Large Language Models: A Survey

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu
Richard Socher, Xavier Amatriain, Jianfeng Gao

***Abstract*—Large Language Models (LLMs) have drawn a lot of attention due to their strong performance on a wide range of natural language tasks, since the release of ChatGPT in November 2022. LLMs' ability of general-purpose language understanding and generation is acquired by training billions of model's parameters on massive amounts of text data, as predicted by scaling laws [1], [2]. The research area of LLMs, while very recent, is evolving rapidly in many different ways. In this paper, we review some of the most prominent LLMs, including three popular LLM families (GPT, LLaMA, PaLM), and discuss their characteristics, contributions and limitations. We also give an overview of techniques developed to build, and augment LLMs. We then survey popular datasets prepared for LLM training, fine-tuning, and evaluation, review widely used LLM evaluation metrics, and compare the performance of several popular LLMs on a set of representative benchmarks. Finally, we conclude the paper by discussing open challenges and future research directions.**

I. Introduction

Language modeling is a long-standing research topic, dating back to the 1950s with Shannon's application of information theory to human language, where he measured how well simple n-gram language models predict or compress natural language text [3]. Since then, statistical language modeling became fundamental to many natural language understanding and generation tasks, ranging from speech recognition, machine translation, to information retrieval [4], [5], [6].

The recent advances on transformer-based large language models (LLMs), pretrained on Web-scale text corpora, significantly extended the capabilities of language models (LLMs). For example, OpenAI's ChatGPT and GPT-4 can be used not only for natural language processing, but also as general task solvers to power Microsoft's Co-Pilot systems, for instance, can follow human instructions of complex new tasks performing multi-step reasoning when needed. LLMs are thus becoming the basic building block for the development of general-purpose AI agents or artificial general intelligence (AGI).

As the field of LLMs is moving fast, with new findings, models and techniques being published in a matter of months or weeks [7], [8], [9], [10], [11], AI researchers and practitioners often find it challenging to figure out the best recipes to build LLM-powered AI systems for their tasks. This paper gives a timely survey of the recent advances on LLMs. We hope this survey will prove a valuable and accessible resource for students, researchers and developers.

LLMs are large-scale, pre-trained, statistical language models based on neural networks. The recent success of LLMs is an accumulation of decades of research and development of language models, which can be categorized into four waves that have different starting points and velocity: statistical language models, neural language models, pre-trained language models and LLMs.

Statistical language models (SLMs) view text as a sequence of words, and estimate the probability of text as the product of their word probabilities. The dominating form of SLMs are Markov chain models known as the n-gram models, which compute the probability of a word conditioned on its immediate proceeding $n-1$ words. Since word probabilities are estimated using word and n-gram counts collected from text corpora, the model needs to deal with data sparsity (i.e., assigning zero probabilities to unseen words or n-grams) by using *smoothing*, where some probability mass of the model is reserved for unseen n-grams [12]. N-gram models are widely used in many NLP systems. However, these models are incomplete in that they cannot fully capture the diversity and variability of natural language due to data sparsity.

Early neural language models (NLMs) [13], [14], [15], [16] deal with data sparsity by mapping words to low-dimensional continuous vectors (embedding vectors) and predict the next word based on the aggregation of the embedding vectors of its proceeding words using neural networks. The embedding vectors learned by NLMs define a hidden space where the semantic similarity between vectors can be readily computed as their distance. This opens the door to computing semantic similarity of any two inputs regardless their forms (e.g., queries vs. documents in Web search [17], [18], sentences in different languages in machine translation [19], [20]) or modalities (e.g., image and text in image captioning [21], [22]). Early NLMs are task-specific models, in that they are trained on task-specific data and their learned hidden space is task-specific.

Pre-trained language models (PLMs), unlike early NLMs, are task-agnostic. This generality also extends to the learned hidden embedding space. The training and inference of PLMs follows the *pre-training and fine-tuning* paradigm, where language models with recurrent neural networks [23] or transformers [24], [25], [26] are pre-trained on Web-scale unlabeled text corpora for general tasks such as word prediction, and then finetuned to specific tasks using small amounts of (labeled) task-specific data. Recent surveys on PLMs include [8], [27], [28].

Large language models (LLMs) mainly refer to transformer-based neural language models [1] that contain tens to hundreds of billions of parameters, which are pre-trained on massive text data, such as PaLM [31], LLaMA [32], and GPT-4 [33], as summarized in Table III. Compared

[1]Recently, several very promising non-transformer LLMs have been proposed, such as the LLMs based on structured state space models [29], [30]. See Section VII for more details.

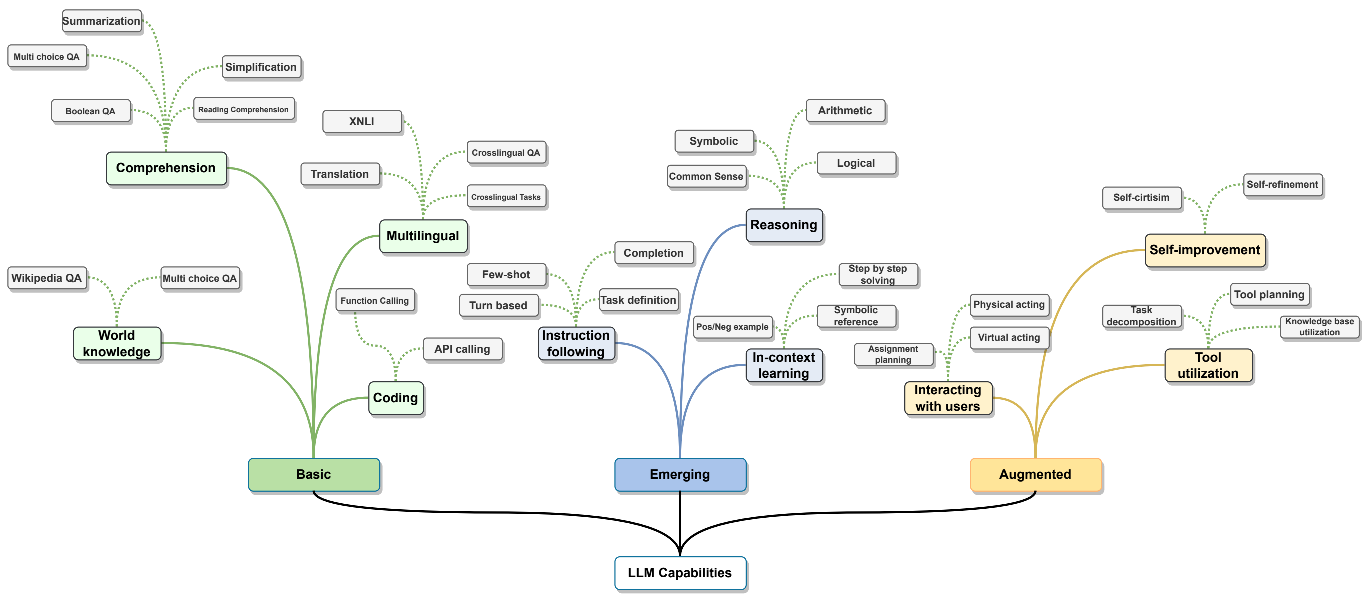

Fig. 1: LLM Capabilities.

to PLMs, LLMs are not only much larger in model size, but also exhibit stronger language understanding and generation abilities, and more importantly, emergent abilities that are not present in smaller-scale language models. As illustrated in Fig. 1, these emergent abilities include (1) in-context learning, where LLMs learn a new task from a small set of examples presented in the prompt at inference time, (2) instruction following, where LLMs, after instruction tuning, can follow the instructions for new types of tasks without using explicit examples, and (3) multi-step reasoning, where LLMs can solve a complex task by breaking down that task into intermediate reasoning steps as demonstrated in the chain-of-thought prompt [34]. LLMs can also be augmented by using external knowledge and tools [35], [36] so that they can effectively interact with users and environment [37], and continually improve itself using feedback data collected through interactions (e.g. via reinforcement learning with human feedback (RLHF)).

Through advanced usage and augmentation techniques, LLMs can be deployed as so-called AI agents: artificial entities that sense their environment, make decisions, and take actions. Previous research has focused on developing agents for specific tasks and domains. The emergent abilities demonstrated by LLMs make it possible to build general-purpose AI agents based on LLMs. While LLMs are trained to produce responses in static settings, AI agents need to take actions to interact with dynamic environment. Therefore, LLM-based agents often need to augment LLMs to e.g., obtain updated information from external knowledge bases, verify whether a system action produces the expected result, and cope with when things do not go as expected, etc. We will discuss in detail LLM-based agents in Section IV.

In the rest of this paper, Section II presents an overview of state of the art of LLMs, focusing on three LLM families (GPT, LLaMA and PaLM) and other representative models. Section III discusses how LLMs are built. Section IV discusses how LLMs are used, and augmented for real-world applications Sections V and VI review popular datasets and benchmarks for evaluating LLMs, and summarize the reported LLM evaluation results. Finally, Section VII concludes the paper by summarizing the challenges and future research directions.

II. Large Language Models

In this section we start with a review of early pre-trained neural language models as they are the base of LLMs, and then focus our discussion on three families of LLMs: GPT, LlaMA, and PaLM. Table I provides an overview of some of these models and their characteristics.

A. *Early Pre-trained Neural Language Models*

Language modeling using neural networks was pioneered by [38], [39], [40]. Bengio et al. [13] developed one of the first neural language models (NLMs) that are comparable to n-gram models. Then, [14] successfully applied NLMs to machine translation. The release of RNNLM (an open source NLM toolkit) by Mikolov [41], [42] helped significantly popularize NLMs. Afterwards, NLMs based on recurrent neural networks (RNNs) and their variants, such as long short-term memory (LSTM) [19] and gated recurrent unit (GRU) [20], were widely used for many natural language applications including machine translation, text generation and text classification [43].

Then, the invention of the Transformer architecture [44] marks another milestone in the development of NLMs. By applying self-attention to compute in parallel for every word in a sentence or document an "attention score" to model the influence each word has on another, Transformers allow for much more parallelization than RNNs, which makes it possible to efficiently pre-train very big language models on large amounts of data on GPUs. These pre-trained language models (PLMs) can be fine-tuned for many downstream tasks.

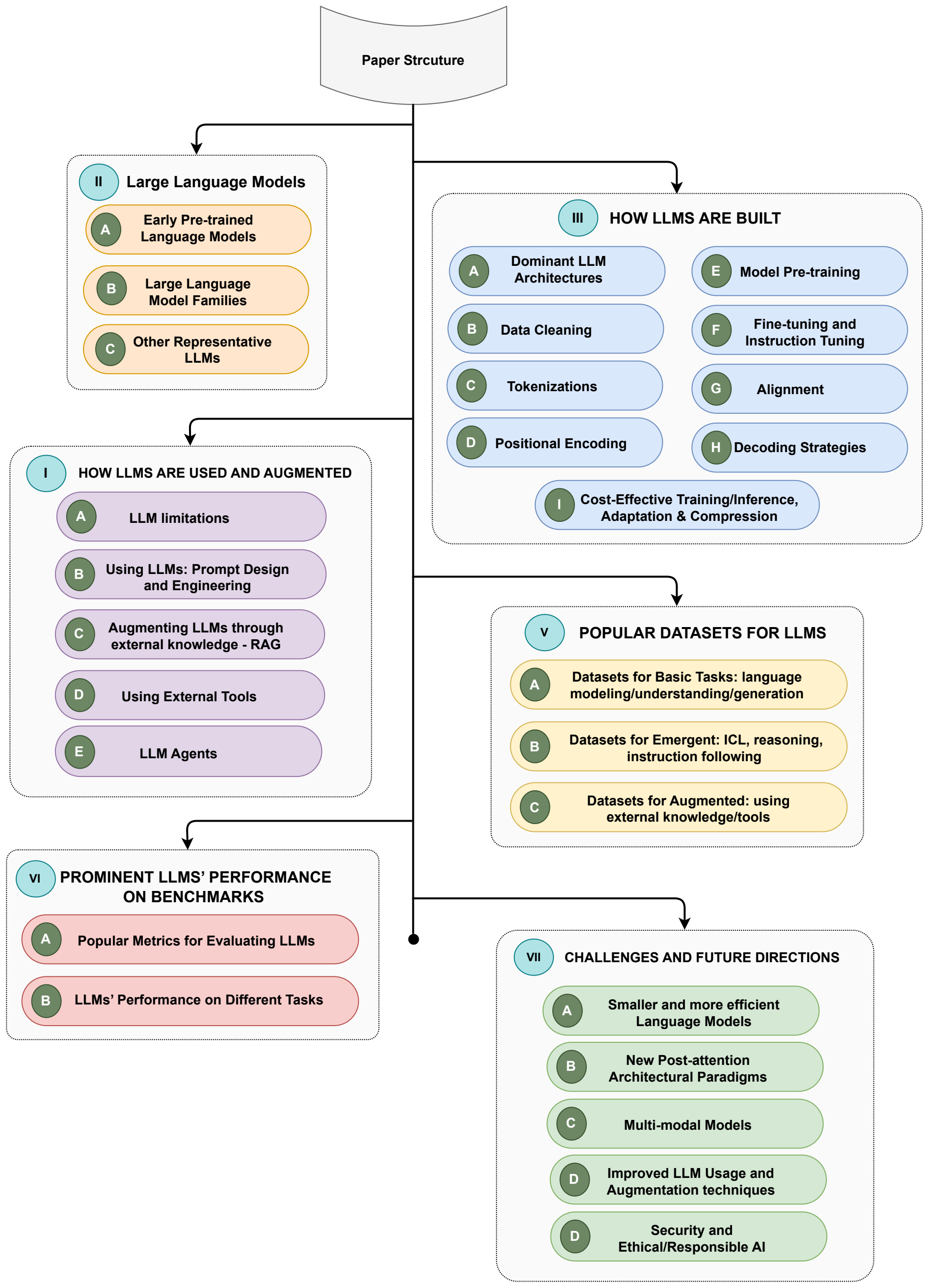
Paper Strcuture
II Large Language Models
A Early Pre-trained Language Models
B Large Language Model Families
C Other Representative LLMs
III HOW LLMS ARE BUILT
A Dominant LLM Architectures
B Data Cleaning
C Tokenizations
D Positional Encoding
E Model Pre-training
F Fine-tuning and Instruction Tuning
G Alignment
H Decoding Strategies
I Cost-Effective Training/Inference, Adaptation & Compression
I HOW LLMS ARE USED AND AUGMENTED
A LLM limitations
B Using LLMs: Prompt Design and Engineering
C Augmenting LLMs through external knowledge - RAG
D Using External Tools
E LLM Agents
V POPULAR DATASETS FOR LLMS
A Datasets for Basic Tasks: language modeling/understanding/generation
B Datasets for Emergent: ICL, reasoning, instruction following
C Datasets for Augmented: using external knowledge/tools
VI PROMINENT LLMS' PERFORMANCE ON BENCHMARKS
A Popular Metrics for Evaluating LLMs
B LLMs' Performance on Different Tasks
VII CHALLENGES AND FUTURE DIRECTIONS
A Smaller and more efficient Language Models
B New Post-attention Architectural Paradigms
C Multi-modal Models
D Improved LLM Usage and Augmentation techniques
D Security and Ethical/Responsible AI

Fig. 2: The paper structure.

We group early popular Transformer-based PLMs, based on their neural architectures, into three main categories: encoder-only, decoder-only, and encoder-decoder models. Comprehensive surveys of early PLMs are provided in [43], [28].

1) Encoder-only PLMs: As the name suggests, the encoder-only models only consist of an encoder network. These models are originally developed for language understanding tasks, such as text classification, where the models need to predict a class label for an input text. Representative encoder-only models include BERT and its variants, e.g., RoBERTa, ALBERT, DeBERTa, XLM, XLNet, UNILM, as to be described below.

BERT (Birectional Encoder Representations from Transformers) [24] is one of the most widely used encoder-only language models. BERT consists of three modules: (1) an embedding module that converts input text into a sequence of embedding vectors, (2) a stack of Transformer encoders that converts embedding vectors into contextual representation vectors, and (3) a fully connected layer that converts the representation vectors (at the final layer) to one-hot vectors. BERT is pre-trained uses two objectives: masked language modeling (MLM) and next sentence prediction. The pre-trained BERT model can be fine-tuned by adding a classifier layer for many language understanding tasks, ranging from text

TABLE I: High-level Overview of Popular Language Models

Type	Model Name	#Parameters	Release	Base Models	Open Source	#Tokens	Training dataset
Encoder-Only	BERT	110M, 340M	2018	-	✓	137B	BooksCorpus, English Wikipedia
	RoBERTa	355M	2019	-	✓	2.2T	BooksCorpus, English Wikipedia, CC-NEWS, STORIES (a subset of Common Crawl), Reddit
	ALBERT	12M, 18M, 60M, 235M	2019	-	✓	137B	BooksCorpus, English Wikipedia
	DeBERTa	-	2020	-	✓	-	BooksCorpus, English Wikipedia, STORIES, Reddit content
	XLNet	110M, 340M	2019	-	✓	32.89B	BooksCorpus, English Wikipedia, Giga5, Common Crawl, ClueWeb 2012-B
Decoder-only	GPT-1	120M	2018	-	✓	1.3B	BooksCorpus
	GPT-2	1.5B	2019	-	✓	10B	Reddit outbound
Encoder-Decoder	T5 (Base)	223M	2019	-	✓	156B	Common Crawl
	MT5 (Base)	300M	2020	-	✓	-	New Common Crawl-based dataset in 101 languages (m Common Crawl)
	BART (Base)	139M	2019	-	✓	-	Corrupting text
GPT Family	GPT-3	125M, 350M, 760M, 1.3B, 2.7B, 6.7B, 13B, 175B	2020		×	300B	Common Crawl (filtered), WebText2, Books1, Books2, Wikipedia
	CODEX	12B	2021	GPT	✓	-	Public GitHub software repositories
	WebGPT	760M, 13B, 175B	2021	GPT-3	×	-	ELI5
	GPT-4	1.76T	2023	-	×	13T	-
LLaMA Family	LLaMA1	7B, 13B, 33B, 65B	2023	-	✓	1T, 1.4T	Online sources
	LLaMA2	7B, 13B, 34B, 70B	2023	-	✓	2T	Online sources
	Alpaca	7B	2023	LLaMA1	✓	-	GPT-3.5
	Vicuna-13B	13B	2023	LLaMA1	✓	-	GPT-3.5
	Koala	13B	2023	LLaMA	✓	-	Dialogue data
	Mistral-7B	7.3B	2023		✓	-	-
	Code Llama	34	2023	LLaMA2	✓	500B	Publicly available code
	LongLLaMA	3B, 7B	2023	OpenLLaMA	✓	1T	-
	LLaMA-Pro-8B	8.3B	2024	LLaMA2-7B	✓	80B	Code and math corpora
	TinyLlama-1.1B	1.1B	2024	LLaMA1.1B	✓	3T	SlimPajama, Starcoderdata
PaLM Family	PaLM	8B, 62B, 540B	2022	-	×	780B	Web documents, books, Wikipedia, conversations, GitHub code
	U-PaLM	8B, 62B, 540B	2022	-	×	1.3B	Web documents, books, Wikipedia, conversations, GitHub code
	PaLM-2	340B	2023	-	✓	3.6T	Web documents, books, code, mathematics, conversational data
	Med-PaLM	540B	2022	PaLM	×	780B	HealthSearchQA, MedicationQA, LiveQA
	Med-PaLM 2	-	2023	PaLM 2	×	-	MedQA, MedMCQA, HealthSearchQA, LiveQA, MedicationQA
Other Popular LLMs	FLAN	137B	2021	LaMDA-PT	✓	-	Web documents, code, dialog data, Wikipedia
	Gopher	280B	2021	-	×	300B	MassiveText
	ERNIE 4.0	10B	2023	-	×	4TB	Chinese text
	Retro	7.5B	2021	-	×	600B	MassiveText
	LaMDA	137B	2022	-	×	168B	public dialog data and web documents
	ChinChilla	70B	2022	-	×	1.4T	MassiveText
	Galactia-120B	120B	2022	-		450B	
	CodeGen	16.1B	2022	-	✓	-	THE PILE, BIGQUERY, BIGPYTHON
	BLOOM	176B	2022	-	✓	366B	ROOTS
	Zephyr	7.24B	2023	Mistral-7B	✓	800B	Synthetic data
	Grok-0	33B	2023	-	×	-	Online source
	ORCA-2	13B	2023	LLaMA2	-	2001B	-
	StartCoder	15.5B	2023	-	✓	35B	GitHub
	MPT	7B	2023	-	✓	1T	RedPajama, m Common Crawl, S2ORC, Common Crawl
	Mixtral-8x7B	46.7B	2023	-	✓	-	Instruction dataset
	Falcon 180B	180B	2023	-	✓	3.5T	RefinedWeb
	Gemini	1.8B, 3.25B	2023		✓	-	Web documents, books, and code, image data, audio data, video data
	DeepSeek-Coder	1.3B, 6.7B, 33B	2024	-	✓	2T	GitHub's Markdown and StackExchange
	DocLLM	1B,7B	2024	-	×	2T	IIT-CDIP Test Collection 1.0, DocBank

classification, question answering to language inference. A high-level overview of BERT framework is shown in Fig 3. As BERT significantly improved state of the art on a wide range of language understanding tasks when it was published, the AI community was inspired to develop many similar encoder-only language models based on BERT.

RoBERTa [25] significantly improves the robustness of BERT using a set of model design choices and training strategies, such as modifying a few key hyperparameters, removing the next-sentence pre-training objective and training with much larger mini-batches and learning rates. ALBERT [45] uses two parameter-reduction techniques to lower memory consumption and increase the training speed of BERT: (1) splitting the embedding matrix into two smaller matrices, and (2) using repeating layers split among groups. DeBERTa (Decoding-enhanced BERT with disentangled attention) [26] improves the BERT and RoBERTa models using two novel techniques. The first is the disentangled attention mechanism, where each word is represented using two vectors that encode its content and position, respectively, and the attention weights among words