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

Background for LLMs
Typically, large language models (LLMs) refer to Transformer language models that contain hundreds of billions (or more) of parameters4, 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., languagemodeling) assmall language models. However,LLMs significantly extend the model size, data size, and total 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. 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]. • KMscaling law5. 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 law6: L(N) = Nc N αN , αN ~0.076,Nc ~8.8× 1013 (1) L(D) = Dc D αD , αD~0.095,Dc ~5.4× 1013 L(C) = Cc C αC , αC ~0.050,Cc ~3.1× 108 where L(·) 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 (768M to 1.5B non-embedding parameters) and training compute, under some assumptions (e.g., the analysis of one factor should be not bottlenecked by the other two factors). They showed that the model performance has a strong dependence relation on the three factors. • Chinchilla scaling law. As another representative study, Hoffmann et al. [34] (the Google DeepMind team) proposed an alternative form for scaling laws to instruct the computeoptimal 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+ A Nα + B Dβ , (2) where E = 1.69,A = 406.4,B = 410.7, α = 0.34 and β = 0.28. By optimizing the loss L(N,D) under the constraint C ≈ 6ND, they showed that the optimal allocation of compute budget to model size and data size can be derived as follows: Nopt(C) = G C 6 a , Dopt(C) = G−1 C 6 b , where a = α (3) α+β, b = β α+β and G is a scaling coefficient that can be computed by A, B, α and β. As

analyzed in [34], 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, Nc, Dc and Cc are measured in the number of nonembedding parameters, the number of training tokens and the number of FP-days, respectively. According to the original paper [30], Cc and C should be denoted by Cmin c and Cmin, corresponding to the optimal use of compute. We use the simplified notations for ease of discussions.

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 ability8. • 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 update9. Among the GPTseries 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).

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., grokking10) about the working mechanism of LLMs is still in need to understand the emergence of certain abilities. The subtle relation between scaling law andemergentabilities can be explained by analogy with the ability acquisition of human11. 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).

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] andMegatron-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.