler, and M. J. Hausknecht, "Alfworld: Aligning text and embodied environments for interactive learning," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[489] S. Yao, H. Chen, J. Yang, and K. Narasimhan, "Webshop: Towards scalable real-world web interaction with grounded language agents," in *NeurIPS*, 2022.

[490] X. Deng, Y. Gu, B. Zheng, S. Chen, S. Stevens, B. Wang, H. Sun, and Y. Su, "Mind2web: Towards a generalist agent for the web," *CoRR*, vol. abs/2306.06070, 2023.

[491] W. H. Guss, B. Houghton, N. Topin, P. Wang, C. Codel, M. Veloso, and R. Salakhutdinov, "Minerl: A large-scale dataset of minecraft demonstrations," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, S. Kraus, Ed. ijcai.org, 2019, pp. 2442–2448.

[492] L. Fan, G. Wang, Y. Jiang, A. Mandlekar, Y. Yang, H. Zhu, A. Tang, D. Huang, Y. Zhu, and A. Anandkumar, "Minedojo: Building open-ended embodied agents with internet-scale knowledge," in *NeurIPS*, 2022.

[493] P. Lu, L. Qiu, K. Chang, Y. N. Wu, S. Zhu, T. Rajpurohit, P. Clark, and A. Kalyan, "Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning," *CoRR*, vol. abs/2209.14610, 2022.

[494] B. Zhang, K. Zhou, X. Wei, W. X. Zhao, J. Sha, S. Wang, and J. rong Wen, "Evaluating and improving tool-augmented computation-intensive math reasoning," *CoRR*, vol. abs/2306.02408, 2023.

[495] R. Yang, L. Song, Y. Li, S. Zhao, Y. Ge, X. Li, and Y. Shan, "Gpt4tools: Teaching large language model to use tools via self-instruction," *CoRR*, vol. abs/2305.18752, 2023.

[496] S. G. Patil, T. Zhang, X. Wang, and J. E. Gonzalez, "Go-

rilla: Large language model connected with massive apis," *CoRR*, vol. abs/2305.15334, 2023.

[497] W. Yih, M. Richardson, C. Meek, M. Chang, and J. Suh, "The value of semantic parse labeling for knowledge base question answering," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*. The Association for Computer Linguistics, 2016.

[498] H. Puerto, G. G. Sahin, and I. Gurevych, "Metaqa: Combining expert agents for multi-skill question answering," in *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*, A. Vlachos and I. Augenstein, Eds. Association for Computational Linguistics, 2023, pp. 3548–3562.

[499] P. Pasupat and P. Liang, "Compositional semantic parsing on semi-structured tables," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. The Association for Computer Linguistics, 2015, pp. 1470–1480.

[500] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," *CoRR*, vol. abs/1709.00103, 2017.

[501] W. Chen, H. Wang, J. Chen, Y. Zhang, H. Wang, S. Li, X. Zhou, and W. Y. Wang, "Tabfact: A large-scale dataset for table-based fact verification," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[502] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. R. Radev, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, Eds. Association for Computational Linguistics, 2018, pp. 3911–3921.

[503] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015.

[504] K. Papineni, S. Roukos, T. Ward, and W. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*. ACL, 2002, pp. 311–318.

[505] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Association for Computational Linguistics, Jul. 2004, pp. 74–81.

[506] W. Jiao, W. Wang, J.-t. Huang, X. Wang, and Z. Tu, "Is chatgpt a good translator? a preliminary study," *arXiv preprint arXiv:2301.08745*, 2023.

[507] T. Zhang, F. Ladhak, E. Durmus, P. Liang, K. R. McKeown, and T. B. Hashimoto, "Benchmarking large language models for news summarization," *CoRR*, vol. abs/2301.13848, 2023.

[508] T. Goyal, J. J. Li, and G. Durrett, "News summarization and evaluation in the era of GPT-3," *CoRR*, vol. abs/2209.12356, 2022.

[509] S. Gehrmann, E. Clark, and T. Sellam, "Repairing the cracked foundation: A survey of obstacles in evaluation practices for generated text," *CoRR*, vol. abs/2202.06935, 2022.

[510] J. Wang, Y. Liang, F. Meng, H. Shi, Z. Li, J. Xu, J. Qu, and J. Zhou, "Is chatgpt a good NLG evaluator? A preliminary study," *CoRR*, vol. abs/2303.04048, 2023.

[511] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, and C. Zhu, "G-eval: NLG evaluation using GPT-4 with better human alignment," *CoRR*, vol. abs/2303.16634, 2023.

[512] J. Jiang, K. Zhou, Z. Dong, K. Ye, W. X. Zhao, and J. Wen, "Structgpt: A general framework for large language model to reason over structured data," *CoRR*, vol. abs/2305.09645, 2023.

[513] K. Yang, Y. Tian, N. Peng, and D. Klein, "Re3: Generating longer stories with recursive reprompting and revision," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds. Association for Computational Linguistics, 2022, pp. 4393–4479.

[514] W. Zhou, Y. E. Jiang, P. Cui, T. Wang, Z. Xiao, Y. Hou, R. Cotterell, and M. Sachan, "Recurrentgpt: Interactive generation of (arbitrarily) long text," *CoRR*, vol. abs/2305.13304, 2023.

[515] S. Gulwani, O. Polozov, and R. Singh, "Program synthesis," *Found. Trends Program. Lang.*, vol. 4, no. 1-2, pp. 1–119, 2017.

[516] S. Zhang, Z. Chen, Y. Shen, M. Ding, J. B. Tenenbaum, and C. Gan, "Planning with large language models for code generation," 2023.

[517] M. Welsh, "The end of programming," *Commun. ACM*, vol. 66, no. 1, pp. 34–35, 2023.

[518] Y. Bang, S. Cahyawijaya, N. Lee, W. Dai, D. Su, B. Wilie, H. Lovenia, Z. Ji, T. Yu, W. Chung, Q. V. Do, Y. Xu, and P. Fung, "A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity," *CoRR*, vol. abs/2302.04023, 2023.

[519] Y. Liu, A. R. Fabbri, P. Liu, Y. Zhao, L. Nan, R. Han, S. Han, S. R. Joty, C. Wu, C. Xiong, and D. Radev, "Revisiting the gold standard: Grounding summarization evaluation with robust human evaluation," *CoRR*, vol. abs/2212.07981, 2022.

[520] A. R. Fabbri, W. Kryscinski, B. McCann, C. Xiong, R. Socher, and D. R. Radev, "Summeval: Re-evaluating summarization evaluation," *Trans. Assoc. Comput. Linguistics*, vol. 9, pp. 391–409, 2021.

[521] T. Tang, H. Lu, Y. E. Jiang, H. Huang, D. Zhang, W. X. Zhao, and F. Wei, "Not all metrics are guilty: Improving NLG evaluation with LLM paraphrasing," *CoRR*, vol. abs/2305.15067, 2023.

[522] X. Wang, X. Tang, W. X. Zhao, J. Wang, and J. Wen, "Rethinking the evaluation for conversational rec-

ommendation in the era of large language models," *CoRR*, vol. abs/2305.13112, 2023.

[523] M. Gao, J. Ruan, R. Sun, X. Yin, S. Yang, and X. Wan, "Human-like summarization evaluation with chatgpt," *CoRR*, vol. abs/2304.02554, 2023.

[524] Y. Ji, Y. Gong, Y. Peng, C. Ni, P. Sun, D. Pan, B. Ma, and X. Li, "Exploring chatgpt's ability to rank content: A preliminary study on consistency with human preferences," *CoRR*, vol. abs/2303.07610, 2023.

[525] Y. Bai, J. Ying, Y. Cao, X. Lv, Y. He, X. Wang, J. Yu, K. Zeng, Y. Xiao, H. Lyu, J. Zhang, J. Li, and L. Hou, "Benchmarking foundation models with language-model-as-an-examiner," *CoRR*, vol. abs/2306.04181, 2023.

[526] Y. Liu, S. Feng, D. Wang, Y. Zhang, and H. Schütze, "Evaluate what you can't evaluate: Unassessable generated responses quality," *CoRR*, vol. abs/2305.14658, 2023.

[527] P. Wang, L. Li, L. Chen, D. Zhu, B. Lin, Y. Cao, Q. Liu, T. Liu, and Z. Sui, "Large language models are not fair evaluators," *CoRR*, vol. abs/2305.17926, 2023.

[528] J. Ye, X. Chen, N. Xu, C. Zu, Z. Shao, S. Liu, Y. Cui, Z. Zhou, C. Gong, Y. Shen, J. Zhou, S. Chen, T. Gui, Q. Zhang, and X. Huang, "A comprehensive capability analysis of gpt-3 and gpt-3.5 series models," *arXiv preprint arXiv:2303.10420*, 2023.

[529] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*, 1989, pp. 109–165.

[530] R. Kemker, M. McClure, A. Abitino, T. L. Hayes, and C. Kanan, "Measuring catastrophic forgetting in neural networks," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 2018, pp. 3390–3398.

[531] T. Xie, C. H. Wu, P. Shi, R. Zhong, T. Scholak, M. Yasunaga, C. Wu, M. Zhong, P. Yin, S. I. Wang, V. Zhong, B. Wang, C. Li, C. Boyle, A. Ni, Z. Yao, D. Radev, C. Xiong, L. Kong, R. Zhang, N. A. Smith, L. Zettlemoyer, and T. Yu, "Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models," in *EMNLP*. Association for Computational Linguistics, 2022, pp. 602–631.

[532] A. Roberts, C. Raffel, and N. Shazeer, "How much knowledge can you pack into the parameters of a language model?" in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, 2020, pp. 5418–5426.

[533] G. Izacard, P. S. H. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin, S. Riedel, and E. Grave, "Few-shot learning with retrieval augmented language models," *CoRR*, vol. abs/2208.03299, 2022.

[534] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, "Retrieval augmented language model pre-training," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, 2020, pp. 3929–3938.

[535] P. S. H. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[536] Y. Lan, G. He, J. Jiang, J. Jiang, W. X. Zhao, and J. Wen, "Complex knowledge base question answering: A survey," *CoRR*, vol. abs/2108.06688, 2021.

[537] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. van den Driessche, J. Lespiau, B. Damoc, A. Clark, D. de Las Casas, A. Guy, J. Menick, R. Ring, T. Hennigan, S. Huang, L. Maggiore, C. Jones, A. Cassirer, A. Brock, M. Paganini, G. Irving, O. Vinyals, S. Osindero, K. Simonyan, J. W. Rae, E. Elsen, and L. Sifre, "Improving language models by retrieving from trillions of tokens," in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 2022, pp. 2206–2240.

[538] S. Xu, L. Pang, H. Shen, X. Cheng, and T.-S. Chua, "Search-in-the-chain: Towards accurate, credible and traceable large language models for knowledge-intensive tasks," *CoRR*, vol. abs/2304.14732, 2023.

[539] B. Peng, M. Galley, P. He, H. Cheng, Y. Xie, Y. Hu, Q. Huang, L. Liden, Z. Yu, W. Chen, and J. Gao, "Check your facts and try again: Improving large language models with external knowledge and automated feedback," *CoRR*, vol. abs/2302.12813, 2023.

[540] Z. Jiang, F. F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig, "Active retrieval augmented generation," *CoRR*, vol. abs/2305.06983, 2023.

[541] Y. Li, Y. Du, K. Zhou, J. Wang, W. X. Zhao, and J. Wen, "Evaluating object hallucination in large vision-language models," *CoRR*, vol. abs/2305.10355, 2023.

[542] S. Kadavath, T. Conerly, A. Askell, T. J. Henighan, D. Drain, E. Perez, N. Schiefer, Z. Dodds, N. DasSarma, E. Tran-Johnson, S. Johnston, S. El-Showk, A. Jones, N. Elhage, T. Hume, A. Chen, Y. Bai, S. Bowman, S. Fort, D. Ganguli, D. Hernandez, J. Jacobson, J. Kernion, S. Kravec, L. Lovitt, K. Ndousse, C. Olsson, S. Ringer, D. Amodei, T. B. Brown, J. Clark, N. Joseph, B. Mann, S. McCandlish, C. Olah, and J. Kaplan, "Language models (mostly) know what they know," *CoRR*, vol. abs/2207.05221, 2022.

[543] P. Manakul, A. Liusie, and M. J. F. Gales, "Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models," *ArXiv*, vol. abs/2305.06983, 2023.

[544] S. Agarwal, I. Akkaya, V. Balcom, M. Bavarian, G. Bernadett-Shapiro, G. Brockman, M. Brundage, J. Chan, F. Chantzis, N. Deutsch, B. Eastman, A. Eleti,

N. Felix, S. P. Fishman, I. Fulford, C. Gibson, J. Gross, M. Heaton, J. Hilton, X. Hu, S. Jain, H. Jin, L. Kilpatrick, C. Kim, M. Kolhede, A. Mayne, P. McMillan, D. Medina, J. Menick, A. Mishchenko, A. Nair, R. Nayak, A. Neelakantan, R. Nuttall, J. Parish, A. T. Passos, A. Perelman, F. de Avila Belbute Peres, V. Pong, J. Schulman, E. Sigler, N. Staudacher, N. Turley, J. Tworek, R. Greene, A. Vijayvergiya, C. Voss, J. Weng, M. Wiethoff, S. Yoo, K. Yu, W. Zaremba, S. Zhao, W. Zhuk, and B. Zoph, "Chatgpt plugins," *OpenAI Blog*, March 2023.

[545] A. Lazaridou, E. Gribovskaya, W. Stokowiec, and N. Grigorev, "Internet-augmented language models through few-shot prompting for open-domain question answering," *CoRR*, vol. abs/2203.05115, 2022.

[546] H. Qian, Y. Zhu, Z. Dou, H. Gu, X. Zhang, Z. Liu, R. Lai, Z. Cao, J. Nie, and J. Wen, "Webbrain: Learning to generate factually correct articles for queries by grounding on large web corpus," *CoRR*, vol. abs/2304.04358, 2023.

[547] J. Liu, J. Jin, Z. Wang, J. Cheng, Z. Dou, and J. Wen, "RETA-LLM: A retrieval-augmented large language model toolkit," *CoRR*, vol. abs/2306.05212, 2023.

[548] D. Dai, L. Dong, Y. Hao, Z. Sui, B. Chang, and F. Wei, "Knowledge neurons in pretrained transformers," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Association for Computational Linguistics, 2022, pp. 8493–8502.

[549] K. Meng, D. Bau, A. J. Andonian, and Y. Belinkov, "Locating and editing factual associations in gpt," in *Advances in Neural Information Processing Systems*, 2022.

[550] M. Geva, R. Schuster, J. Berant, and O. Levy, "Transformer feed-forward layers are key-value memories," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, M. Moens, X. Huang, L. Specia, and S. W. Yih, Eds. Association for Computational Linguistics, 2021, pp. 5484–5495.

[551] Y. Yao, P. Wang, B. Tian, S. Cheng, Z. Li, S. Deng, H. Chen, and N. Zhang, "Editing large language models: Problems, methods, and opportunities," *CoRR*, vol. abs/2305.13172, 2023.

[552] P. Wang, N. Zhang, X. Xie, Y. Yao, B. Tian, M. Wang, Z. Xi, S. Cheng, K. Liu, G. Zheng, and H. Chen, "Easyedit: An easy-to-use knowledge editing framework for large language models," *CoRR*, vol. abs/2308.07269, 2023.

[553] Z. Shao, Y. Gong, Y. Shen, M. Huang, N. Duan, and W. Chen, "Synthetic prompting: Generating chain-of-thought demonstrations for large language models," *CoRR*, vol. abs/2302.00618, 2023.

[554] Sifatkaur, M. Singh, V. S. B, and N. Malviya, "Mind meets machine: Unravelling gpt-4's cognitive psychology," *CoRR*, vol. abs/2303.11436, 2023.

[555] M. I. Nye, A. J. Andreassen, G. Gur-Ari, H. Michalewski, J. Austin, D. Bieber, D. Dohan, A. Lewkowycz, M. Bosma, D. Luan, C. Sutton, and A. Odena, "Show your work: Scratchpads for intermediate computation with language models," *CoRR*, vol. abs/2112.00114, 2021.

[556] J. Qian, H. Wang, Z. Li, S. Li, and X. Yan, "Limitations of language models in arithmetic and symbolic induction," *CoRR*, vol. abs/2208.05051, 2022.

[557] W. X. Zhao, K. Zhou, Z. Gong, B. Zhang, Y. Zhou, J. Sha, Z. Chen, S. Wang, C. Liu, and J. Wen, "Jiuzhang: A chinese pre-trained language model for mathematical problem understanding," in *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, A. Zhang and H. Rangwala, Eds. ACM, 2022, pp. 4571–4581.

[558] Q. Wang, C. Kaliszyk, and J. Urban, "First experiments with neural translation of informal to formal mathematics," in *Intelligent Computer Mathematics - 11th International Conference, CICM 2018, Hagenberg, Austria, August 13-17, 2018, Proceedings*, ser. Lecture Notes in Computer Science, F. Rabe, W. M. Farmer, G. O. Passmore, and A. Youssef, Eds., vol. 11006. Springer, 2018, pp. 255–270.

[559] S. Polu and I. Sutskever, "Generative language modeling for automated theorem proving," *CoRR*, vol. abs/2009.03393, 2020.

[560] A. Q. Jiang, W. Li, S. Tworkowski, K. Czechowski, T. Odrzygózdz, P. Milos, Y. Wu, and M. Jamnik, "Thor: Wielding hammers to integrate language models and automated theorem provers," *CoRR*, vol. abs/2205.10893, 2022.

[561] S. Polu, J. M. Han, K. Zheng, M. Baksys, I. Babuschkin, and I. Sutskever, "Formal mathematics statement curriculum learning," *CoRR*, vol. abs/2202.01344, 2022.

[562] Y. Wu, A. Q. Jiang, W. Li, M. N. Rabe, C. Staats, M. Jamnik, and C. Szegedy, "Autoformalization with large language models," *CoRR*, vol. abs/2205.12615, 2022.

[563] A. Q. Jiang, S. Welleck, J. P. Zhou, W. Li, J. Liu, M. Jamnik, T. Lacroix, Y. Wu, and G. Lample, "Draft, sketch, and prove: Guiding formal theorem provers with informal proofs," *CoRR*, vol. abs/2210.12283, 2022.

[564] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Welleck, B. P. Majumder, S. Gupta, A. Yazdanbakhsh, and P. Clark, "Self-refine: Iterative refinement with self-feedback," *CoRR*, vol. abs/2303.17651, 2023.

[565] N. Shinn, B. Labash, and A. Gopinath, "Reflexion: an autonomous agent with dynamic memory and self-reflection," *CoRR*, vol. abs/2303.11366, 2023.

[566] Z. Gou, Z. Shao, Y. Gong, Y. Shen, Y. Yang, N. Duan, and W. Chen, "CRITIC: large language models can self-correct with tool-interactive critiquing," *CoRR*, vol. abs/2305.11738, 2023.

[567] J. Uesato, N. Kushman, R. Kumar, H. F. Song, N. Y. Siegel, L. Wang, A. Creswell, G. Irving, and I. Higgins, "Solving math word problems with process- and outcome-based feedback," *CoRR*, vol. abs/2211.14275, 2022.

[568] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever,

and K. Cobbe, "Let's verify step by step," *CoRR*, vol. abs/2305.20050, 2023.

[569] Z. Yuan, H. Yuan, C. Tan, W. Wang, and S. Huang, "How well do large language models perform in arithmetic tasks?" *CoRR*, vol. abs/2304.02015, 2023.

[570] X. Pi, Q. Liu, B. Chen, M. Ziyadi, Z. Lin, Q. Fu, Y. Gao, J. Lou, and W. Chen, "Reasoning like program executors," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, 2022, pp. 761–779.

[571] H. Zhou, A. Nova, H. Larochelle, A. C. Courville, B. Neyshabur, and H. Sedghi, "Teaching algorithmic reasoning via in-context learning," *CoRR*, vol. abs/2211.09066, 2022.

[572] A. Parisi, Y. Zhao, and N. Fiedel, "TALM: tool augmented language models," *CoRR*, vol. abs/2205.12255, 2022.

[573] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 162. PMLR, 2022, pp. 9118–9147.

[574] T. Carta, C. Romac, T. Wolf, S. Lamprier, O. Sigaud, and P. Oudeyer, "Grounding large language models in interactive environments with online reinforcement learning," *CoRR*, vol. abs/2302.02662, 2023.

[575] X. Zhu, Y. Chen, H. Tian, C. Tao, W. Su, C. Yang, G. Huang, B. Li, L. Lu, X. Wang, Y. Qiao, Z. Zhang, and J. Dai, "Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory," *CoRR*, vol. abs/2305.17144, 2023.

[576] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, "Voyager: An open-ended embodied agent with large language models," *CoRR*, vol. abs/2305.16291, 2023.

[577] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, and M. Yan, "Do as I can, not as I say: Grounding language in robotic affordances," *CoRR*, vol. abs/2204.01691, 2022.

[578] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," *CoRR*, vol. abs/2209.07753, 2022.

[579] Y. Fu, H. Peng, T. Khot, and M. Lapata, "Improving language model negotiation with self-play and in-context learning from AI feedback," *CoRR*, vol. abs/2305.10142, 2023.

[580] N. Mehta, M. Teruel, P. F. Sanz, X. Deng, A. H. Awadallah, and J. Kiseleva, "Improving grounded language understanding in a collaborative environment by interacting with agents through help feedback," *CoRR*, vol. abs/2304.10750, 2023.

[581] S. G. Patil, T. Zhang, X. Wang, and J. E. Gonzalez, "Gorilla: Large language model connected with massive apis," *CoRR*, vol. abs/2305.15334, 2023.

[582] S. Hao, T. Liu, Z. Wang, and Z. Hu, "Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings," *CoRR*, vol. abs/2305.11554, 2023.

[583] Y. Liang, C. Wu, T. Song, W. Wu, Y. Xia, Y. Liu, Y. Ou, S. Lu, L. Ji, S. Mao, Y. Wang, L. Shou, M. Gong, and N. Duan, "Taskmatrix.ai: Completing tasks by connecting foundation models with millions of apis," *CoRR*, vol. abs/2303.16434, 2023.

[584] T. Cai, X. Wang, T. Ma, X. Chen, and D. Zhou, "Large language models as tool makers," *CoRR*, vol. abs/2305.17126, 2023.

[585] J. Huang, S. S. Gu, L. Hou, Y. Wu, X. Wang, H. Yu, and J. Han, "Large language models can self-improve," *CoRR*, vol. abs/2210.11610, 2022.

[586] E. Beeching, C. Fourrier, N. Habib, S. Han, N. Lambert, N. Rajani, O. Sanseviero, L. Tunstall, and T. Wolf, "Open llm leaderboard," https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard, 2023.

[587] W. Zhong, R. Cui, Y. Guo, Y. Liang, S. Lu, Y. Wang, A. Saied, W. Chen, and N. Duan, "Agieval: A human-centric benchmark for evaluating foundation models," *CoRR*, vol. abs/2304.06364, 2023.

[588] H. Zeng, "Measuring massive multitask chinese understanding," *CoRR*, vol. abs/2304.12986, 2023.

[589] C. Liu, R. Jin, Y. Ren, L. Yu, T. Dong, X. Peng, S. Zhang, J. Peng, P. Zhang, Q. Lyu, X. Su, Q. Liu, and D. Xiong, "M3KE: A massive multi-level multi-subject knowledge evaluation benchmark for chinese large language models," *CoRR*, vol. abs/2305.10263, 2023.

[590] Y. Huang, Y. Bai, Z. Zhu, J. Zhang, J. Zhang, T. Su, J. Liu, C. Lv, Y. Zhang, J. Lei, Y. Fu, M. Sun, and J. He, "C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models," *CoRR*, vol. abs/2305.08322, 2023.

[591] Z. Gu, X. Zhu, H. Ye, L. Zhang, J. Wang, S. Jiang, Z. Xiong, Z. Li, Q. He, R. Xu, W. Huang, W. Zheng, H. Feng, and Y. Xiao, "Xiezhi: An ever-updating benchmark for holistic domain knowledge evaluation," *CoRR*, vol. abs/2306.05783, 2023.

[592] O. Contributors, "Opencompass: A universal evaluation platform for foundation models," https://github.com/InternLM/OpenCompass, 2023.

[593] Y. Fu, L. Ou, M. Chen, Y. Wan, H. Peng, and T. Khot, "Chain-of-thought hub: A continuous effort to measure large language models' reasoning performance," *CoRR*, vol. abs/2305.17306, 2023.

[594] J. Yu, X. Wang, S. Tu, S. Cao, D. Zhang-li, X. Lv, H. Peng, Z. Yao, X. Zhang, H. Li, C. Li, Z. Zhang, Y. Bai, Y. Liu, A. Xin, N. Lin, K. Yun, L. Gong, J. Chen, Z. Wu, Y. Qi, W. Li, Y. Guan, K. Zeng, J. Qi, H. Jin, J. Liu, Y. Gu, Y. Yao, N. Ding, L. Hou, Z. Liu, B. Xu, J. Tang, and J. Li, "Kola: Carefully benchmarking world knowledge of large language models," *CoRR*, vol. abs/2306.09296, 2023.

[595] T. Sawada, D. Paleka, A. Havrilla, P. Tadepalli, P. Vi-

das, A. Kranias, J. J. Nay, K. Gupta, and A. Komatsuzaki, "ARB: advanced reasoning benchmark for large language models," *CoRR*, vol. abs/2307.13692, 2023.

[596] Y. Peng, S. Li, W. Gu, Y. Li, W. Wang, C. Gao, and M. R. Lyu, "Revisiting, benchmarking and exploring API recommendation: How far are we?" *IEEE Trans. Software Eng.*, vol. 49, no. 4, pp. 1876–1897, 2023.

[597] M. Li, F. Song, B. Yu, H. Yu, Z. Li, F. Huang, and Y. Li, "Api-bank: A benchmark for tool-augmented llms," *CoRR*, vol. abs/2304.08244, 2023.

[598] Q. Tang, Z. Deng, H. Lin, X. Han, Q. Liang, and L. Sun, "Toolalpaca: Generalized tool learning for language models with 3000 simulated cases," *CoRR*, vol. abs/2306.05301, 2023.

[599] Q. Xu, F. Hong, B. Li, C. Hu, Z. Chen, and J. Zhang, "On the tool manipulation capability of open-source large language models," *CoRR*, vol. abs/2305.16504, 2023.

[600] Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian, S. Zhao, R. Tian, R. Xie, J. Zhou, M. Gerstein, D. Li, Z. Liu, and M. Sun, "Toolllm: Facilitating large language models to master 16000+ real-world apis," *CoRR*, vol. abs/2307.16789, 2023.

[601] Z. Liu, W. Yao, J. Zhang, L. Xue, S. Heinecke, R. Murthy, Y. Feng, Z. Chen, J. C. Niebles, D. Arpit, R. Xu, P. Mui, H. Wang, C. Xiong, and S. Savarese, "BOLAA: benchmarking and orchestrating llm-augmented autonomous agents," *CoRR*, vol. abs/2308.05960, 2023.

[602] X. Liu, H. Yu, H. Zhang, Y. Xu, X. Lei, H. Lai, Y. Gu, H. Ding, K. Men, K. Yang, S. Zhang, X. Deng, A. Zeng, Z. Du, C. Zhang, S. Shen, T. Zhang, Y. Su, H. Sun, M. Huang, Y. Dong, and J. Tang, "Agentbench: Evaluating llms as agents," *CoRR*, vol. abs/2308.03688, 2023.

[603] K. Zhu, J. Wang, J. Zhou, Z. Wang, H. Chen, Y. Wang, L. Yang, W. Ye, N. Z. Gong, Y. Zhang, and X. Xie, "Promptbench: Towards evaluating the robustness of large language models on adversarial prompts," *CoRR*, vol. abs/2306.04528, 2023.

[604] R. S. Shah, K. Chawla, D. Eidnani, A. Shah, W. Du, S. Chava, N. Raman, C. Smiley, J. Chen, and D. Yang, "WHEN FLUE MEETS FLANG: benchmarks and large pre-trained language model for financial domain," *CoRR*, vol. abs/2211.00083, 2022.

[605] N. Guha, D. E. Ho, J. Nyarko, and C. Ré, "Legalbench: Prototyping a collaborative benchmark for legal reasoning," *CoRR*, vol. abs/2209.06120, 2022.

[606] L. Zheng, W. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica, "Judging llm-as-a-judge with mt-bench and chatbot arena," *CoRR*, vol. abs/2306.05685, 2023.

[607] X. Wang, Z. Hu, P. Lu, Y. Zhu, J. Zhang, S. Subramaniam, A. R. Loomba, S. Zhang, Y. Sun, and W. Wang, "Scibench: Evaluating college-level scientific problem-solving abilities of large language models," *CoRR*, vol. abs/2307.10635, 2023.

[608] X. Li, T. Zhang, Y. Dubois, R. Taori, I. Gulrajani, C. Guestrin, P. Liang, and T. B. Hashimoto, "Alpacaeval: An automatic evaluator of instruction-following models," https://github.com/tatsu-lab/alpaca_eval, 2023.

[609] Y. Huang, Q. Zhang, P. S. Yu, and L. Sun, "Trustgpt: A benchmark for trustworthy and responsible large language models," *CoRR*, vol. abs/2306.11507, 2023.

[610] Y. Bai, J. Ying, Y. Cao, X. Lv, Y. He, X. Wang, J. Yu, K. Zeng, Y. Xiao, H. Lyu, J. Zhang, J. Li, and L. Hou, "Benchmarking foundation models with language-model-as-an-examiner," *CoRR*, vol. abs/2306.04181, 2023.

[611] C. Chan, W. Chen, Y. Su, J. Yu, W. Xue, S. Zhang, J. Fu, and Z. Liu, "Chateval: Towards better llm-based evaluators through multi-agent debate," *CoRR*, vol. abs/2308.07201, 2023.

[612] Y. Chang, X. Wang, J. Wang, Y. Wu, K. Zhu, H. Chen, L. Yang, X. Yi, C. Wang, Y. Wang, W. Ye, Y. Zhang, Y. Chang, P. S. Yu, Q. Yang, and X. Xie, "A survey on evaluation of large language models," *CoRR*, vol. abs/2307.03109, 2023.

[613] Z. Zhuang, Q. Chen, L. Ma, M. Li, Y. Han, Y. Qian, H. Bai, Z. Feng, W. Zhang, and T. Liu, "Through the lens of core competency: Survey on evaluation of large language models," *CoRR*, vol. abs/2308.07902, 2023.

[614] M. Suzgun, N. Scales, N. Schärli, S. Gehrmann, Y. Tay, H. W. Chung, A. Chowdhery, Q. V. Le, E. H. Chi, D. Zhou, and J. Wei, "Challenging big-bench tasks and whether chain-of-thought can solve them," *CoRR*, vol. abs/2210.09261, 2022.

[615] J. H. Clark, J. Palomaki, V. Nikolaev, E. Choi, D. Garrette, M. Collins, and T. Kwiatkowski, "Tydi QA: A benchmark for information-seeking question answering in typologically diverse languages," *Trans. Assoc. Comput. Linguistics*, vol. 8, pp. 454–470, 2020.

[616] L. Gao, J. Tow, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, K. McDonell, N. Muennighoff, J. Phang, L. Reynolds, E. Tang, A. Thite, B. Wang, K. Wang, and A. Zou, "A framework for few-shot language model evaluation," Sep. 2021.

[617] R. Shah, K. Chawla, D. Eidnani, A. Shah, W. Du, S. Chava, N. Raman, C. Smiley, J. Chen, and D. Yang, "When flue meets flang: Benchmarks and large pre-trained language model for financial domain," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022, pp. 2322–2335.

[618] C. Zan, K. Peng, L. Ding, B. Qiu, B. Liu, S. He, Q. Lu, Z. Zhang, C. Liu, W. Liu, Y. Zhan, and D. Tao, "Vega-mt: The JD explore academy machine translation system for WMT22," in *Proceedings of the Seventh Conference on Machine Translation, WMT 2022, Abu Dhabi, United Arab Emirates (Hybrid), December 7-8, 2022*, P. Koehn, L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussà, C. Federmann, M. Fishel, A. Fraser, M. Freitag, Y. Graham, R. Grundkiewicz, P. Guzman, B. Haddow, M. Huck, A. Jimeno-Yepes, T. Kocmi, A. Martins, M. Morishita, C. Monz, M. Nagata, T. Nakazawa, M. Negri, A. Névéol, M. Neves, M. Popel, M. Turchi, and M. Zampieri, Eds. Association for Computational Linguistics, 2022, pp. 411–422.

[619] Y. Zhao, M. Khalman, R. Joshi, S. Narayan, M. Saleh, and P. J. Liu, "Calibrating sequence likelihood

improves conditional language generation," *CoRR*, vol. abs/2210.00045, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2210.00045

[620] D. Khashabi, S. Min, T. Khot, A. Sabharwal, O. Tafjord, P. Clark, and H. Hajishirzi, "Unifiedqa: Crossing format boundaries with a single QA system," in *EMNLP (Findings)*, ser. Findings of ACL, vol. EMNLP 2020. Association for Computational Linguistics, 2020, pp. 1896–1907.

[621] X. Zhu, J. Wang, L. Zhang, Y. Zhang, R. Gan, J. Zhang, and Y. Yang, "Solving math word problem via cooperative reasoning induced language models," *arXiv preprint arXiv:2210.16257*, 2022.

[622] A. Nguyen, N. Karampatziakis, and W. Chen, "Meet in the middle: A new pre-training paradigm," *CoRR*, vol. abs/2303.07295, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2303.07295

[623] H. Li, J. Zhang, C. Li, and H. Chen, "RESDSQL: decoupling schema linking and skeleton parsing for text-to-sql," *CoRR*, vol. abs/2302.05965, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2302.05965

[624] W. Kang and J. J. McAuley, "Self-attentive sequential recommendation," in *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018*. IEEE Computer Society, 2018, pp. 197–206.

[625] B. Yang, C. Han, Y. Li, L. Zuo, and Z. Yu, "Improving conversational recommendation systems' quality with context-aware item meta-information," in *Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, M. Carpuat, M. de Marneffe, and I. V. M. Ruíz, Eds. Association for Computational Linguistics, 2022, pp. 38–48.

[626] E. Almazrouei, H. Alobeidli, A. Alshamsi, A. Cappelli, R. Cojocaru, M. Debbah, E. Goffinet, D. Heslow, J. Launay, Q. Malartic, B. Noune, B. Pannier, and G. Penedo, "Falcon-40B: an open large language model with state-of-the-art performance," 2023.

[627] S. K. K. Santu and D. Feng, "Teler: A general taxonomy of LLM prompts for benchmarking complex tasks," *CoRR*, vol. abs/2305.11430, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2305.11430

[628] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt, "A prompt pattern catalog to enhance prompt engineering with chatgpt," *arXiv preprint arXiv:2302.11382*, 2023.

[629] OpenAI, "Gpt best practices," *OpenAI*, 2023. [Online]. Available: https://platform.openai.com/docs/guides/gpt-best-practices

[630] Contributors, "Ai short," 2023. [Online]. Available: https://www.aishort.top/

[631] ——, "Awesome chatgpt prompts," *Github*, 2023. [Online]. Available: https://github.com/f/awesome-chatgpt-prompts/

[632] V. Liu and L. B. Chilton, "Design guidelines for prompt engineering text-to-image generative models," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, 2022, pp. 1–23.

[633] L. Beurer-Kellner, M. Fischer, and M. Vechev, "Prompting is programming: A query language for large language models," *Proceedings of the ACM on Programming Languages*, vol. 7, no. PLDI, pp. 1946–1969, 2023.

[634] P. Lu, B. Peng, H. Cheng, M. Galley, K.-W. Chang, Y. N. Wu, S.-C. Zhu, and J. Gao, "Chameleon: Plug-and-play compositional reasoning with large language models," *arXiv preprint arXiv:2304.09842*, 2023.

[635] Y. Hou, J. Zhang, Z. Lin, H. Lu, R. Xie, J. J. McAuley, and W. X. Zhao, "Large language models are zero-shot rankers for recommender systems," *CoRR*, vol. abs/2305.08845, 2023.

[636] S. Chang and E. Fosler-Lussier, "How to prompt llms for text-to-sql: A study in zero-shot, single-domain, and cross-domain settings," *CoRR*, vol. abs/2305.11853, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2305.11853

[637] R. Tang, X. Han, X. Jiang, and X. Hu, "Does synthetic data generation of llms help clinical text mining?" *arXiv preprint arXiv:2303.04360*, 2023.

[638] O. Nov, N. Singh, and D. M. Mann, "Putting chatgpt's medical advice to the (turing) test," *CoRR*, vol. abs/2301.10035, 2023.

[639] K. Yang, S. Ji, T. Zhang, Q. Xie, and S. Ananiadou, "On the evaluations of chatgpt and emotion-enhanced prompting for mental health analysis," *CoRR*, vol. abs/2304.03347, 2023.

[640] K. Jeblick, B. Schachtner, J. Dexl, A. Mittermeier, A. T. Stüber, J. Topalis, T. Weber, P. Wesp, B. O. Sabel, J. Ricke, and M. Ingrisch, "Chatgpt makes medicine easy to swallow: An exploratory case study on simplified radiology reports," *CoRR*, vol. abs/2212.14882, 2022.

[641] K. Singhal, T. Tu, J. Gottweis, R. Sayres, E. Wulczyn, L. Hou, K. Clark, S. Pfohl, H. Cole-Lewis, D. Neal, M. Schaekermann, A. Wang, M. Amin, S. Lachgar, P. A. Mansfield, S. Prakash, B. Green, E. Dominowska, B. A. y Arcas, N. Tomasev, Y. Liu, R. Wong, C. Semturs, S. S. Mahdavi, J. K. Barral, D. R. Webster, G. S. Corrado, Y. Matias, S. Azizi, A. Karthikesalingam, and V. Natarajan, "Towards expert-level medical question answering with large language models," *CoRR*, vol. abs/2305.09617, 2023.

[642] S. Chen, B. H. Kann, M. B. Foote, H. J. Aerts, G. K. Savova, R. H. Mak, and D. S. Bitterman, "The utility of chatgpt for cancer treatment information," *medRxiv*, 2023.

[643] K. Malinka, M. Peresíni, A. Firc, O. Hujnak, and F. Janus, "On the educational impact of chatgpt: Is artificial intelligence ready to obtain a university degree?" *CoRR*, vol. abs/2303.11146, 2023.

[644] T. Susnjak, "Chatgpt: The end of online exam integrity?" *CoRR*, vol. abs/2212.09292, 2022.

[645] K. Tan, T. Pang, and C. Fan, "Towards applying powerful large ai models in classroom teaching: Opportunities, challenges and prospects," 2023.

[646] F. Kamalov and I. Gurrib, "A new era of artificial intelligence in education: A multifaceted revolution," *CoRR*, vol. abs/2305.18303, 2023.

[647] E. Kasneci, K. Seßler, S. Küchemann, M. Bannert,

D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Günnemann, E. Hüllermeier *et al.*, "Chatgpt for good? on opportunities and challenges of large language models for education," *Learning and Individual Differences*, vol. 103, p. 102274, 2023.

[648] A. Blair-Stanek, N. Holzenberger, and B. V. Durme, "Can GPT-3 perform statutory reasoning?" *CoRR*, vol. abs/2302.06100, 2023.

[649] D. Trautmann, A. Petrova, and F. Schilder, "Legal prompt engineering for multilingual legal judgement prediction," *CoRR*, vol. abs/2212.02199, 2022.

[650] J. H. Choi, K. E. Hickman, A. Monahan, and D. Schwarcz, "Chatgpt goes to law school," *Available at SSRN*, 2023.

[651] J. J. Nay, "Law informs code: A legal informatics approach to aligning artificial intelligence with humans," *CoRR*, vol. abs/2209.13020, 2022.

[652] F. Yu, L. Quartey, and F. Schilder, "Legal prompting: Teaching a language model to think like a lawyer," *CoRR*, vol. abs/2212.01326, 2022.

[653] D. Trautmann, A. Petrova, and F. Schilder, "Legal prompt engineering for multilingual legal judgement prediction," *CoRR*, vol. abs/2212.02199, 2022.

[654] A. Tamkin, M. Brundage, J. Clark, and D. Ganguli, "Understanding the capabilities, limitations, and societal impact of large language models," *CoRR*, vol. abs/2102.02503, 2021.

[655] Z. Sun, "A short survey of viewing large language models in legal aspect," *CoRR*, vol. abs/2303.09136, 2023.

[656] A. Abid, M. Farooqi, and J. Zou, "Persistent anti-muslim bias in large language models," in *AIES '21: AAAI/ACM Conference on AI, Ethics, and Society, Virtual Event, USA, May 19-21, 2021*, M. Fourcade, B. Kuipers, S. Lazar, and D. K. Mulligan, Eds. ACM, 2021, pp. 298–306.

[657] A. Shah and S. Chava, "Zero is not hero yet: Benchmarking zero-shot performance of llms for financial tasks," *CoRR*, vol. abs/2305.16633, 2023.

[658] D. Araci, "Finbert: Financial sentiment analysis with pre-trained language models," *CoRR*, vol. abs/1908.10063, 2019.

[659] J. C. S. Alvarado, K. Verspoor, and T. Baldwin, "Domain adaption of named entity recognition to support credit risk assessment," in *Proceedings of the Australasian Language Technology Association Workshop, ALTA 2015, Parramatta, Australia, December 8 - 9, 2015*, B. Hachey and K. Webster, Eds. ACL, 2015, pp. 84–90.

[660] G. Son, H. Jung, M. Hahm, K. Na, and S. Jin, "Beyond classification: Financial reasoning in state-of-the-art language models," *CoRR*, vol. abs/2305.01505, 2023.

[661] X. Zhang, Q. Yang, and D. Xu, "Xuanyuan 2.0: A large chinese financial chat model with hundreds of billions parameters," *arXiv preprint arXiv:2305.12002*, 2023.

[662] H. Yang, X.-Y. Liu, and C. D. Wang, "Fingpt: Open-source financial large language models," *CoRR*, vol. abs/2306.06031, 2023.

[663] Q. Jin, B. Dhingra, Z. Liu, W. W. Cohen, and X. Lu, "Pubmedqa: A dataset for biomedical research question answering," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, 2019, pp. 2567–2577.

[664] A. Krithara, A. Nentidis, K. Bougiatiotis, and G. Paliouras, "Bioasq-qa: A manually curated corpus for biomedical question answering," 2022.

[665] C. Zhang, C. Zhang, C. Li, Y. Qiao, S. Zheng, S. K. Dam, M. Zhang, J. U. Kim, S. T. Kim, J. Choi, G. Park, S. Bae, L. Lee, P. Hui, I. S. Kweon, and C. S. Hong, "One small step for generative ai, one giant leap for AGI: A complete survey on chatgpt in AIGC era," *CoRR*, vol. abs/2304.06488, 2023.

[666] M. Haman and M. Skolnik, "Using chatgpt to conduct a literature review." *Accountability in research*, 2023.

[667] Ö. Aydın and E. Karaarslan, "Openai chatgpt generated literature review: Digital twin in healthcare," *SSRN Electronic Journal*, 2022.

[668] Y. J. Park, D. Kaplan, Z. Ren, C. Hsu, C. Li, H. Xu, S. Li, and J. Li, "Can chatgpt be used to generate scientific hypotheses?" *CoRR*, vol. abs/2304.12208, 2023.

[669] M. M. Hassan, R. A. Knipper, and S. K. K. Santu, "Chatgpt as your personal data scientist," *CoRR*, vol. abs/2305.13657, 2023.

[670] L. Cheng, X. Li, and L. Bing, "Is GPT-4 a good data analyst?" *CoRR*, vol. abs/2305.15038, 2023.

[671] S. I. M. Hussam Alkaissi, "Artificial hallucinations in chatgpt: Implications in scientific writing," *PubMed*, 2023.

[672] A. Azaria, R. Azoulay, and S. Reches, "Chatgpt is a remarkable tool – for experts," *CoRR*, vol. abs/2306.03102, 2023.

[673] O. O. Buruk, "Academic writing with GPT-3.5: reflections on practices, efficacy and transparency," *CoRR*, vol. abs/2304.11079, 2023.

[674] R. Liu and N. B. Shah, "Reviewergpt? an exploratory study on using large language models for paper reviewing," *CoRR*, vol. abs/2306.00622, 2023.

[675] M. Kosinski, "Theory of mind may have spontaneously emerged in large language models," *CoRR*, vol. abs/2302.02083, 2023.

[676] M. M. Amin, E. Cambria, and B. W. Schuller, "Will affective computing emerge from foundation models and general ai? A first evaluation on chatgpt," *CoRR*, vol. abs/2303.03186, 2023.

[677] G. Sridhara, R. H. G., and S. Mazumdar, "Chatgpt: A study on its utility for ubiquitous software engineering tasks," *CoRR*, vol. abs/2305.16837, 2023.

[678] W. Sun, C. Fang, Y. You, Y. Miao, Y. Liu, Y. Li, G. Deng, S. Huang, Y. Chen, Q. Zhang, H. Qian, Y. Liu, and Z. Chen, "Automatic code summarization via chatgpt: How far are we?" *CoRR*, vol. abs/2305.12865, 2023.

[679] C. S. Xia and L. Zhang, "Conversational automated program repair," *CoRR*, vol. abs/2301.13246, 2023.

[680] H. Cho, H. J. Kim, J. Kim, S. Lee, S. Lee, K. M. Yoo, and T. Kim, "Prompt-augmented linear probing: Scaling beyond the limit of few-shot in-context learners," *CoRR*, vol. abs/2212.10873, 2022.

[681] S. Shin, S. Lee, H. Ahn, S. Kim, H. Kim, B. Kim, K. Cho, G. Lee, W. Park, J. Ha, and N. Sung, "On the effect of pretraining corpora on in-context learning by a large-scale language model," in *NAACL-HLT*. Association

for Computational Linguistics, 2022, pp. 5168–5186.
[682] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *ACM Comput. Surv.*, vol. 55, no. 6, pp. 109:1–109:28, 2023.
[683] W. Kuang, B. Qian, Z. Li, D. Chen, D. Gao, X. Pan, Y. Xie, Y. Li, B. Ding, and J. Zhou, "Federatedscope-llm: A comprehensive package for fine-tuning large language models in federated learning," 2023.