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

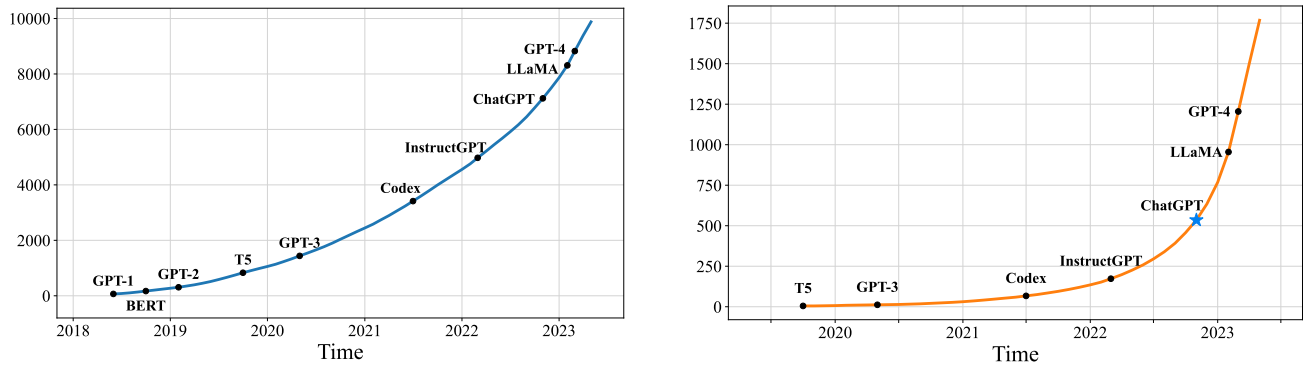
Language is a prominent ability in human beings to express and communicate, which develops in early childhood and evolves over a lifetime [3, 4]. Machines, however, cannot naturally grasp the abilities of understanding and communicating in the form of human language, unless equipped with powerful artificial intelligence (AI) algorithms. It has been a longstanding research challenge to achieve this goal, to enable machines to read, write, and communicate like humans [5].

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 [6–9] 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) [10, 11] and natural language processing (NLP) [12–14]. 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 [15] and Good–Turing estimation [16] have been introduced to alleviate the data sparsity problem.

• *Neural language models (NLM)*. NLMs [1, 17, 18] characterize the probability of word sequences by neural networks, *e.g.,* multi-layer perceptron (MLP) and recurrent neural networks (RNNs). As a remarkable contribution, the work in [1] 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 text data, a general neural network approach was developed to build a unified, end-to-end solution for

• *Version: v16 (major update on March 11, 2025).*
• *GitHub link: https://github.com/RUCAIBox/LLMSurvey*
• *Chinese book link: lmbook-zh.github.io*
• ** 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)).

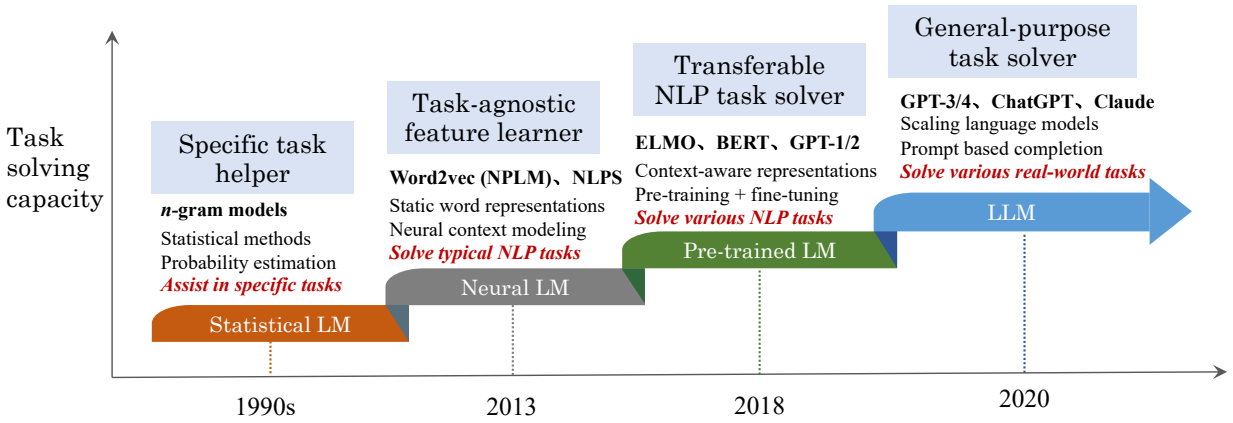


Fig. 2: An evolution process of the four generations of language models (LM) from the perspective of task solving capacity. Note that the time period for each stage may not be very accurate, and we set the time mainly according to the publish date of the most representative studies at each stage. For neural language models, we abbreviate the paper titles of two representative studies to name the two approaches: NPLM [1] ("*A neural probabilistic language model*") and NLPS [2] ("*Natural language processing (almost) from scratch*"). Due to the space limitation, we don't list all representative studies in this figure.

various NLP tasks [2]. Furthermore, 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. Furthermore, 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.

As discussed before, language model is not a new technical concept specially for LLMs, but has evolved with the advance of artificial intelligence over the decades. Early language models mainly aim to model and generate text data, while latest language models (*e.g.,* GPT-4) focus on complex task solving. From *language modeling* to *task solving*, it is an important leap in scientific thinking, which is the key to understand the development of language models in the research history. From the perspective of task solving, the four generations of language models have exhibited different levels of model capacities. In Figure 2, we describe the evolution process of language models in terms of the task solving capacity. At first, statistical language models mainly assisted in some specific tasks (*e.g.,* retrieval or speech tasks), in which the predicted or estimated probabilities can enhance the performance of task-specific approaches. Subsequently, neural language models focused on learning task-agnostic representations (*e.g.,* features), aiming to reduce the efforts for human feature engineering. Furthermore, pre-trained language models learned context-aware representations that can be optimized according to downstream tasks. For the latest generation of language model, LLMs are enhanced by exploring the scaling effect on model capacity, which can be considered as general-purpose task solvers. To summarize, in the evolution process, the task scope that can be solved by language models have been greatly extended, and the task performance attained by language models have been significantly enhanced.

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 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

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/

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

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.

Formulation of 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 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, and a follow-up study [58] from OpenAI has shown that the language modeling loss can be decomposed into two parts, namely *irreducible loss* (the entropy of the true data distribution) and *reducible loss* (an estimate of the KL divergence between the true and model distributions). The three laws were derived by fitting the model performance with varied data sizes (22M to 23B tokens), model sizes (768 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],

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.

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.

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).

Discussion on Scaling Laws. After introducing the formulations, we continue to discuss scaling law in the following two aspects, to enhance its understanding:

• *Predictable scaling*. In practice, scaling law can be used to instruct the training of LLMs, and it has been proven feasible to reliably estimate the performance of larger models based on that of smaller models, called *predictable scaling* [46]. The benefits of predictable scaling for training LLMs are mainly twofold. Firstly, for large models, it is infeasible to rigorously examine various training tricks or variants, and it would be very helpful if experiences gained from small models could also apply to large models. For instance, small proxy models can be trained to find the optimal schedule of the data mixture for large models [59]. Secondly, the training of large-scale models takes a long time, often suffering from issues such as training loss spike, and scaling law can be employed to monitor the training status of LLMs, *e.g.,* identifying abnormal performance at an early time. Despite that scaling law characterizes a smooth trend of performance increase (or loss decrease), it also indicates that *diminishing returns*[7] might occur as model scaling. An empirical study [58] from the OpenAI team has shown that representation quality or semantic content can still effectively improve even if approaching the point of diminishing returns (*i.e.,* approaching the irreducible loss) [58]. This finding suggests that training large models are promising for improving the performance of downstream tasks. To further explore scaling effect, a potential issue is that the amount of available data for training LLMs is actually limited. With the ever-increasing model scale, the public text data would be soon "exhausted" for LLMs [60]. Thus, it will be meaningful to study how scaling laws apply to a data-constrained regime [61], where data repetition or augmentation might be useful to alleviate data scarcity.

• *Task-level predictability*. Existing research of scaling laws are mostly conducted in terms of language modeling loss (*e.g.,* per-token cross-entropy loss in nats [30]), while in practice we are more concerned about the performance of LLMs on actual tasks. Thus, a basic problem is that how the decrease of language modeling loss translates into the improvement of task performance [58]. Intuitively, a model with a smaller language modeling loss tends to yield a better performance on downstream tasks, since language modeling loss can be considered as a general measure of the overall model capacity. GPT-4 [46] has reported that some capabilities (*e.g.,* coding ability) can be accurately predicted via scaling law. Despite that, readers should be aware that a direct decrease in language modeling loss does not always indicate an improvement of model performance on downstream tasks. Specially, the phenomenon of *inverse scaling* would occur for some tasks, where task performance surprisingly becomes worse as the language modeling loss decreases [62]. Overall, it is more difficult to explore and characterize task-level scaling laws, since it might be also dependent on task-related information (task metric, task difficulty, etc.). Furthermore, some capacities (*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, 63]. In principle, emergent abilities can be defined in relation to some complex tasks [31, 64], 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[8].

• *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[9]. 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, 66, 67]. 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 [67], instruction-tuned LaMDA-PT [68] 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 [69] 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

7. https://en.wikipedia.org/wiki/Diminishing_returns

8. 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.

9. In a recent study [65], it also shows that in-context learning implicitly performs meta-optimization through the attention mechanism.

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 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].

How Emergent Abilities Relate to Scaling Laws. In existing literature [30, 31, 34], scaling laws and emergent abilities provide two perspectives to understand the advantage of large models over small models. In general, scaling law (often measured by *language modeling loss*) describes predictable performance relation with the potential effect of diminishing returns, while emergent abilities (often measured by *task performance*) are unpredictable but very profitable once such abilities actually emerge. Since the two perspectives reflect different performance trends (continuous improvement *v.s.* sharp performance leap), they might lead to misaligned findings or observations. There are also extensive debates on the rationality of emergent abilities. A popular speculation is that emergent abilities might be partially attributed to the evaluation setting for special tasks (*e.g.,* the discontinuous evaluation metrics) [70, 71]: when evaluation metrics are altered accordingly, the sharpness of the emergent ability curve would disappear. However, the performance of LLMs on most tasks are perceived by users naturally in a discontinuous way. For instance, end users prefer a reliable code generated by LLMs that can successfully pass the test case, but are less interested in selecting a better code with fewer errors between two failed ones. More recently, a study [72] proposes a new evaluation setting that can enlarge the resolution of task metrics, making task performance more predictable. Despite these efforts, more fundamental research (*e.g.,* grokking[10]) about the working mechanism of LLMs is still in need to understand the emergence of certain abilities. The subtle relation between scaling law and emergent abilities can be explained by analogy with the ability acquisition of human[11]. Take the speaking ability as an example. For children, language development (especially infants) can be also considered as a multi-level process where "emergent abilities" occur. Specially, the language ability would relatively stable within a time interval, but qualitative change only occurs when evolving into another ability level (*e.g.,* from speaking simple words to speaking simple sentences). Such a learning process is essentially not *smooth* and *stable* (*i.e.,* language ability does not develop at a constant rate over time), though a child actually grows every day. It is interesting that young parents would be often surprised by unexpected progress of the speaking ability exhibited by their babies.

10. Grokking refers that "a pattern in the data, improving generalization performance from random chance level to perfect generalization", quoted from the original paper [73].

11. This explanation is only for ease of understanding, and there is not direct evidence to connect the two points.

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 [74] and Megatron-LM [75–77]. 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 [78]. 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 [66]

designs an effective tuning approach that enables LLMs to follow the expected instructions, which utilizes the technique of *reinforcement learning with human feedback* [66, 79]. 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 [80, 81]. For example, LLMs can utilize the calculator for accurate computation [80] and employ search engines to retrieve unknown information [81]. More recently, ChatGPT has enabled the mechanism of using external plugins (existing or newly created apps)[12], 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 4. 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[13].

Early Explorations. According to one interview with Ilya Sutskever[14] (a co-founder and chief scientist of OpenAI), 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) [121]. With the advent of Transformer, OpenAI developed two initial GPT models, namely GPT-1 [122] 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 [122], 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 pre-training 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 [123]), 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][15]. 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..."[16].

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 [124, 125]. Based on GPT-2, GPT-3

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

13. 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.

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

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

16. https://lifearchitect.ai/ilya/

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 [82]	Oct-2019	11	-	-	-	1T tokens	Apr-2019	1024 TPU v3	-	✓	-
	mT5 [83]	Oct-2020	13	-	-	-	1T tokens	-	-	-	✓	-
	PanGu-α [84]	Apr-2021	13*	-	-	-	1.1TB	-	2048 Ascend 910	-	✓	-
	CPM-2 [85]	Jun-2021	198	-	-	-	2.6TB	-	-	-	-	-
	T0 [28]	Oct-2021	11	T5	✓	-	-	-	512 TPU v3	27 h	✓	-
	CodeGen [86]	Mar-2022	16	-	-	-	577B tokens	-	-	-	✓	-
	GPT-NeoX-20B [87]	Apr-2022	20	-	-	-	825GB	-	96 40G A100	-	✓	-
	Tk-Instruct [88]	Apr-2022	11	T5	✓	-	-	-	256 TPU v3	4 h	✓	-
	UL2 [89]	May-2022	20	-	-	-	1T tokens	Apr-2019	512 TPU v4	-	✓	✓
	OPT [90]	May-2022	175	-	-	-	180B tokens	-	992 80G A100	-	✓	-
	NLLB [91]	Jul-2022	54.5	-	-	-	-	-	-	-	✓	-
	CodeGeeX [92]	Sep-2022	13	-	-	-	850B tokens	-	1536 Ascend 910	60 d	✓	-
	GLM [93]	Oct-2022	130	-	-	-	400B tokens	-	768 40G A100	60 d	✓	-
	Flan-T5 [69]	Oct-2022	11	T5	✓	-	-	-	-	-	✓	✓
	BLOOM [78]	Nov-2022	176	-	-	-	366B tokens	-	384 80G A100	105 d	✓	-
	mT0 [94]	Nov-2022	13	mT5	✓	-	-	-	-	-	✓	-
	Galactica [35]	Nov-2022	120	-	-	-	106B tokens	-	-	-	✓	✓
	BLOOMZ [94]	Nov-2022	176	BLOOM	✓	-	-	-	-	-	✓	-
	OPT-IML [95]	Dec-2022	175	OPT	✓	-	-	-	128 40G A100	-	✓	✓
	LLaMA [57]	Feb-2023	65	-	-	-	1.4T tokens	-	2048 80G A100	21 d	✓	-
	Pythia [96]	Apr-2023	12	-	-	-	300B tokens	-	256 40G A100	-	✓	-
	CodeGen2 [97]	May-2023	16	-	-	-	400B tokens	-	-	-	✓	-
	StarCoder [98]	May-2023	15.5	-	-	-	1T tokens	-	512 40G A100	-	✓	✓
	LLaMA2 [99]	Jul-2023	70	-	✓	✓	2T tokens	-	2000 80G A100	-	✓	-
	Baichuan2 [100]	Sep-2023	13	-	✓	✓	2.6T tokens	-	1024 A800	-	✓	-
	QWEN [101]	Sep-2023	14	-	✓	✓	3T tokens	-	-	-	✓	-
	FLM [102]	Sep-2023	101	-	✓	-	311B tokens	-	192 A800	22 d	✓	-
	Skywork [103]	Oct-2023	13	-	-	-	3.2T tokens	-	512 80G A800	-	✓	-
Closed Source	GPT-3 [55]	May-2020	175	-	-	-	300B tokens	-	-	-	✓	-
	GShard [104]	Jun-2020	600	-	-	-	1T tokens	-	2048 TPU v3	4 d	-	-
	Codex [105]	Jul-2021	12	GPT-3	-	-	100B tokens	May-2020	-	-	✓	-
	ERNIE 3.0 [106]	Jul-2021	10	-	-	-	375B tokens	-	384 V100	-	✓	-
	Jurassic-1 [107]	Aug-2021	178	-	-	-	300B tokens	-	800 GPU	-	✓	-
	HyperCLOVA [108]	Sep-2021	82	-	-	-	300B tokens	-	1024 A100	13.4 d	✓	-
	FLAN [67]	Sep-2021	137	LaMDA-PT	✓	-	-	-	128 TPU v3	60 h	✓	-
	Yuan 1.0 [109]	Oct-2021	245	-	-	-	180B tokens	-	2128 GPU	-	✓	-
	Anthropic [110]	Dec-2021	52	-	-	-	400B tokens	-	-	-	✓	-
	WebGPT [81]	Dec-2021	175	GPT-3	-	✓	-	-	-	-	✓	-
	Gopher [64]	Dec-2021	280	-	-	-	300B tokens	-	4096 TPU v3	920 h	✓	-
	ERNIE 3.0 Titan [111]	Dec-2021	260	-	-	-	-	-	-	-	✓	-
	GLaM [112]	Dec-2021	1200	-	-	-	280B tokens	-	1024 TPU v4	574 h	✓	-
	LaMDA [68]	Jan-2022	137	-	-	-	768B tokens	-	1024 TPU v3	57.7 d	-	-
	MT-NLG [113]	Jan-2022	530	-	-	-	270B tokens	-	4480 80G A100	-	✓	-
	AlphaCode [114]	Feb-2022	41	-	-	-	967B tokens	Jul-2021	-	-	-	-
	InstructGPT [66]	Mar-2022	175	GPT-3	✓	✓	-	-	-	-	✓	-
	Chinchilla [34]	Mar-2022	70	-	-	-	1.4T tokens	-	-	-	✓	-
	PaLM [56]	Apr-2022	540	-	-	-	780B tokens	-	6144 TPU v4	-	✓	✓
	AlexaTM [115]	Aug-2022	20	-	-	-	1.3T tokens	-	128 A100	120 d	✓	✓
	Sparrow [116]	Sep-2022	70	-	-	✓	-	-	64 TPU v3	-	✓	-
	WeLM [117]	Sep-2022	10	-	-	-	300B tokens	-	128 A100 40G	24 d	✓	-
	U-PaLM [118]	Oct-2022	540	PaLM	-	-	-	-	512 TPU v4	5 d	✓	✓
	Flan-PaLM [69]	Oct-2022	540	PaLM	✓	-	-	-	512 TPU v4	37 h	✓	✓
	Flan-U-PaLM [69]	Oct-2022	540	U-PaLM	✓	-	-	-	-	-	✓	✓
	GPT-4 [46]	Mar-2023	-	-	✓	✓	-	-	-	-	✓	✓
	PanGu-Σ [119]	Mar-2023	1085	PanGu-α	-	-	329B tokens	-	512 Ascend 910	100 d	✓	-
	PaLM2 [120]	May-2023	16	-	✓	-	100B tokens	-	-	-	✓	✓

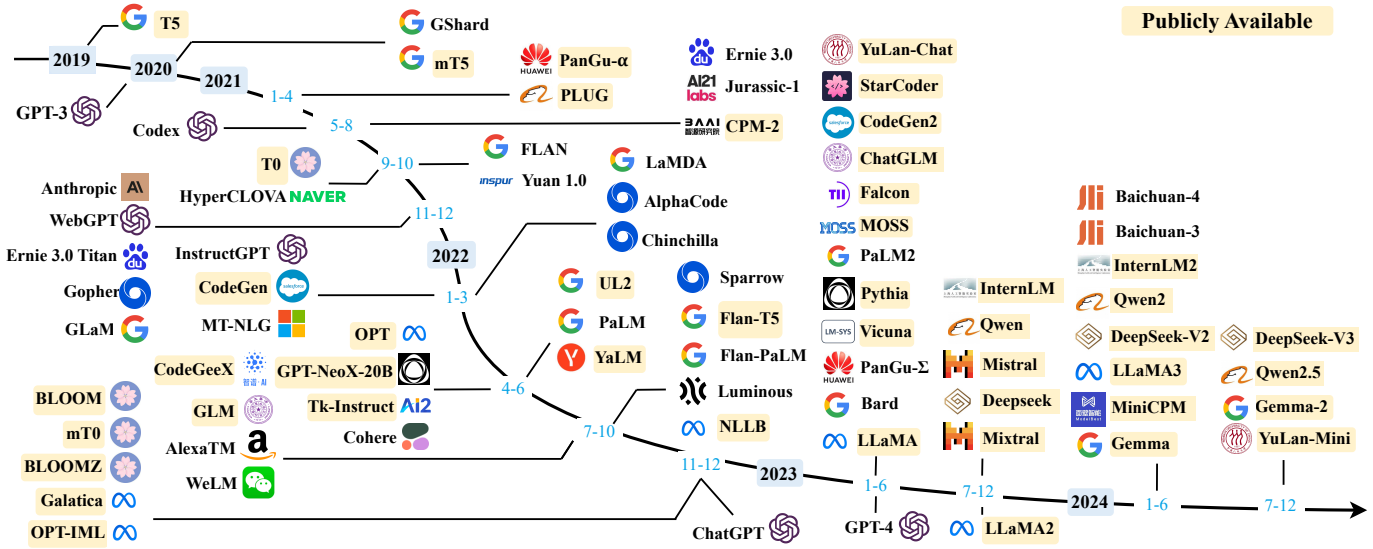


Fig. 3: A timeline of existing large language models (having a size larger than 10B) in recent years. The timeline was established mainly according to the release date (*e.g.*, the submission date to arXiv) of the technical paper for a model. If there was no corresponding paper, we set the date of a 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.

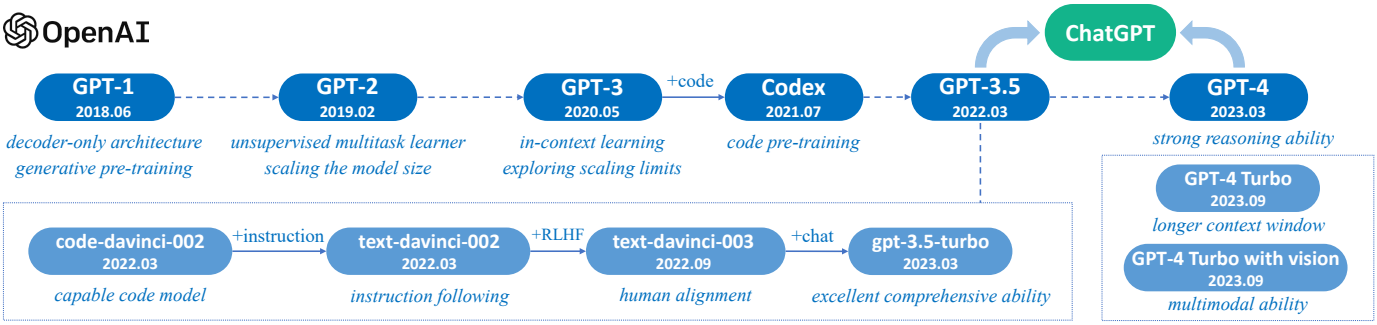


Fig. 4: A brief illustration for the technical evolution of GPT-series models. We plot this figure mainly based on the papers, blog articles and official APIs from OpenAI. Here, *solid lines* denote that there exists an explicit evidence (*e.g.*, the official statement that a new model is developed based on a base model) on the evolution path between two models, while *dashed lines* denote a relatively weaker evolution relation.

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)[17], 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

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

the reasoning ability on complex tasks, *e.g.,* completing the code and solving math problems. To enhance this ability, Codex [105] 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 [126]. Further, a contrastive approach [127] 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"[18] was posted on the OpenAI blog describing a work that applied reinforcement learning (RL) to learn from the *preference comparisons* annotated by humans [79] (similar to the *reward training* step in the aligning algorithm of InstructGPT in Figure 12). Shortly after the release of this RL paper [79], the paper of the Proximal Policy Optimization (PPO) [128] was published in July 2017, which now has been the foundational RL algorithm for learning from human preferences [66]. Later in January 2020, GPT-2 was fine-tuned using the aforementioned RL algorithms [79, 128], which leveraged human preferences to improve the capacities of GPT-2 on NLP tasks. In the same year, another work [129] trained a summarization model for optimizing human preferences in a similar way. Based on these prior work, InstructGPT [66] 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 [66]). In addition to improving the instruction following capacity, the RLHF algorithm is particularly useful to mitigate the issues of generating harm 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 [130], 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 [131] 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 [131], 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. 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* [132] 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.

• *GPT-4V, GPT-4 turbo, and beyond*. Based on the work done for GPT-4 [46], OpenAI further released GPT-4V in September 2023, which focused on the safe deployment of the vision capabilities of GPT-4. In the GPT-4V's system card [133], it has extensively discussed the assessment and mitigation of risks related to visually augmented inputs. Specially, GPT-4V exhibited strong vision capacities in various application scenarios, showing the great potential as a powerful multimodal learning system. More recently, in November 2023, OpenAI released an upgraded generation of GPT-4 model at DevDay, named *GPT-4 Turbo*, with a series of technical improvements. GPT-4 Turbo is featured by the improved model capacity (more capable than GPT-4), the extended knowledge source (up to April 2023),

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

long context window (up to 128k tokens), optimized model performance (cheaper price), and other useful functionality updates (function call, reproducible outputs, etc.). At the same time, Assistants API was launched to ease the rapid development of agent-like assistants. With this API, developers can easily create goal-oriented assistants within their applications, by leveraging specific instruction, extra knowledge and tool use. Furthermore, multimodal capacities (see, hear, and speak) were also enhanced in this new release, supported by GPT-4 Turbo with vision, DALL·E 3, Text-to-speech (TTS), and Listen to voice samples. These improvements have greatly extended the capacity scope and enhanced the task performance of GPT models. More importantly, the application ecosystem will be greatly strengthened with the technology upgrade in improved models, APIs, and functionalities.

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 [134] 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. Due to space limitation, we can only selectively discuss several representative LLMs. 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.

Publicly Available Model Checkpoints. To assist researchers in selecting a suitable model based on the resource budget and usage needs, we focus on discussing the model's parameter size, data and computational resources required for training, the relevant technologies employed by the model, and its performance evaluation in downstream tasks. For more details of LLMs, see Table 1.

• *LLaMA.* The LLaMA series of models has gained immense popularity and widespread attention due to its openness and effectiveness. From LLaMA [57], LLaMA-2 [99], LLaMA-3 [135] to LLaMA-3.1 [136], continuous updates have been made and the development is still ongoing. With increased parameters (the largest version has 405B), more pre-training tokens (15T tokens), and an extended context window (128K), LLaMA-3.1 has significantly enhanced its capabilities, and it also integrates additional components that work in synergy with the model, including new security and safety tools. In evaluation, LLaMa-3.1 (405B version) achieves competitive performance against prominent closed-source LLMs, such as GPT-4, GPT-4o, and Claude 3.5 Sonnet in various benchmarks (*e.g.,* MMLU, GSM8k, and HumanEval). The pre-training of LLaMA (65B version) involves 2,048 A100-80G GPUs, whereas LLaMA-3.1 (405B version) involves more than 16,000 H100 GPUs.

• *Mistral.* The Mistral series [137, 138] consist of Mistral (7B), Mistral NeMo (12B), Mistral Large 2 (123B), and Mixtral (8×7B and 8×22B), which have been widely known for their strong performance on various mainstream benchmarks (*e.g.,* MMLU and GSM8k). Mistral NeMo is featured with a long context window of 128K at the parameter scale of 12B. Although Mistral NeMo is trained with quantization awareness, it enables FP8 inference without sacrificing performance. Mistral Large 2 is the largest and most powerful model of the Mistral series, which supports 11 natural languages and more than 80 programming languages. Mixtral is a kind of sparse Mixture-of-Experts (SMoE) model that activates only part of the parameters during inference, making it more efficient compared to dense models of the same size.

• *Gemma.* Gemma [139, 140] is a series of lightweight, strong, and open models, consisting of Gemma-1 (2B and 7B) and Gemma-2 (2B, 9B, and 27B). During the pre-training stage, Gemma-2 2B, 9B, and 27B versions are trained on 2T, 8T, and 13T primarily English tokens, respectively. The largest version of Gemma-2 is trained on 6144 TPUv5p chips. Gemma-2 has achieved excellent performance in multiple benchmarks (*e.g.,* ARC-c, MMLU, and GSM8k).

• *Qwen.* Qwen [141, 142] is an open-source large model series consisting of Qwen (raging from 7B to 72B), Qwen1.5 (raging from 0.5B to 110B), Qwen2 (ranging from 0.5B to 72B), and Qwen2.5 (ranging from 0.5B to 72B). Qwen2.5 is the newest LLM collection of Qwen, which is pre-trained on up to 18T tokens. Compared to Qwen2, Qwen2.5 demonstrates a significant increase in knowledge retention, as well as notable advancements in coding and mathematical abilities. Qwen2.5 has also shown large improvements in instruction following, long texts generation (over 8K tokens), structured data understanding and generation (*e.g.,* JSON).

• *GLM.* GLM [143] is a series of LLMs featuring comprehensive capabilities in both English and Chinese. GLM has been upgraded to its fourth-generation model, GLM-4, with a parameter scale of up to 9B, possesses excellent conversational abilities. It has achieved excellent performance in evaluations from multiple perspectives including semantics, mathematics, reasoning, code, and knowledge. In addition to the base model GLM-4-9B, it has open-sourced human preference-aligned model GLM-4-9B-Chat, and long context conversational model GLM-4-9B-Chat-1M.

• *Baichuan*. Baichuan is a series of open-source bilingual LLMs and the latest version is Baichuan-2. Both Baichuan

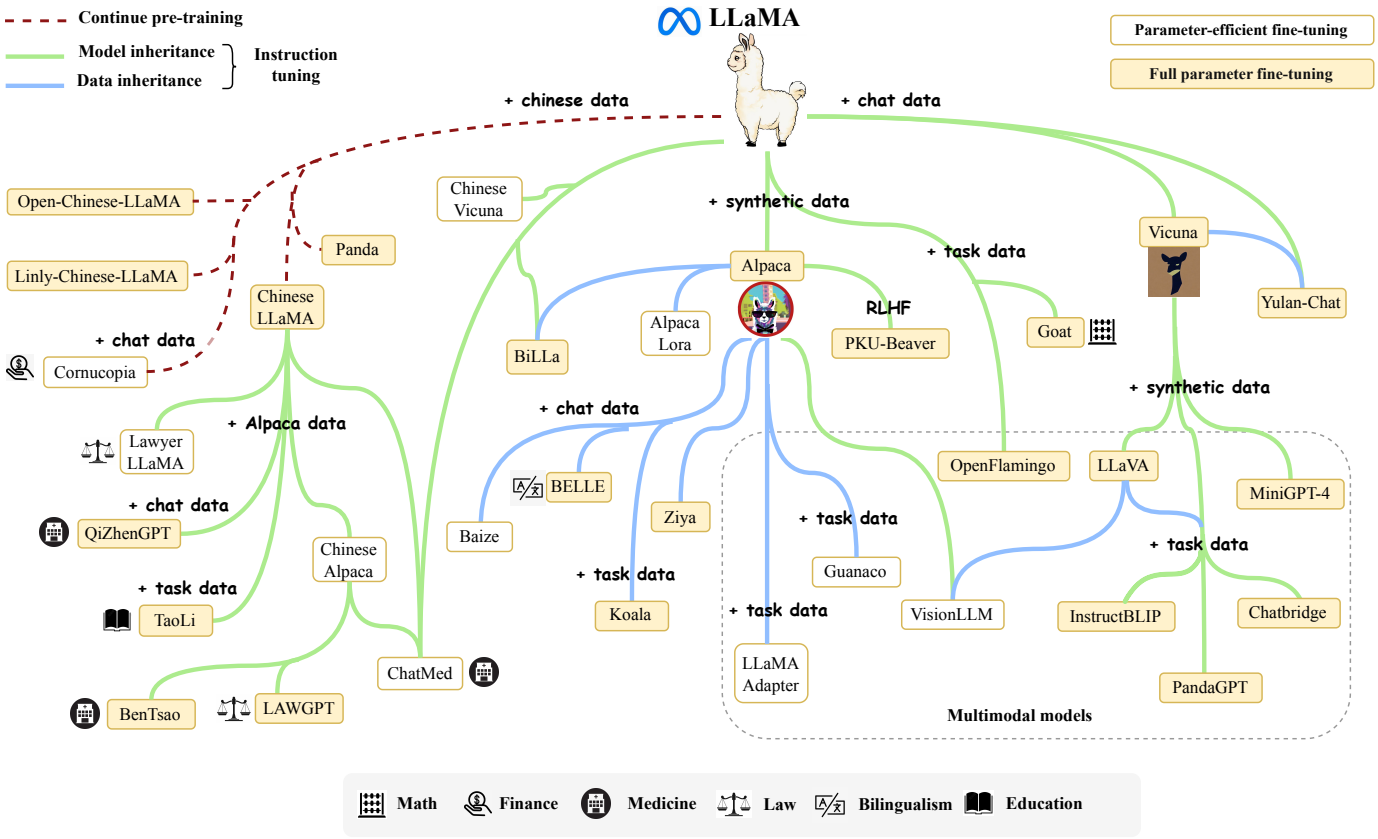


Fig. 5: An evolutionary graph of the research work conducted on LLaMA. Due to the huge number, we cannot include all the LLaMA variants in this figure, even much excellent work. To support incremental update, we share the source file of this figure, and welcome the readers to include the desired models by submitting the pull requests on our GitHub page.

and Baichuan-2 have two available parameter sizes (7B and 13B). Baichuan supports both Chinese and English, with pre-training data reaching 1.2 trillion tokens. Furthermore, Baichuan-2 expands its pre-training data to 2.6 trillion tokens. Baichuan-2 surpasses Baichuan in all evaluation benchmarks, demonstrating excellent multilingual capabilities and showing potential for vertical applications in the domains such as law and healthcare (*e.g.*, JEC-QA [144] and MedQA [145]).

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 pre-training. 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 [146] 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 [147] using `text-davinci-003`. The instruction data, named *Alpaca-52K*, and training code have been extensively adopted in subsequent work, such as Alpaca-LoRA [148] (a reproduction of Stanford Alpaca using LoRA [149]), Koala [150], and BELLE [151]. In addition, Vicuna [152] is another popular LLaMA variant, trained upon user-shared conversations collected from ShareGPT [153]. 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 [154], MiniGPT-4 [155], InstructBLIP [156], and PandaGPT [157]. 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 5.

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, 66, 105] have

been widely used for both academia and industry[19]. 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 [105], called `code-cushman-001` (a powerful and multilingual version of the Codex (12B) [105]) 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`. As more powerful alternatives, in this year, OpenAI has released the model interfaces for GPT-4 series, including `gpt-4`, `gpt-4-32k`, `gpt-4-1106-preview` (*i.e.,* GPT-4 Turbo) and `gpt-4-vision-preview` (*i.e.,* GPT-4 Turbo with vision, a multimodal model). It is worth noting that OpenAI has been maintaining and upgrading these model interfaces (`gpt-3.5-turbo`, `gpt-4`, `gpt-4-32k`), so the API name will actually point to the latest version. Currently, ChatGPT can be powered by either GPT-3.5 or GPT-4 models. Overall, one select the suitable model interface based on the specific application scenarios and response requirements. The detailed usage can be found on their project websites[20].

TABLE 2: Statistics of commonly-used data sources.

Corpora	Size	Source	Latest Update Time
BookCorpus [158]	5GB	Books	Dec-2015
Gutenberg [159]	-	Books	Dec-2021
C4 [82]	800GB	CommonCrawl	Apr-2019
CC-Stories-R [160]	31GB	CommonCrawl	Sep-2019
CC-NEWS [27]	78GB	CommonCrawl	Feb-2019
REALNEWs [161]	120GB	CommonCrawl	Apr-2019
OpenWebText [162]	38GB	Reddit links	Mar-2023
Pushift.io [163]	2TB	Reddit links	Mar-2023
Wikipedia [164]	21GB	Wikipedia	Mar-2023
BigQuery [165]	-	Codes	Mar-2023
the Pile [166]	800GB	Other	Dec-2020
ROOTS [167]	1.6TB	Other	Jun-2022

3.2 Commonly Used Corpora for Pre-training

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 five groups: web pages, books, Wikipedia, code, and others.

Web pages. Web pages are a primary data source for training language models.

• *CommonCrawl.* CommonCrawl [168] 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 it within a specific period or specific needs (*e.g.,* extracting mathematical texts). However, due to the widespread existence of noisy and low-quality information in web page data, it is necessary to perform data preprocessing before usage. One commonly used toolkit for cleaning CommonCrawl is CC-Net [169], which is developed by Facebook and has been used in processing datasets like RedPajama-Data [170].

• *C4.* The Colossal Clean Crawled Corpus (C4) includes five variants[21], namely en (806G), en.noclean (6T), realnewslike (36G), webtextlike (17G), and multilingual (38T). The *en* version has been utilized for pre-training T5 [82], LaMDA [68], Gopher [64], and UL2 [89]. The multilingual C4, also called mC4, has been used in mT5 [83].

• *RedPajama-Data.* RedPajama-Data [170] is a publicly available comprehensive web dataset, comprising 100 billion documents from Common Crawl. It has been cleaned, filtered, and deduplicated using the CCNet tool, resulting in approximately 30T tokens, which is available for download on Hugging Face. RedPajama-Data is a multilingual dataset that includes five languages: English, French, Spanish, German, and Italian. Additionally, it offers over 40 quality labels, making it feasible to filter or reweight the dataset according to specific criteria. The dataset is continuously updated and maintained, with all data processing scripts open-sourced on GitHub for convenient use.

• *RefinedWeb.* RefinedWeb [171] is a web dataset obtained through rigorous selection and deduplication based on data from Common Crawl, encompassing all Common Crawl web records from 2008 to June 2023, totaling around 5T tokens. The open-source portion consists of 600B tokens, with a data size of approximately 500GB. After decompression, it requires 2.8TB of local storage space and is available for download on Hugging Face. This dataset serves as the primary training dataset for the open-source large language model Falcon.

• *WebText.* WebText [26] is a well-known corpus composed of highly upvoted links from Reddit, a social media platform that enables users to submit links and text posts, but it is not publicly available. As a surrogate, there is a readily accessible open-source alternative called OpenWebText [162].

Books & Academic Data. Books and academic data contains a wealth of world knowledge and linguistic information, serving as a high-quality corpus for model learning.

• *Book Data.* BookCorpus [158] is a commonly used dataset in previous small-scale models (*e.g.,* GPT [122] 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 [159], 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 [113] and LLaMA [57]. As for

19. https://platform.openai.com/docs/api-reference/introduction
20. https://platform.openai.com/docs/models/overview
21. https://www.tensorflow.org/datasets/catalog/c4

Books1 [55] and Books2 [55] used in GPT-3 [55], they are much larger than BookCorpus but have not been publicly released so far.

- *Academic Data.* In addition to book data, scientific publication data such as paper is also important for model pre-training. arXiv Dataset [172] is a corpus of 1.7 million academic papers, covering a wide range of papers in the fields of physics, mathematics, and computer science. S2ORC [173] is a corpora that consists of 136M academic papers collected by Semantic Scholar. It also releases a derivative dataset peS2o [174], which contains about 42B tokens.

Wikipedia. Wikipedia [164] 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 [68], 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 [165], 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 [86], a subset of the BigQuery dataset, for training the multilingual version of CodeGen (CodeGen-Multi). In addition, Hugging Face has collected and released a code dataset named The Stack [175], covering more than 30 programming languages. The Stack is continuously updated, and the v1.2 version has expanded to 358 programming languages. Based on this dataset, BigCode further processed it and released StarCoder [98], which is also the pre-training data of the model StarCoder.

Mixed Data. In addition to the aforementioned specific types of datasets, different types of data have been combined to facilitate usage by researchers. The Pile [166] 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) [176], CodeGen (16B) [86], and Megatron-Turing NLG (530B) [113]. ROOTS [167] 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 [78]. Another mixture dataset is Dolma [177], which includes web text from Common Crawl, academic papers from Semantic Scholar, GitHub code, books, social media from Reddit, and Wikipedia data. Dolma consisting of 3T tokens of approximately 200TB of raw text and has been used to train OLMo [178].

In practice, it commonly requires a mixture of different data sources for pre-training LLMs (see Figure 6), 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.

TABLE 3: A detailed list of available collections for instruction tuning.

Categories	Collections	Time	#Examples
Task	Nat. Inst. [179]	Apr-2021	193K
	FLAN [67]	Sep-2021	4.4M
	P3 [180]	Oct-2021	12.1M
	Super Nat. Inst. [88]	Apr-2022	5M
	MVPCorpus [181]	Jun-2022	41M
	xP3 [94]	Nov-2022	81M
	OIG[182]	Mar-2023	43M
Chat	HH-RLHF [183]	Apr-2022	160K
	HC3 [184]	Jan-2023	87K
	ShareGPT [153]	Mar-2023	90K
	Dolly [185]	Apr-2023	15K
	OpenAssistant [186]	Apr-2023	161K
Synthetic	Self-Instruct [147]	Dec-2022	82K
	Alpaca [187]	Mar-2023	52K
	Guanaco [188]	Mar-2023	535K
	Baize [189]	Apr-2023	158K
	BELLE [190]	Apr-2023	1.5M

TABLE 4: A list of available collections for alignment.

Dataset	Release Time	#Examples
Summarize from Feedback [129]	Sep-2020	193K
SHP [191]	Oct-2021	385K
WebGPT Comparisons [81]	Dec-2021	19K
Stack Exchange Preferences [192]	Dec-2021	10M
HH-RLHF [183]	Apr-2022	169K
Sandbox Alignment Data [193]	May-2023	169K
CValues [194]	Jul-2023	145K
PKU-SafeRLHF [195]	Oct-2023	330K

3.3 Commonly Used Datasets for Fine-tuning

After pre-training, it requires further fine-tuning LLMs to enhance the model capacity, which often involve two major steps, namely instruction tuning (supervised fine-tuning) and alignment tuning. In this section, we mainly focus on discussing the related available datasets for the two kinds of tuning approaches, and more algorithm details can be found in Section 5.

3.3.1 Instruction Tuning Datasets

After pre-training, instruction tuning (*a.k.a.,* supervised fine-tuning) is an important method to enhance or unlock specific abilities of LLMs (*e.g.,* instruction following). In this part, we introduce several widely used datasets for instruction tuning, and categorize them into three main types based on the construction method of formatted instruction instances, namely NLP task datasets, daily chat datasets and synthetic datasets. We show their details in Table 3.

NLP Task Datasets. This kind of datasets are formatted based on collected NLP task datasets (*e.g.,* text classification and summarization) with corresponding natural language task descriptions. In this category, P3 [196] and FLAN [67, 197] are two widely used datasets for instruction tuning.

• *P3* [196] is composed of 170 English NLP datasets and 2,052 English prompt templates, where the input and output of each data example have been formatted with specific prompt templates for composing the training instance.

• *FLAN* [67] consists of 62 widely used NLP benchmarks in its original version. Recently, FLAN-v2 [197] is also proposed, which expands FLAN by mixing additional instruction datasets, including Muffin [67], NIV2 [88], T0-SF [28], and CoT [198–200]. Muffin contains 62 tasks from the original FLAN and additional 26 tasks, including conversation and code synthesis tasks. T0-SF is extracted from T0 [28] while ensuring no overlap with Muffin. NIV2 refers to the Natural-Instructions v2 dataset [88], and CoT [198–200] is a combination of nine reasoning tasks with corresponding chain-of-thought prompts and outputs.

Daily Chat Datasets. This kind of datasets are constructed based on real user conversations where queries are posed by humans and responses are mainly generated by human labelers or LLMs (*e.g.,* ChatGPT, GPT-4). The conversation types include open-ended generation, question answering, brainstorming, and chatting. In this category, ShareGPT [153], OpenAssistant [186] and Dolly [185] are three commonly used datasets for LLM fine-tuning.

• *ShareGPT* [153] is collected from a data collection platform where users can upload their conversations with ChatGPT or GPT-4 through the ShareGPT API. Currently, this dataset consists of approximately 90,000 conversations, including real instructions or inquiries from human and responses from ChatGPT.

• *OpenAssistant* [186] is a multilingual corpus containing 66,497 real-world conversation trees between human and AI assistant. Each conversation tree consists of multiple nodes, and each node represents the information generated by a role in the dialogue. It spans 35 languages and includes 461,292 manually annotated quality ratings of responses.

• *Dolly* [185] is an English dataset comprising 15,000 human-generated data instances (prompt-response pairs) from Databricks. This dataset covers seven domains outlined in the InstructGPT [66], including brainstorming, classification, closed-book quality assurance, generation, information extraction, open-book quality assurance, and summarization.

Synthetic Datasets. This kind of datasets are typically constructed by instructing LLMs, based on pre-defined guidance rules or methods. In this category, Self-Instruct-52K [147], Alpaca [146] and Baize [189] are three commonly used synthetic datasets for LLMs.

• *Self-Instruct-52K* [147] is an instruction dataset generated through the self-instruct [147] method, consisting of 82,000 instances with 52,000 instructions. Concretely, the authors construct 175 seed instances, and then iteratively prompt the LLM [55] to synthesize additional instructions based on randomly selected 8 instructions as reference. Subsequently, the LLM is further instructed to generate instance inputs and their corresponding outputs based on the synthetic instructions, and finally obtain the Self-Instruct-52K dataset.

• *Alpaca* [146] is also a synthetic dataset based on the self-instruct [147] method. It utilizes the `text-davinci-003` model on the 175 seed datasets from Self-Instruct-52K to obtain 52,000 new instructions and corresponding inputs and outputs. Moreover, 60% of the examples are pure instructions without the input part in the final dataset.

• *Baize* [189] is an English multi-turn conversation corpus constructed using ChatGPT, comprising 111.5K instances. To create Baize, a method called "self-chat" [189] is purposed, where ChatGPT takes on the roles of both the user and the AI assistant in turns, generating information in a conversational format.

3.3.2 Alignment Datasets

Apart from instruction tuning, it is important to construct high-quality datasets for aligning LLMs with human values and preferences (*e.g.,* helpfulness, honesty, and harmlessness). In this section, we introduce several widely used datasets for alignment tuning, including HH-RLHF [183], SHP [191], PKU-SafeRLHF [195], Stack Exchange Preferences [192] and Sandbox Alignment Data [193]. We show their details in Table 4.

• **HH-RLHF** [183] consists of around 169K instances, and can be divided into two parts that focus on the helpfulness and harmlessness of LLMs, respectively. Each instance is an open-ended conversation between a crowdworker and a chat model, about seeking assistance, advice, or task completion. The chat model provides two responses to each user query, and the more helpful or harmful responses will be chosen as the annotations.

• **SHP** [191] focuses on the helpfulness of responses. It comprises 385K collective human preferences over responses to questions/instructions across 18 diverse subject areas, spanning topics from cooking to legal advice. Each instance is a Reddit post containing a question or instruction and a pair of top-level comments, one of which is deemed as more preferable by Reddit users and the other one is deemed as less helpful. Different from HH-RLHF [183], the data in SHP consists of naturally occurring and human-written responses.

• **PKU-SafeRLHF** [195] encompasses more than 330K instances of expert comparison data, concentrating on the helpfulness and harmlessness. Each instance in the dataset includes a question and two responses, accompanied by safety labels for each response and two preference annotations between the two responses according to helpfulness and harmlessness. The harmlessness of a response indicates its classification as risk-neutral across all 14 harm categories, while the helpfulness of a response is evaluated based on its effectiveness in addressing the question.

• **Stack Exchange Preferences** [192] focuses on the helpfulness of answers. It comprises about 10M questions and answers from Stack Overflow. Each instance consists of a question and more than two corresponding answers. Each answer is annotated with a score calculated based on its votes and a label denoting whether it is selected.

• **Sandbox Alignment Data** [193] is an alignment dataset containing feedback from LLMs rather than human. It

comes from a virtual interaction environment called SANDBOX, where the model simulates social interactions with other models and revise responses according to the feedback from other models. The dataset contains 169K instances, and each instance consists of a societal query, several responses, and corresponding ratings from other models.

3.4 Library Resource

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

• **Transformers** [201] 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** [74] 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 [113] and BLOOM [78]. It provides the support of various optimization techniques for distributed training, such as memory optimization (ZeRO technique, gradient checkpointing), and pipeline parallelism.

• **Megatron-LM** [75–77] 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.

• **JAX** [202] 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** [203] 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 [204]. Recently, a ChatGPT-like model called ColossalChat [205] has been publicly released with two versions (7B and 13B), which are developed using Colossal-AI based on LLaMA [57].

• **BMTrain** [206] 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 [69] and GLM [93]) into its ModelCenter, where developers can use these models directly.

• **FastMoE** [207] 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.

• **vLLM** [208] is a fast, memory efficient, and easy-to-use library for LLM inference and serving. To enable fast inference, it is specially optimized with high serving throughput, effective attention memory management using PagedAttention [208], continuous batching, and optimized CUDA kernels. Furthermore, vLLM also supports various decoding algorithms, tensor parallelism and streaming outputs. To ease the integration with other systems, vLLM is friendly to the use of HuggingFace models, and also provide OpenAI-compatible API servers.

• **DeepSpeed-MII** [209] is also a memory efficient Python library developed by DeepSpeed [74]. It aims to democratize LLMs inference by prioritizing high throughput, low latency, and cost-effectiveness. DeepSpeed-MII achieves accelerated text generation inference by leveraging four essential technologies: blocked KV caching, continuous batching, dynamic SplitFuse, and high-performance CUDA Kernels. It currently supports over 13,000 models across three popular model architectures, such as LLaMA [57], Mistral [137], and OPT [90].

• **DeepSpeed-Chat** [210] is a fast, cost-effective, and easy-to-use system framework that enables the integration of the complete RLHF process during model training. It is featured by three major functionalities: (1) it simplifies the training and inference process for ChatGPT-like models, enabling using a simple script to implement multiple training or inference steps; (2) it replicates the training mode of InstructGPT [66] and provides a complete pipeline for three training steps (*i.e.,* SFT, reward model fine-tuning, and RLHF); (3) it integrates the training engine and inference engine of Deepspeed into a unified hybrid engine (Deepspeed HE) for RLHF training, which enables seamless switch between training and inference modes, and leveraging various optimizations from DeepSpeed Inference.

In addition to the above library resources, existing deep learning frameworks (*e.g.,* PyTorch [211], TensorFlow [212], MXNet [213], PaddlePaddle [214], MindSpore [215] and OneFlow [216]) 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 and Preparation

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

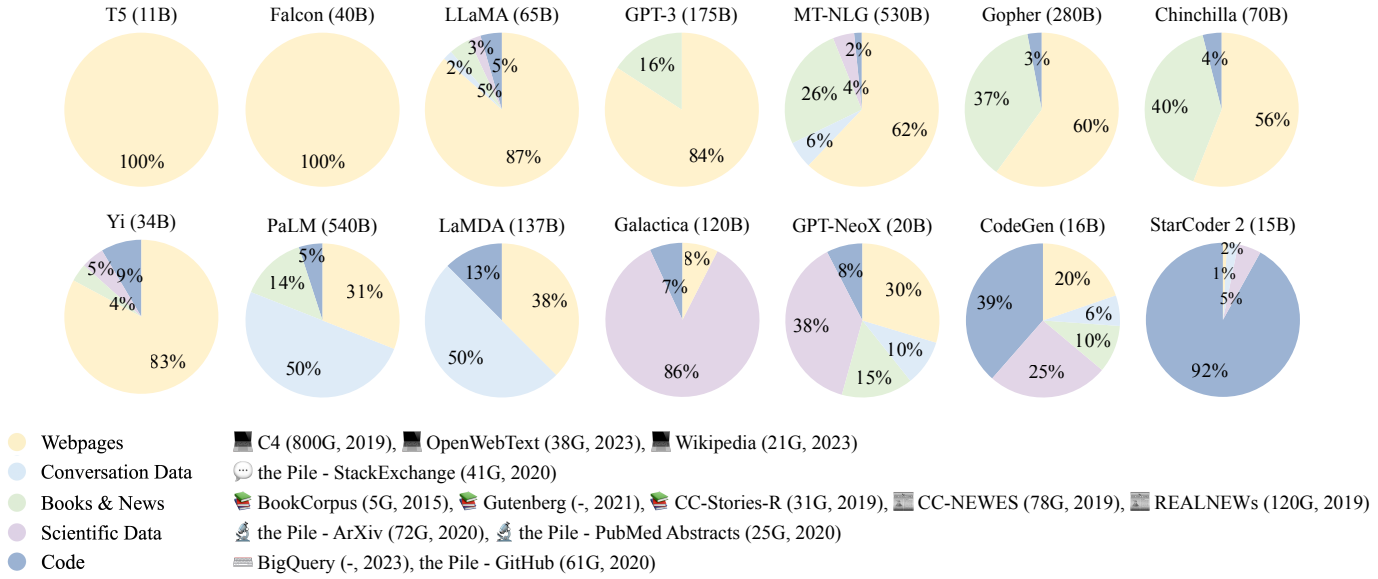


Fig. 6: Ratios of various data sources in the pre-training data for existing LLMs.

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 6 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, 90] 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, 86]. 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 6, 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, 82]. For convenient use of these data resources, a large amount of data is crawled from the web in previous work, such as CommonCrawl [168]. 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 [90] 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) [163, 217] 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 [90]: 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 [166].

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 [78] and PaLM [56] have curated multilingual data covering 46 and 122 languages, respectively, within their pre-training corpora. FLM [102]

mixes Chinese and English corpora in nearly equal proportions. 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, 218], it is useful to incorporate a scientific corpus for model pre-training [35, 218]. By pre-training on a vast amount of scientific text, LLMs can achieve impressive performance in scientific and reasoning tasks [219]. 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 [105, 220–223], especially the use of PLMs trained on code [176, 224]. However, it remains challenging for these PLMs (*e.g.,* GPT-J [176]) to generate high-quality and accurate programs. Recent studies [105, 223] 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 [105] or solve competitive programming questions [114]. 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 [225]. The second source is from public software repositories such as GitHub [86, 105, 223], 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 [226]. 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 [226].

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, 64, 227], which may largely affect the capacity and performance of LLMs. To facilitate the data processing, a recent study [228] 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 [64, 78, 112]. A typical pipeline of preprocessing the pre-training data for LLMs has been illustrated in Figure 7.

Filtering and Selection. To remove low-quality data from the collected corpus, existing work generally adopts two approaches, namely classifier-based and 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 train a binary classifier using positive instances that are: well-curated data (*e.g.,* Wikipedia pages) [55, 56, 112], high-quality synthesized data [135, 229–231], or a combination of both. They sample candidate data as negative instances and predict the score that measures the quality of each data example. However, several studies [64, 112] 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 [78] and Gopher [64], 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.

In addition to the above methods, LLMs (especially relatively small models) can be also employed for data selection, either by computing perplexity [232] or directly prompting LLMs [233] for measuring the sample importance. However, using LLMs is unavoidably computationally intensive for large-scale data selection.

De-duplication. Existing work [234] 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 [235]. 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, 64, 78, 236]. 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

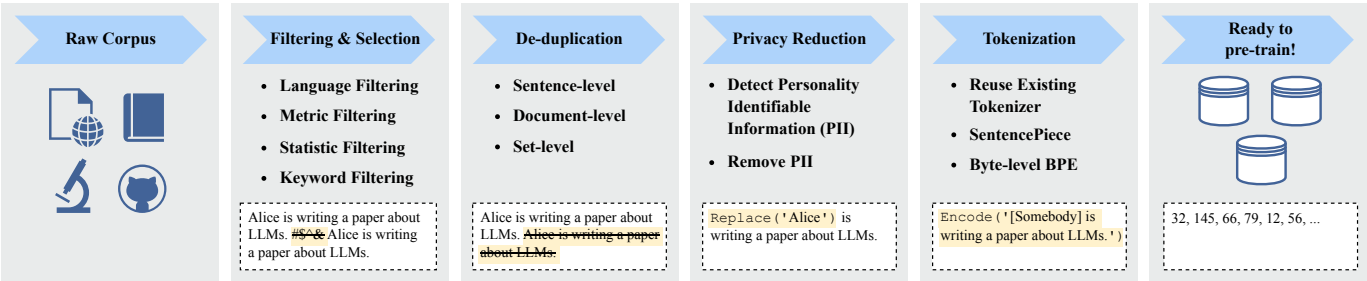


Fig. 7: An illustration of a typical data preprocessing pipeline for pre-training large language models.

levels of de-duplication are useful to improve the training of LLMs [56, 237], which should be jointly used in practice.

Privacy Reduction. 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 [167]. 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 [238]. 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 [239]), 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[22] 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 [240], and then adapted to NLP for tokenization [241]. 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 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 [242]. Then, it was used in the neural machine translation system in 2016 [243], 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 [244] 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 [90] 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 [78], especially for the corpus that consists of diverse domains, languages, and formats. Therefore, recent LLMs often train the customized

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

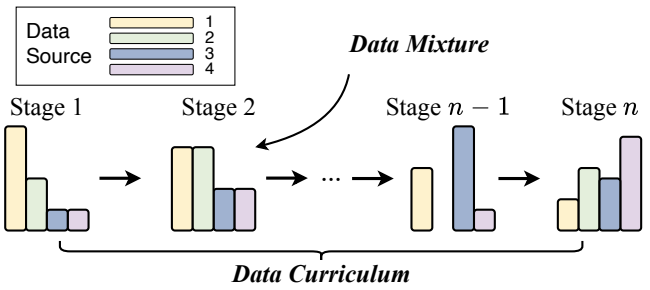


Fig. 8: An illustration of data scheduling for pre-training LLMs.

tokenizers specially for the pre-training corpus with the SentencePiece library [245], which includes Byte-level BPE and Unigram tokenization. A note is that normalization techniques in BPE, such as NFKC [246], may degrade the tokenization performance [34, 64, 78]. 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.

Discussion on Effect of Data Quality. For pre-training, the quality of pre-training data is vital to the model capacities of LLMs. Existing work has shown that pre-training on the low-quality corpus, such as noisy, toxic, and duplicate data, would largely hurt the performance of models [64, 234, 236, 238]. Recent studies, such as T5 [82], GLaM [112], and Gopher [64], have investigated the influence of data quality on the LLMs' capacities. By comparing the performance of models trained on the filtered and unfiltered corpus, they have reached the similar conclusion that pre-training LLMs on cleaned data can improve the model performance. More specifically, the duplication of data may result in "*double descent*" (referring to the phenomenon of performance initially deteriorating and subsequently improving) [234, 247], or even overwhelm the training process [234]. 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 [234]. Therefore, as suggested in [56, 64, 78, 227], it is essential to utilize preprocessing methods like quality filtering, toxic filtering and deduplication to carefully clean the pre-training corpus (as illustrated in Section 4.1.2), to improve stability of the training process and avoid affecting the model performance.

4.1.3 Data Scheduling

After data preprocessing, it is essential to design suitable strategies to schedule these multi-source data for pre-training a capable LLM. Generally, two key aspects should be paid close attention for data scheduling: the proportion of each data source (*data mixture*), and the order in which each data source is scheduled for training (*data curriculum*). Next, we discuss the two aspects in detail. An illustration of data scheduling has been presented in Figure 8.

Data Mixture. Since each kind of data source is closely related to the development of certain capacities for LLMs (referring to the discussions in Section 4.1), it is important to set a suitable distribution to mix these data. The data mixture is generally set in a global level (*i.e.,* the distribution of the entire pre-training data), and can be also locally set to varied proportions at different training stages. During pre-training, data samples from different sources would be selected according to the mixture proportions: more data will be sampled from a data source with a larger weight. Typically, existing LLMs such as LLaMA [57] may employ upsampling or downsampling on the full data of each source to create specific data mixtures as pre-training data. As Figure 6 illustrates, existing LLMs use different data mixtures to construct the pre-training data. As a representative model, the pre-training data of LLaMA [57] mainly consists of webpages (over 80%), alongside 6.5% of code-heavy data from GitHub and StackExchange, 4.5% from books, and 2.5% of scientific data sourced from arXiv, which has become an important reference for training general-purpose LLMs. Furthermore, special data mixtures can be used to facilitate different purposes. For example, Falcon [171] is trained on pure webpages, and CodeGen [86] largely increases the amount of code data. In practice, data mixture is often determined empirically, and we summarize several common strategies for finding an effective data mixture as follows:

- *Increasing the diversity of data sources.* Recent studies have empirically shown that training on excessive data about a certain domain would degrade the generalization capability of LLMs on other domains [35, 64]. In contrast, increasing the data source heterogeneity (*e.g.,* including diverse data sources) is critical for improving the downstream performance of LLMs [227, 248, 249]. To further examine the effect of different data sources, some studies have conducted ablation experiments by removing each data source one by one, and pre-train LLMs with specially curated datasets [227]. It has been shown that dropping data sources with high heterogeneity (*e.g.,* webpages) impacts LLM's abilities more severely than dropping sources with low heterogeneity (*e.g.,* academic corpus).
- *Optimizing data mixtures.* In addition to manually setting the data mixtures, several studies have proposed to optimize the data mixtures for improving the model pre-training [59, 250]. Given the target downstream tasks, one can select pre-training data with either higher proximity in the feature space [250] or those that provide positive influences on downstream task performance [251]. Further, to reduce the reliance of target tasks, DoReMi [59] first trains a small reference model using given initial domain weights, and then trains another small proxy model, upweighting the domains on which the greatest discrepancies in likelihood between the two models are observed. Finally, the learned domain weights of the proxy model are applied to train a much larger LLM. In a more simple way, one can train several small language models with different data mixtures, and select the data mixture that leads to the most desirable performance. However, an assumption made in this approach is, when trained in a similar way, small models would resemble with large models in model abilities or behaviors, which may not always hold in practice.
- *Specializing the targeted abilities.* The model capacities