Published: 24 November 2025
Online AM: 18 September 2025
Accepted: 23 August 2025
Revised: 18 August 2025
Received: 02 January 2025

SURVEY

A Comprehensive Survey of Small Language Models in the Era of Large Language Models: Techniques, Enhancements, Applications, Collaboration with LLMs, and Trustworthiness

FALI WANG, Pennsylvania State University, University Park, PA, United States

ZHIWEI ZHANG, Pennsylvania State University, University Park, PA, United States

XIANREN ZHANG, Pennsylvania State University, University Park, PA, United States

ZONGYU WU, Pennsylvania State University, University Park, PA, United States

TZU HAO MO, University of Pennsylvania, Philadelphia, PA, United States

QIUHAO LU, University of Texas Health Science Center at Houston, Houston, TX, United States

View all

Open Access Support provided by:

Amazon.com, Inc.

University of Texas Health Science Center at Houston

Rensselaer Polytechnic Institute

Pennsylvania State University

University of Pennsylvania

A Comprehensive Survey of Small Language Models in the Era of Large Language Models: Techniques, Enhancements, Applications, Collaboration with LLMs, and Trustworthiness

FALI WANG, ZHIWEI ZHANG, XIANREN ZHANG, and ZONGYU WU, The Pennsylvania State University, University Park, PA, USA
TZUHAO MO, University of Pennsylvania, Philadelphia, PA, USA
QIUHAO LU, WANJING WANG, and RUI LI, The University of Texas Health Science Center at Houston, Houston, TX, USA
JUNJIE XU, The Pennsylvania State University, University Park, PA, USA
XIANFENG TANG and QI HE, Amazon.com Inc., Palo Alto, CA, USA
YAO MA, Rensselaer Polytechnic Institute, Troy, NY, USA
MING HUANG, The University of Texas Health Science Center at Houston, Houston, TX, USA
SUHANG WANG, The Pennsylvania State University, University Park, PA, USA

Large language models (LLMs) have demonstrated emergent abilities in text generation, question answering, and reasoning, facilitating various tasks and domains. Despite their proficiency in various tasks, LLMs like PaLM 540B and Llama-3.1 405B face limitations due to large parameter sizes and computational demands, often requiring cloud API use, which raises privacy concerns, limits real-time applications on edge devices, and increases fine-tuning costs. Additionally, LLMs often underperform in specialized domains such as healthcare and law due to insufficient domain-specific knowledge, necessitating specialized models. Therefore, Small Language Models (SLMs) are increasingly favored for their low inference latency, cost-effectiveness, efficient development, and easy customization and adaptability. These models are particularly well-suited for resource-limited environments and domain knowledge acquisition, addressing LLMs' challenges and proving ideal for applications that require localized data handling for privacy, minimal inference latency for efficiency, and domain knowledge acquisition through lightweight fine-tuning. The rising demand for SLMs has spurred

This material is based upon work supported by, or in part by, the Army Research Office under grant number W911NF21-1-0198, the Department of Homeland Security CINA under grant number 17STCIN0000105-00, Cisco Faculty Research Award. The findings and conclusions in this article do not necessarily reflect the views of the funding agencies.
Authors' Contact Information: Fali Wang, The Pennsylvania State University, University Park, PA, USA; e-mail: fqw5095@psu.edu; Zhiwei Zhang, The Pennsylvania State University, University Park, PA, USA; e-mail: zbz5349@psu.edu; Xianren Zhang, The Pennsylvania State University, University Park, PA, USA; e-mail: xzz5508@psu.edu; Zongyu Wu, The Pennsylvania State University, University Park, PA, USA; e-mail: zzw5373@psu.edu; TzuHao Mo, University of Pennsylvania, Philadelphia, PA, USA; e-mail: investdmo@gmail.com; Qiuhao Lu, e-mail: qiuhao.lu@uth.tmc.edu; Wanjing Wang, The University of Texas Health Science Center at Houston, Houston, TX, USA; e-mail: wanjing.wang@uth.tmc.edu; Rui Li, The University of Texas Health Science Center at Houston, Houston, TX, USA; e-mail: rui.li.1@uth.tmc.edu; Junjie Xu, The Pennsylvania State University, University Park, PA, USA; e-mail: jmx5097@psu.edu; Xianfeng Tang; Amazon.com Inc., Palo Alto, CA, USA; e-mail: tangxianfeng@outlook.com; Qi He, Amazon.com Inc., Palo Alto, CA, USA; e-mail: qih@amazon.com; Yao Ma, Rensselaer Polytechnic Institute, Troy, NY, USA; e-mail: may13@rpi.edu; Ming Huang, The University of Texas Health Science Center at Houston, Houston, USA; e-mail: ming.huang@uth.tmc.edu; Suhang Wang (corresponding author), The Pennsylvania State University, University Park, PA, USA; e-mail: szw494@psu.edu.

ACM 2157-6912/2025/11-ART145
https://doi.org/10.1145/3768165

extensive research and development. However, a comprehensive survey investigating issues related to the definition, acquisition, application, enhancement, and reliability of SLM remains lacking, prompting us to conduct a detailed survey on these topics. The definition of SLMs varies widely; thus, to standardize, we propose defining SLMs by their capability to perform specialized tasks and suitability for resource-constrained settings, setting boundaries based on the minimal size for emergent abilities and the maximum size sustainable under resource constraints. For other aspects, we provide a taxonomy of relevant models/methods and develop general frameworks for each category to enhance and utilize SLMs effectively. We have compiled the collected SLM models and related methods on GitHub: https://github.com/FairyFali/SLMs-Survey.

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

Additional Key Words and Phrases: Small Language Models, On-Device LLMs, Domain-specific Models, Trustworthiness

ACM Reference format:
Fali Wang, Zhiwei Zhang, Xianren Zhang, Zongyu Wu, TzuHao Mo, Qiuhao Lu, Wanjing Wang, Rui Li, Junjie Xu, Xianfeng Tang, Qi He, Yao Ma, Ming Huang, and Suhang Wang. 2025. A Comprehensive Survey of Small Language Models in the Era of Large Language Models: Techniques, Enhancements, Applications, Collaboration with LLMs, and Trustworthiness. *ACM Trans. Intell. Syst. Technol.* 16, 6, Article 145 (November 2025), 87 pages.
https://doi.org/10.1145/3768165

1 Introduction

The evolution of neural **language models (LMs)** from BERT's [86] pre-training and fine-tuning paradigm to T5's [291] pre-training plus prompting approach and finally to GPT-3's [38] pre-training plus in-context learning has greatly enhanced **natural language processing (NLP)**. These advancements have broadened NLP's application across various fields, including language understanding [356], programming [262, 337], recommendation systems [375], information retrieval [45, 155, 232, 323], mobile-device control [90], scientific discovery [318, 441], medical question answering [35, 372], and legal question answering [12]. In particular, the recent emergence of proprietary commercial models, including ChatGPT, Bard, and Claude, and open-sourced models such as Llama [96, 344, 345] has led to rapid growth in the development of **large language models (LLMs)**. Even though neural networks consistently improve on various tasks with longer training times, larger datasets, and increased model sizes—a phenomenon known as a neural scaling law [169], these models unpredictably exhibit a sudden acquisition of versatile abilities, termed "*emergent ability*," once they reach a critical scale threshold, thereby supporting the "larger is better" trend. This ability is not present in small-scale models. For instance, the latest Llama-3.1 model with 405 billion parameters performs better in dialogue, logical reasoning, and programming compared to the smaller 7B counterpart [96].

Despite their prowess in complex tasks, LLMs' huge parameters and computational needs impose significant limitations, hindering their adoption in many real-world applications. For example, the LLaMa 3.1 model with 405 billion parameters [96], trained on 16 K H100 GPUs for 54 days, requires about 202.5 GB of GPU memory using int4 precision and has large inference latency. These issues present several challenges in specific contexts: (1) LLMs are generally hosted in the cloud and used via cloud-based APIs due to the large GPU memory and computational cost. Users need to upload their data to query LLMs, raising data leakage and privacy concerns, especially in high-stake scenarios such as healthcare, finance, and e-commerce; (2) Driven by personal agents, on-device deployment is a critical requirement. Several factors, including cloud costs, latency, and privacy concerns, hinder the on-device processing of cloud-based LLMs, and direct deployment is impractical due to their high parameter and cache requirements, which often exceed the capabilities

- Small Language Models
  - Introduction (§1)
  - Concepts in Building LMs (§2)
    - Architecture (§2.1)
    - Training techniques (§2.2)
    - Obtain SLMs from LLMs (§2.3)
      - Pruning (§2.3.1)
        - Unstructured Pruning [79, 104, 205, 307, 327, 443, 447]; Structured Pruning [14, 20, 53, 126, 191, 200, 237, 248, 390, 415, 450]
      - Knowledge Distillation (§2.3.2)
        - White-Box KD [6, 121, 160, 172, 176, 272, 439]; Black-Box KD [50, 278, 366]
      - Quantization (§2.3.3)
        - SqueezeLLM [173]; JSQ [124]; FrameQuant [4]; OneBit [405]; BiLLM [150]; LQER [437]; I-LLM [147]; PV-Tuning [244]; BitNet [363]; BitNet b1.58 [236]; PEQA [171]; QLoRA [84]
  - Enhancement of SLMs (§3)
    - Training Methods for SLMs from Scratch (§3.1)
      - MobiLlama [342]; MobileLLMs [227]; MindLLMs [417]; Tang et al. [334]
    - Supervised Fine-Tuning (§3.2)
      - Alpaca [335]; UltraChat [89]; WizardLM [397]; SlimOrca [207]; ShareGPT [362]; Capybara [75]; Deita [221]; MetaMathQA [425]; MobileBERT [328]; StableLM [27, 346]; RLHF [271]; DPO [290]
    - Data quality in Knowledge Distillation (§3.3)
      - TinyStory [98]; Self-Amplify [30]; AS-ES Learning [389]; Huang et al. [148]; Bhan et al. [30]; Tian et al. [343]
    - Distillation techniques for enhancing SLM (§3.4)
      - GKD [6]; DistiLLM [176]; Adapt-and-Distill [419]; Bit-level Inference Scaling Laws [85]
    - Performance improvement through quantization (§3.5)
      - Bit-level Inference Scaling Laws [85]; BiLLM [150]; LLM.int8() [83]; PB-LLM [306]; OneBit [405]; BitNet [363]; LLM-QAT [226]
    - Techniques in LLMs contributing SLMs (§3.6)
      - RAG for SLMs [220, 420]; MoE for SLMs [175, 197]
  - Applications of SLMs (§4)
    - Task-specific SLM applications (§4.1)
      - SLM applications in QA (§4.1.1)
        - Phi-series [1, 122]; Orca 2 [253]; BioGPT-Large [127]; Stable Beluga 7B [243]; Phogat et al. [282]; Hartill et al. [131]; Gichamba et al. [114]; Jeong et al. [159]; Rationale Ranking [130]
      - SLM applications in Coding (§4.1.2)
        - DeepSeek-Coder [123]; Phi-1 [122]; Phi-3.5-mini [1]; CodeGemma [337]; CodeLlama [299]
      - SLM applications in Recommender Systems (§4.1.3)
        - PromptRec [387]; SLIM [375]; BiLLP [316]; RecLoRA [459]; Wu et al. [384]
      - SLM applications in Web Search (§4.1.4)
        - Content encoder [45, 155, 232]; H-ERNIE [65]; Zou et al. [466]; Peng et al. [279]; CoCondenser [110]; Implicit Interaction (I^3) [92]; InPars [37]; rewrite-retrieve-read [238]
      - SLM applications in Mobile-device (§4.1.5)
        - Octopus [54]; MobileAgent [90]; Carreira et al. [43]; AutoDroid [381]; Qin et al. [285]; Zhu et al. [462]
    - Techniques during SLM Deployment (§4.2)
      - Memory efficiency (§4.2.1)
        - MoE-I[2] [409]; MobileAIBench [260]; MobileLLM [227]; EdgeMoE [421]; GEAR [168]; HETLORA [64]; MobiLlama [342]
      - Computing efficiency (§4.2.2)
        - COST-EFF [312]; MobileLLM [227]; MobiLlama [342]; Merino [455]; EdgeMoE [421]; Nawrot et al. [263]; LLMCadLLM-Cad[399]; LinguaLinked [451]
  - Models (§5)
    - Generic-domain SLMs (§5.1)
      - PhoneLM [422]; Llama 3.2; Qwen [24, 407]; Gemma [338, 339]; StableLM [27, 346]; TinyLlama [444]; OLMo [118]; H2O-Danube3 [281]; Fox-1 [340]; MiniCPM [146]; Phi [1, 1, 122, 158, 204]; BLOOM and BLOOMZ [184]; Galactica [336]; OPT [445]; XGLM [214]; Megatron-gpt2 [317]; MINITRON [259]; LaMini-LM [174]; FlanT5 [66] ...
    - Specific-domain SLMs (§5.2)
      - Hippocrates [3]; BioMedLM [35]; MentaLLaMA [411]; ChemLLM [441]; SciGLM [440]; AstroLLaMA [265]; MindLLM [417]
  - SLMs for LLMs (§6)
    - SLM for reliable LLM generation (§6.1)
      - POLAR [454]; SAPLMA [22]; SuperICL [398]; SuperContext [413]; Self-RAG [19]; SKR [371]; SlimPLM [331]; CRAG [406]; Feng et al. [103]; Liu et al. [224]; HaluAgent [60]; Toolformer [302]
    - SLM for extracting LLM prompts (§6.2)
      - Prompt Stealing Attacks [304]; Output2prompt [438]; Model Purifying [201]; [448]
    - SLM for fine-tuning LLMs (§6.3)
      - Weak-to-Strong Search [457]; Emulated Fine-tuning [252]; CROSSLM [82]; Swayamdipta et al. [329]; Mekala et al. [247]; Proxy-tuning [217]
    - SLM for LLM applications (§6.4)
      - SLCoLM [333]; HEF [418]; BLADE [192]; Contrastive Decoding [202]; “Train-Guide-Predict” [333]; Sennrich et al. [303]
    - SLM for LLM safety (§6.5)
      - Llama Guard [156]; Kwon et al. [180]
    - SLM for LLM evaluation (§6.6)
      - SLIDE [453]; Semantic uncertainty [178]; Selfcheckgpt [245]; Proxylm [17]; Factscore [154]
  - Synergy between SLMs and LLMs (§7)
    - Cloud-edge synergy (§7.1)
      - CoGenesis [442]; Hao et al. [129]; Xu et al. [402]; LLM-to-SLM [29]; Synergy of Thoughts [305]; CROSSLM [82]
    - Task-centric synergy (§7.2)
      - α-UMi [314]; SynCID [208]; Filter-then-rerank Framework [239]; LLMCad [399]
  - Trustworthiness in SLMs (§8)
    - HELM [209]; Do-Not-Answer [370]; PromptRobust [460]; GLUE-X [412]; HaluEval [194]; PrivLM-Bench [193]; FFT [74]; ROBBIE [100]; TrustLLM [326]; RAmBLA [36]; JailbreakBench [46]; Xie et al. [395]; OR-Bench [73]; SORRY-Bench [392]; BeHonest [61]; Hong et al. [141]; Nakka et al. [261]; RUPBench [376]
  - Future Directions (§9)

Fig. 1. Overview of small language models.

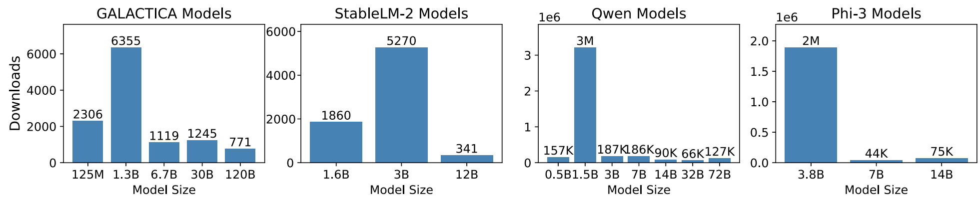


Fig. 2. Download statistics last month in Hugging face for LLMs with various model sizes, obtained on October 7, 2024.

of devices such as mobile phones; (3) Their large parameter count can cause inference delays from seconds to minutes, unsuitable for real-time applications. For instance, Llama 2 7B takes approximately 84 seconds to process 100 tokens on benchmarks including HellaSwag, TruthfulQA, MMLU, and Arc_C when run on a smartphone equipped with a Snapdragon 685 processor [342]; (4) To boost performance in specialized domains such as healthcare and law, where generic LLMs underperform, LLMs are often fine-tuned. However, this process is computationally expensive due to their large size. (5) Though general-purpose LLMs are powerful, many real-world applications require only specific abilities and domain knowledge; deploying general-purpose LLMs would be a waste of resources, and such LLMs often cannot match the performance of models tailored for specific tasks [54, 127, 159, 282, 375].

Recently, **small language models (SLMs)** have shown great potential in alleviating these issues while achieving performance comparable to LLMs for domain-specific problems [1, 27, 118, 146, 227, 281, 339, 342, 359, 407, 444]. Owing to fewer parameters, SLMs excel in efficiency, cost, flexibility, and customization. They provide significant computational savings in pre-training and inference with reduced memory and storage needs, which is vital for applications requiring efficient resource use. These small models are especially effective in resource-limited settings, performing well on low-power devices such as edge devices. Besides, SLMs improve on-device processing by enhancing privacy, security, response times, and personalization. This supports advanced personal assistants and cloud-independent applications, boosting energy efficiency and reducing carbon emissions. For example, the Llama 3.2 models (1B and 3B) demonstrate that local processing enables immediate execution of prompts and responses [7]. This approach protects privacy by keeping sensitive data such as **patient health information (PHI)**, business data, personal messages, and calendar details local, enhancing confidentiality. It also allows for precise control over which queries are processed on-device versus those requiring cloud-based models. Therefore, SLMs are gaining increasing attention as alternatives to LLMs, as indicated in Figure 2, which shows that SLMs are downloaded more frequently than larger models in the Hugging Face community, and Figure 3, which illustrates the growing popularity of SLM releases over time.

Typically, LMs that exhibit emergent abilities are classified as LLMs. However, the categorization of SLMs remains unclear. Studies vary in their contexts: some define SLMs as models with fewer than one billion parameters [227], while others consider the term "small language model" relative to the larger counterparts [186, 333, 375], with no consensus on a unified definition in the current landscape of LLMs. Research suggests SLMs for mobile devices, typically possessing around 6 GB of memory, consist of sub-billion parameter models [227], whereas others classify models with up to 10 billion parameters as small, noting their lack of emergent abilities [107]. Given their use in resource-constrained environments and for specific tasks, we propose a generalized definition: *Given specific tasks and resource constraints, we define SLMs as falling within a range where the lower bound is the minimum size at which the model exhibits emergent abilities for a specialized task, and the upper bound is the largest size manageable within limited resource conditions.* This definition

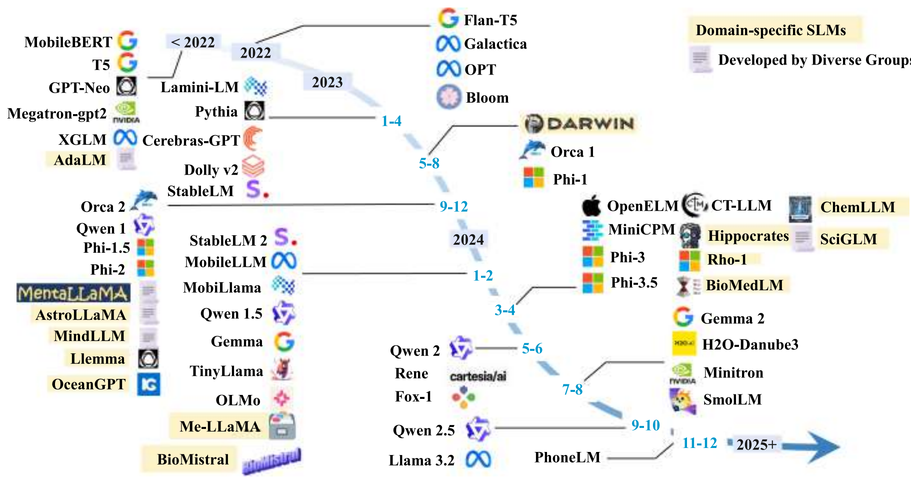


Fig. 3. A timeline of existing SLMs.

integrates various perspectives and addresses factors related to mobile computing and capability thresholds.

Due to the growing demand for SLMs, extensive literature has emerged on various aspects of SLMs. For example, several resource-efficient techniques [403] and training methods optimized for SLMs, such as **Quantization-Aware Training (QAT)** [226, 363, 405] and selective architectural component choices [227, 287, 342], aim to enhance performance in specific applications [9, 37, 54, 282, 299, 387]. These methods have led to the development of numerous open-source, general-purpose, and domain-specific SLMs [3, 27, 35, 339, 407, 440]. Beyond their inherent capabilities, SLMs can enhance LLMs by serving as modules or effective proxies [252, 304, 388, 404, 418, 454]. Furthermore, the complementary advantages of SLMs and LLMs can be leveraged collectively to better complete tasks [82, 208, 239, 314, 360, 442]. Despite the commendable performance of SLMs, it is crucial not to overlook their credibility issues, such as the risks of adversarial attacks, producing hallucinations, and privacy breaches [91, 97, 141, 179, 261, 280, 254, 357, 376, 430]. However, currently, there is no comprehensive survey thoroughly exploring these works on SLMs in the era of LLMs. Therefore, this article presents the first comprehensive survey analyzing various aspects of SLMs in the LLM era and their future directions. The overview structure of our article is shown in Figure 1. To summarize, our major contributions are:

— In Section 3, we examine various techniques for improving the performance of SLMs, including training from scratch, fine-tuning, **knowledge distillation (KD)**, quantization, and leveraging LLM-enhancing technologies to optimize SLMs.
— In Section 4, we discuss the tasks that SLMs can enhance and the deployment strategies that enable models to fit within the resource constraints of edge devices while maintaining acceptable inference speed.
— In Section 5, we collect SLMs with fewer than 7 billion parameters across both general-purpose and domain-specific applications, reviewing common architectural choices, training techniques, and datasets, and providing a comparative summary of performance across different model sizes. Recent SLMs are listed.
— In Section 6, we explore how SLMs can address key challenges faced by LLMs, such as high inference latency, labor-intensive fine-tuning, susceptibility to knowledge noise, and risks of copyright infringement.

— In Section 7, we survey two kinds of synergies between LLMs and SLMs: one involves cloud-based LLMs and local SLMs, while the other leverages the unique advantages of both to more effectively solve tasks.
— In Section 8, we investigate the trustworthiness issues of SLMs, including hallucination and privacy concerns, by providing a taxonomic summary of current evaluation methods.

Concurrently with our survey, Lu et al. [233] evaluate open-source SLMs, focusing on their architectures, datasets, algorithms, and on-device performance metrics such as inference latency and memory usage. Van Nguyen et al. [351] delve into optimization strategies for SLMs, including model compression, pruning, and quantization. Chen and Varoquaux [51] investigate how SLMs enhance LLMs and vice versa. In contrast, our survey offers a more comprehensive review with the following differences: (1) we present a detailed taxonomy of recent advancements in SLMs in the era of LLMs; (2) we define SLMs based on emergent capabilities and device specifications, which refines previous unclear definitions related to LLMs; (3) we discuss SLM applications, especially in on-device tasks and deployment, topics previously unexplored; (4) we examine domain-specific SLMs previously overlooked; and (5) we additionally consider the synergy between SLMs and LLMs.

2 Foundational Concepts in Building LMs

This section will introduce foundational concepts and background knowledge for LMs, including the concepts of architecture and the training process, as well as methods for obtaining SLMs from LLMs. The advanced training strategy to improve SLM performance will be introduced in Section 3.

2.1 Architecture of SLMs

SLMs commonly employ the Transformer architecture [352] (see Figure 4), which utilizes self-attention mechanisms to manage long-range text dependencies, essential for maintaining performance with constrained resources. However, due to the attention mechanism, Transformers have a large inference cost. Hence, to alleviate the issue, several subquadratic-time architectures such as Mamba [119], Hymba [93], and xLSTM [26] are proposed. Next, we will give details of Transformer due to its popularity and briefly introduce newly emerged models.

2.1.1 Transformer. The Transformer's self-attention mechanism [352] allows LMs to efficiently capture contextual information across longer sequences, even with limited resources. The Transformer generally adopts an encoder–decoder structure featuring self-attention mechanisms, FFNs positional embeddings, and layer normalization. The Transformer architecture design tailored for SLMs is detailed in Section 5; this section will provide only foundational concepts.

The Self-Attention Mechanism enables the model to evaluate the importance of tokens relative to each other. The self-attention mechanism is written as

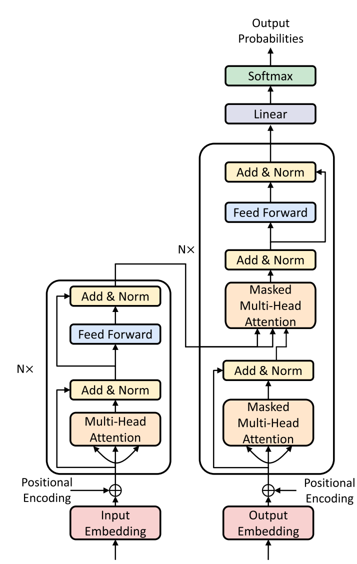


Fig. 4. Transformer architecture [352].

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V},$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are query, key, and value matrices, scaled by $\sqrt{d_k}$ for stability where d_k is the dimension of key matrices. The dot product $\mathbf{Q}\mathbf{K}^\top$ reflects the similarity between the query and key vectors.

Multi-Head Attention (MHA) [352] is the first method that uses multiple heads to capture diverse information. MHA allows the model to attend to different parts of the input sequence using multiple attention heads as

$$\begin{aligned}\text{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) &= \text{Concat}(\text{head}_1,\text{head}_2,\ldots,\text{head}_h)\mathbf{W}^O, \text{ with } \text{head}_i \\ &= \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q,\mathbf{K}\mathbf{W}_i^K,\mathbf{V}\mathbf{W}_i^V),\end{aligned} \tag{1}$$

Each head in the MHA mechanism operates independently, allowing the model to capture diverse aspects of the data. The outputs are combined using learned projection matrices $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, and $\mathbf{W}_i^V$, concatenated, and passed through the output projection matrix $\mathbf{W}^O$.

Building on this foundation, several modifications have been introduced to further optimize self-attention mechanisms for specific challenges such as memory efficiency and computational speed. To address the KV-cache bottleneck in MHA, **Multi-Query Attention (MQA)** [310] proposes that all attention heads share the same set of keys and values, which reduces the memory and computational overhead associated with storing and managing multiple key-value pairs. **Grouped Query Attention (GQA)** [8] serves as a middle ground between MHA and MQA. It introduces subgroups of query heads (fewer than the total number of attention heads), where each subgroup shares a single key and value head. Unlike MQA and GQA, which reduce the number of key and value heads, **Multi-Head Latent Attention (MLA)** [216] compresses the keys and values into a joint latent vector. This compression allows for efficient handling of key-value pairs while maintaining high performance, significantly reducing the KV-cache and improving inference efficiency. *Flash Attention* [76, 77] accelerates the self-attention mechanism by minimizing the memory overhead typical of standard attention calculations. This optimization allows SLMs to process longer sequences more efficiently, enhancing their functionality under strict hardware constraints.

Feedforward Network (FFN) comprises two linear transformations separated by a non-linearity, typically modeled as $\text{FFN}(\mathbf{x}) = \sigma(\mathbf{x}\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2$, where $\mathbf{W}_1$ and $\mathbf{W}_2$ are the weight matrices, and b_1 and b_2 are bias terms. σ is the activation function, which introduces non-linearity, allowing models to learn complex patterns. Generally, ReLU is used as the activation function. In addition to ReLU, activation functions such as GeLU and SiLU are also used in SLMs to improve performance. We give the details here: (i) **ReLU (Rectified Linear Unit)** [5] is defined as $\sigma(x) = \max(0, x)$, which is commonly used for its simplicity and effectiveness. (ii) **Gaussian Error Linear Unit (GELU)** [138] is defined as $\text{GELU}(x) = x \cdot \Phi(x) = x \cdot \frac{1}{2}\left[1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right]$, where $\Phi(x)$ is the standard Gaussian CDF and *erf* is the error function. It is smoother than ReLU and widely used in models such as BERT [86] and GPT [289] for better gradient flow control. Since calculating the Gaussian error function for each neuron is computationally expensive and time-consuming, there are approximations using tanh and sigmoid functions, corresponding to $\text{GELU}_{\text{tanh}}$ and SiLU: (iii) *GELU with tanh* is defined as $\text{GELU}_{\text{tanh}}(x) = 0.5 \cdot x \cdot \left(1 + \tanh\left(\sqrt{\frac{2}{\pi}} \cdot (x + 0.044715 \cdot x^3)\right)\right)$. This approximation uses the Tanh function to simplify computations. (iv) **Sigmoid Linear Unit (SiLU)** [99] is calculated as $\text{SiLU}(x) = x \cdot \text{sigmoid}(x) = x \cdot \frac{1}{1+e^{-x}}$. It effectively combines the sigmoid function with its input, enhancing modeling capabilities. (v) **Swish-Gated Linear Units (SwiGLU)** [311] integrates the Swish activation function with Gated Linear Units, defined as $\text{SwiGLU}(x) = \text{Swish}(x \cdot W + b) \odot (x \cdot V + c)$, where W, V are the weight matrix and b, c are the bias terms. The Swish function is expressed as

Swish$(x) = x \cdot \text{sigmoid}(x)$. This combination enhances expressiveness and computational efficiency, making it a preferred choice in advanced models such as the Qwen series [407].

Positional Embeddings in Transformer models [352] are essential for capturing token order, providing context about relative positions within a sequence. Traditional positional embeddings in the Transformer architecture utilize a sinusoidal function, defined as: $PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$, $PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$, where pos represents the position within the sequence, i is the dimension index, and d_{model} is the dimensionality of the model. To improve the model's capacity for understanding the relative positions of tokens within a sequence, **Rotary Positional Embedding (RoPE)** [324] introduces a rotational matrix to the embeddings. RoPE significantly enhances the positional encoding by maintaining the relative distances through rotational transformations, thus optimizing the model's interpretative ability regarding sequence dynamics.

Layer Normalization [188] stabilizes the training process by normalizing layer outputs, accelerating convergence. Two types of layer normalization are commonly used [188]: (i) *Non-Parametric Layer Norm* normalizes inputs using the mean and variance calculated across the layer's dimensions without learnable parameters as $\text{LN}(x) = \frac{x-\mu}{\sigma}$, where μ is the mean and σ is the standard deviation of the inputs. Its simplicity makes it ideal for SLMs. (ii) *Parametric Layer Norm* includes learnable parameters γ and β for adaptive scaling and bias, enhancing model flexibility: $\text{PLN}(x) = \gamma\left(\frac{x-\mu}{\sigma}\right) + \beta$ Additionally, *RMSNorm (Root Mean Square Layer Normalization)* [436] simplifies the calculation by using the root mean square of inputs, reducing computational demands: $\text{RMSNorm}(x) = \gamma \frac{x}{\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2 + \epsilon}} + \beta$, where N is the number of inputs, x_i is the ith input, and ϵ is a small constant to prevent division by zero.

2.1.2 Mamba. The attention mechanism in Transformer suffers from a drawback: it requires recalculating attention scores with every previous token for each new token generated during inference, leading to quadratic time complexity. This increases the inference cost as sequence lengths grow. In contrast, Mamba [78, 119], based on **state space models (SSMs)** [167], which are a superclass of recurrent neural networks, relies only on the last hidden state for generating the next token, enabling faster inference speeds, as shown in Figure 5. To address the Linear Time Invariant nature of traditional SSMs, which hinders their ability to focus on or ignore specific inputs, Mamba improves SSMs with a dynamic selection mechanism. This mechanism selectively filters out irrelevant information while retaining essential data, tailored to the content of the input. Leveraging this selective SSM foundation, Mamba adeptly captures complex global relationships within sequence data. Due to its focus on the immediate previous hidden state, as opposed to Transformer, which requires access to all previous hidden states, Mamba achieves a higher utilization rate of model parameters. This makes it more suitable for SLMs. However, we identify two drawbacks of Mamba: (i) its focus on selectively capturing global information may compromise performance on tasks that require nuanced understanding, such as detailed sentiment analysis or complex entity recognition and (ii) to balance inference speed, Mamba's recurrent structure primarily encodes static global information, which limits its effectiveness in

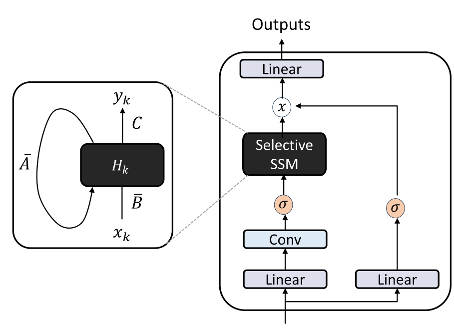


Fig. 5. Mamba 1 architecture [119].

handling multi-round tasks within a single query, such as interactive dialogue systems or iterative problem-solving scenarios.

In language modeling, Mamba 1 [119] is pre-trained on the Pile dataset [109] using the training recipe from [38] and ranges from 125M to 1.3 B parameters. It outperforms comparable models such as Pythia [32] and RWKV [277] in various tasks; for instance, Mamba-1.4B achieves a 32.8% accuracy on the Arc-Challenge [68] dataset, surpassing Pythia-1.4B's 28.5% and RWKV-1.5B's 29.4%. Mamba 2 [78] develops a theoretical framework linking SSMs with attention mechanisms through structured semi-separable matrices, enhancing the selective SSM to achieve 2–8× faster speeds while competing with Transformer models. Training and configuration for Mamba 2 align with Mamba 1. Additionally, Mamba-series models are applied widely across different fields [163, 287, 288, 467]. Other follow-up Mamba-based LMs, such as Falcon Mamba 7B [467] and Jamba [210], also demonstrate the efficiency and scalability of the Mamba architecture for NLP tasks. Falcon Mamba 7B scales Mamba's long-sequence processing capabilities to large-scale language data, reducing memory overhead and excelling in long-form generation. Jamba further extends this by blending Transformer and Mamba layers (with **Mixture-of-Experts (MoEs)**), achieving both high throughput and a compact memory footprint even at a massive scale.

2.1.3 Hymba. Attention heads in the Transformer facilitate high-resolution recall, while SSM heads in Mamba efficiently summarize context. To balance performance and efficiency for SLMs, Hymba [93] integrates both attention and SSM heads within the same layer, allowing for parallel and complementary processing of inputs, as depicted in Figure 6. This hybrid-head approach enables each layer to simultaneously leverage the high-resolution recall of attention heads and the contextual summarization of SSMs, increasing the model's flexibility and expressiveness in managing diverse information flows and memory access patterns.

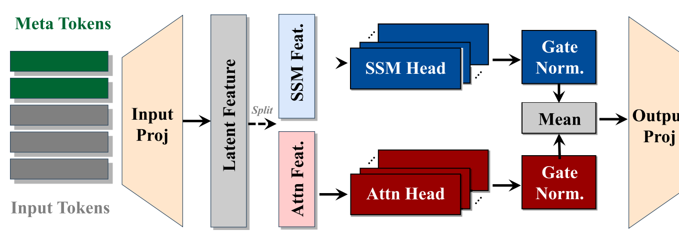


Fig. 6. Hymba [93] architecture.

Hymba has been developed in models of varying sizes—125M, 350M, and 1.5B, trained on a combination of the DCLM-Baseline-1.0 [195], SmolLM-Corpus [28], and a proprietary high-quality dataset, with token counts of 1 trillion, 250 billion, and 50 billion, respectively. The models incorporate the **Warmup-Stable-Decay (WSD)** learning rate scheduler [146] and the data annealing technique [96] to ensure stable pretraining, conducted on 128 NVIDIA A100 GPUs. The 1.5B base model was post-trained using **full fine-tuning (FFT)**, followed by **direct preference optimization (DPO)** [290] to develop the Hymba-1.5B-Instruct model. In commonsense reasoning tasks, the Hymba 1.5B model surpasses Llama-3.2-3B [7] by achieving 1.32% higher average accuracy, requiring an 11.67× smaller cache size, and delivering a 3.49× increase in processing speed.

2.1.4 xLSTM. **Long Short-Term Memory (LSTM)** [140] shares a conceptual similarity with Mamba that achieves success in language modeling through the introduction of time-dependent weights. This similarity raises an intriguing question: how effective would LSTMs be at language modeling if scaled to billions of parameters, incorporating advanced techniques from modern LLMs while addressing known LSTM limitations? Inspired by this question, Beck et al. [26] propose the xLSTM architecture, which performs favorably compared to state-of-the-art Transformers and SSMs in empirical evaluations. To address the limitations of LSTM, xLSTM designs *exponential* *gates* to enhance effectiveness with long sequences, expands memory cells from scalars to matrices to increase storage capacity, and removes memory mixing to enable parallel processing.

To test the language modeling capabilities of xLSTM scaled to billions of parameters, it is trained on a large dataset comprising 300 billion tokens from SlimPajama [321] across various model sizes (125M, 350M, 760M, 1.3B). The performance of pre-trained xLSTM is compared against RWKV-4 [277], Llama [156], and Mamba [119] across 571 text domains of the PALOMA benchmark [242] and various downstream tasks. Across all model sizes and the majority of tasks, xLSTM consistently outperforms the others, suggesting that larger xLSTM models could become formidable competitors to existing LLMs that utilize Transformer technology.

2.2 Training SLMs from Scratch

Training SLMs from scratch entails several critical steps: (i) Pre-training, focused on acquiring general features and knowledge from the corpus; (ii) Fine-tuning, targeted at boosting the model's abilities and performance for specific tasks; (iii) Decoding strategies, which involve the methods used for iteratively selecting the next token during generation.

2.2.1 Pre-Training. Typically, the pre-training paradigm for LMs is divided into encoder-based and decoder-based approaches. Encoder-based models, such as BERT [86], utilize **Masked Language Modeling (MLM)** tasks where the goal is to predict masked tokens within a sentence. This is achieved by maximizing: $P(\text{masked token} \mid \text{context}) = \text{softmax}(\mathbf{W} \cdot \mathbf{h}_{\text{mask}} + b)$, where the masked token is the original token that has been masked, context represents the other unmasked tokens in the sentence, $\mathbf{W}$ and b are trainable parameters of a linear output layer, $\mathbf{h}_{\text{mask}}$ is the output from the transformer encoder for the masked position, and softmax is the activation function that converts logits to probabilities over the vocabulary. This process enhances the model's language encoding capabilities. Decoder-based models, such as GPT [289], employ Next Token Prediction (NTP) tasks, aiming to model the distribution of the next token by maximizing $P(\text{next token} \mid \text{context}) = \text{softmax}(\mathbf{W} \cdot \mathbf{h}_{\text{last}} + b)$, where next token is the token that the model aims to predict, context represents the sequence of tokens preceding the token to be predicted, and $\mathbf{h}_{\text{last}}$ is the output from the transformer encoder for the last token in the context. Effective data preprocessing, crucial for optimizing the performance of SLMs trained from scratch, involves meticulous data cleaning and strategic tokenization.

Data Cleaning involves techniques such as filtering, deduplication, and noise reduction, which improve data quality and help the model generalize better. Filtering noisy or irrelevant data, addressing outliers, and handling imbalances in the dataset ensure that the training data is both representative and efficient. Deduplication, in particular, helps prevent overfitting by removing repeated instances, making the model more robust with efficient parameter usage.

2.2.2 Fine-Tuning. After the initial training, SLMs are fine-tuned on specific tasks using task-specific data and loss functions. Parameter-efficient fine-tuning methods [120, 143, 145, 203], such as **Low-Rank Adaptation (LoRA)**, prefix-tuning, and adapter modules, are particularly effective for SLMs. LoRA [145] modifies Transformer weights by introducing trainable low-rank matrices $\mathbf{A}$ and $\mathbf{B}$ for efficient fine-tuning, avoiding significant alterations to pretrained weights. The update is represented as: $\Delta\mathbf{W} = \mathbf{A}\mathbf{B}^\top$ The fine-tuned weight matrix used in Transformer operations then becomes: $\mathbf{W}_{\text{ft}} = \mathbf{W} + \alpha\Delta\mathbf{W}$, where α is a scaling factor adjusting the adaptation's impact, allowing fine-tuning on a smaller set of parameters while retaining the model's foundational capabilities. *Prefix-Tuning* [203] prepends learnable prefixes to the input sequence, guiding the model's attention without altering core model parameters. It is especially useful for generative tasks. *Adapter Modules* [143] are small, trainable layers inserted into the pre-trained model. These layers are fine-tuned on task-specific data, allowing the base model to remain fixed while the adapters learn the necessary adjustments. The typical structure of an adapter module includes a down-projection, a non-linearity, and an up-projection: $\text{Adapter}(\mathbf{h}) = \mathbf{h} + \mathbf{W}_{\text{up}} \cdot \sigma(\mathbf{W}_{\text{down}} \cdot \mathbf{h} + \mathbf{b}_{\text{down}}) + \mathbf{b}_{\text{up}}$, where $\mathbf{h}$ is the input hidden state, $\mathbf{W}_{\text{down}}$ and $\mathbf{W}_{\text{up}}$ are the projection matrices, $\mathbf{b}_{\text{down}}$ and $\mathbf{b}_{\text{up}}$ are the bias terms, and σ is a non-linear activation function.

2.2.3 Decoding Strategies. After pre-training or fine-tuning, employing an effective decoding strategy is crucial for generating output from language models. Decoding, the process of text generation from SLMs, involves iteratively selecting the next word. A fundamental method is the greedy search, which predicts the most likely token at each step. This is formally modeled as: $x_i = \arg\max_x P(x \mid x_{<i})$, where x_i is the token with the highest probability at the ith step, conditioned on the preceding context $x_{<i}$. Other decoding strategies, such as beam search or top-k sampling, are crucial for generating high-quality outputs. Beam search balances exploration and exploitation by considering multiple possible sequences simultaneously, while top-k sampling introduces diversity and creativity in text generation. These strategies collectively ensure that SLMs are efficient and capable of delivering high performance across various natural language processing tasks.

2.3 Obtain SLMs from LLMs

Obtaining an SLM from an LLM is crucial for deploying in resource-constrained environments. Instead of training from scratch, leveraging an LLM allows for knowledge transfer, enabling SLMs to retain much of the LLM's linguistic and domain knowledge with reduced training time and data. To obtain SLMs from LLMs, three primary techniques are used: pruning, KD, and quantization. Pruning removes less critical parameters, reducing model size while aiming to maintain performance. KD transfers knowledge from a large teacher model to a smaller student model, preserving much of the original model's understanding. Quantization decreases parameter precision, significantly lowering memory and computation needs with minimal impact on accuracy. These methods balance size reduction, efficiency, and performance retention.

2.3.1 Pruning. Pruning is a technique used to reduce a model's size and computational requirements (e.g., LLMs) without significantly sacrificing its performance [128]. This process involves identifying and removing less important or redundant parameters and components from the model. The primary goal of LLM pruning is to make the model more efficient, faster, and suitable for deployment in resource-constrained environments. Typically, pruning can be categorized into two main types: *unstructured pruning* and *structured pruning*. An illustration of unstructured pruning and structured pruning is shown in Figure 7.

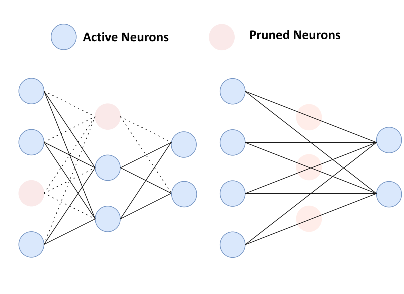


Fig. 7. Unstructured and structured pruning.

Unstructured Pruning [79, 104, 205, 307, 327, 443, 447] prunes an LLM by removing weights individually without considering its internal structure. The least significant parameters are pruned according to specific criteria, e.g., magnitude or impact on the output). This method can achieve significant compression while maintaining performance. However, it can also lead to irregular memory access patterns and reduced hardware efficiency because the pruned model lacks a regular structure. SparseGPT [104] is a representative unstructured pruning method that can reduce large-scale GPT models like OPT-175B [445] and BLOOM-176B [184] to up to 60% sparsity using a novel sparse regression solver. Wanda [327] combines weight magnitudes with input activations to efficiently identify and discard less impactful parameters.

It operates in a single forward pass, rapidly achieving high sparsity without retraining. It is also worth noting that recent studies specifically address the compatibility issues between pruning and LoRA [145], such as LoRAPrune [443].

Structured Pruning [14, 20, 53, 112, 126, 191, 200, 237, 248, 312, 313, 390, 415, 450], which prunes an LLM by targeting entire structural components—such as neurons, channels, or layers—rather. This approach allows for a direct reduction in dimensionality, thus efficiently reducing model complexity and memory usage. Although structured pruning may lead to higher accuracy degradation than unstructured pruning, it simplifies implementation without requiring specialized hardware. ShortGPT [248] proposes the **Block Influence (BI)** metric, which measures the significance of each layer based on its transformation of hidden states. Essentially, a transformer block's influence is measured by how much it alters the hidden states. By calculating BI scores, ShortGPT determines which layers contribute minimally to the overall performance and removes these low-importance layers. This simple yet effective layer removal strategy significantly reduces the model's parameters and computational requirements. LLM Pruner [237] offers a method to efficiently prune LLMs without access to the original training dataset. It employs a three-step compression pipeline: Discovery (identifying interdependent structures), Estimation (evaluating the performance impact of each group), and Recovery (post-training to address performance loss). NutePrune [200] enhances structured pruning with a Numerous-teacher method, employing variable sparsity masks and LoRA modules to guide the pruning process. This approach effectively reduces model size and complexity. COST-EFF [312] introduces a slenderized backbone—a form of structured pruning—and a multi-exit model that employs task-specific calibration through KD. This slenderization reduces the model's spatial footprint, while the multi-exit strategy effectively balances utility and runtime costs. To enhance the flexibility of structural pruning, DISP-LLM [112] breaks the structural dependencies in regular methods by allowing different layers to have different subsets of features along the embedding dimension.

2.3.2 KD. KD compresses a larger teacher model into a smaller student model by training the student to mimic the teacher's outputs [139]. This enables the student to retain much of the teacher's capabilities with fewer parameters, making it ideal for scaling down LLMs for resource-limited environments while maintaining performance. KD can be categorized into *white-box* and *black-box* approaches [367, 408, 461] as shown in Figure 8. In *White-Box KD*, the student has access to the teacher's internal states or output distributions [6, 121, 160, 172, 176, 272, 386, 439]. **Generalized Knowledge Distillation (GKD)** [176] introduces skew KL divergence to stabilize gradients and enhance performance, using an adaptive off-policy approach to minimize noisy feedback and improve efficiency. *Black-Box KD* relies only on teacher outputs without having access to model internals [50, 185, 198, 278, 366]. Methods like Distilling Step-by-Step [144] use teacher-generated rationales to train smaller models, improving performance with fewer examples. LaMini-LM [385] creates a diverse instruction dataset with GPT-3.5 Turbo responses, enabling robust performance in smaller models.

2.3.3 Quantization. Quantization reduces the storage and computational demands of LLMs by converting floating-point representations into lower-precision formats, significantly cutting both storage requirements and computational complexity. Existing methods fall into two categories: **Post-Training Quantization (PTQ)** and QAT. Figure 9 illustrates the two quantization methods. *PTQ*, applied after training, simplifies model compression without altering the architecture or requiring retraining, though it may result in precision loss. Consider a group or block of weights $\mathbf{w}$; the linear operation can be expressed as $y = \mathbf{wx}$, while the quantized version is given by $y = Q(\mathbf{w})\mathbf{x}$. Generally, the quantization function Q is defined as [212]: $Q(\mathbf{w}) = \Delta \cdot \text{Round}\left(\frac{\mathbf{w}}{\Delta}\right), \Delta = \frac{\max(|\mathbf{w}|)}{2^{N-1}}$,

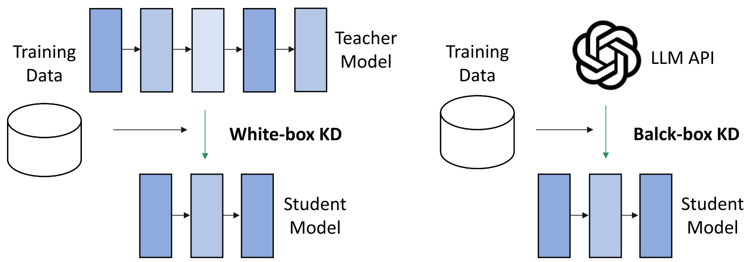


Fig. 8. Illustration of white-box and black-box KD [251].

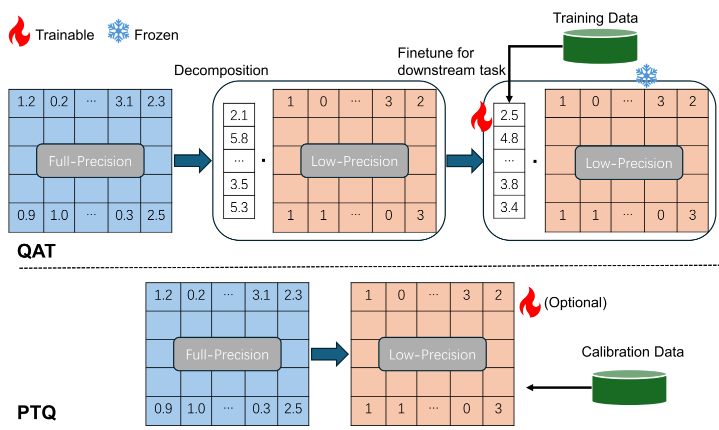


Fig. 9. Illustration of QAT and PTQ.

where N is the number of quantization bits, and Δ is the quantization scale factor determined by the absolute maximum value of $\mathbf{w}$. Quantizing weights reduces model size and storage bandwidth, while quantizing activations directly reduces memory traffic and inference latency—especially critical for hardware accelerators such as GPUs and TPUs. However, activation quantization is often more sensitive due to its dynamic range variation across inputs and layers. *QAT* enhances LLM efficiency by directly incorporating quantization into the training process, often resulting in higher accuracy than PTQ. During QAT, the forward pass utilizes quantized weights $Q(\mathbf{W})$ and activations $Q(\mathbf{X})$, while retaining full-precision values during the backward pass and for updating gradients to ensure stable learning dynamics. The comparisons of the PTQ methods are summarized in Table 1, detailing precision, addressed problems, and technical contributions of each method.

In Table 2, we summarize recent quantization methods for language models, which demonstrate that substantial compression can be achieved with minimal performance degradation. Several approaches, such as OneBit [405], BitNet [363], and BiLLM [150], achieve 8–16× parameter compression while maintaining perplexity or accuracy within 1–2% of full-precision baselines. Notably, BitNet b1.58 [236] and PEQA [171] even match or surpass the performance of their full-precision counterparts. These methods are evaluated on a wide range of benchmarks, including WikiText2 [249], C4 [291], MMLU [136], and LM-Eval [111], ensuring a comprehensive assessment. Hybrid compression techniques are also gaining popularity; for example, JSQ [124] combines 4-bit quantization with 50% sparsity, achieving up to 8× compression with less than a 2% accuracy drop. In addition, hardware-aware strategies are becoming increasingly important: I-LLM [147] emphasizes integer-only inference for efficient deployment, and QLoRA [84] reduces memory usage by 4× while enabling low-resource fine-tuning with negligible performance loss. Overall, these results suggest

Table 1. Representative Quantization Methods

Methods	Bit	Type	Technical Contribution	Problems
SqueezeLLM [173]	3-Bit	PTQ	Sensitivity-based non-uniform quantization, dense and sparse decomposition	Ultra-low bit quantization
JSQ [124]	Flexible	PTQ	Joint Sparsification and Quantization	Better compression-accuracy trade-offs.
FrameQuant [4]	Fractional bit	PTQ	Fractional bit widths	Better compression-accuracy trade-offs.
OneBit [405]	1-bit	PTQ	Quantization-aware KD	1-Bit quantization
BiLLM [150]	1-bit	PTQ	Crucial Weights Selection, Block-based error compensation	1-Bit quantization
LQER [437]	Flexible	PTQ	Quantization Error Minimization	Better compression-accuracy trade-offs
I-LLM [147]	Flexible	PTQ	Fully Smooth Block Reconstruction, Dynamic Integer-only MatMul and Integer-only Non-linear Operators	Integer-only Quantization
PV-Tuning [244]	1-Bit/2-bit	PTQ	PV algorithm	Better compression-accuracy trade-offs.
BitNet [363]	1-Bit	QAT	1-Bit Transformer Architecture	1-Bit quantization
BitNet b1.58 [236]	−1, 0, 1	QAT	Ternary Parameters	1-Bit quantization
PEQA [171]	Flexible	QAT	Quantization Scales Optimization	Parameter-Efficient Fine-tuning
QLoRA [84]	NF4	QAT	4-Bit Normal Float and Double Quantization	Parameter-Efficient Fine-tuning

that 3–4 bit quantization strikes an effective balance between compression, inference efficiency, and accuracy, making it a practical solution for deploying LLMs in resource-constrained settings.

2.3.4 Low-Rank Techniques. Low-rank techniques compress LLMs by approximating a high-dimensional weight matrix with two lower-dimensional matrices, reducing computational and memory requirements. A matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ is approximated as $\mathbf{W} \approx \mathbf{A} \times \mathbf{B}$, where $\mathbf{A} \in \mathbb{R}^{m \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times n}$, with r much smaller than m or n, reducing the number of parameters. Building on this concept, Ji et al. [161] propose a low-rank method tailored for LLMs, leveraging the observation that while LLMs have high-rank weights, their feature interactions tend to exhibit low-rank properties. The method estimates feature distributions using pooled covariance matrices and allocates distinct compression ratios to layers based on their sensitivity to low-rank compression. A Bayesian optimization strategy, using a Gaussian process as the surrogate model, optimizes the allocation of low-rank dimensions, ensuring the model maintains performance while achieving significant compression. Transitioning from model compression to fine-tuning, Cho et al. [64] tackle system and data heterogeneity with the HETLORA method, which uses heterogeneous low-rank approximations to accommodate the diverse capabilities of clients and data complexities. By combining local rank self-pruning with sparsity-weighted aggregation, it balances high- and low-rank LoRA modules, improving convergence speed and performance compared to uniform approaches. LLM-Neo [414] combines KD with low-rank adaptation (LoRA) to improve the efficiency of transferring knowledge from a teacher LLM to a compact student model.

3 Advanced Enhancement Strategies for SLMs

With the foundational concepts introduced in Section 2, this section explores various advanced techniques that enhance the performance of SLMs, including innovative training methods for training SLMs from scratch, **supervised fine-tuning (SFT)** to align SLMs to adhere to instructions, advanced KD and quantization techniques, and techniques frequently used in LLMs, such as MoEs to enhance SLMs for specific applications. A summary of enhancement techniques is also summarized in Table 3.

Table 2. Summary of Quantization Methods with Evaluation Details

Method	Evaluated Base Models	Benchmarks	Full-Precision vs Quantized Performance	Compression	Speed up
SqueezeLLM [173]	LLaMA-2 7/13/70B	WikiText2 PPL [249]; C4 [291]	FP16 PPL 5.52 (C4); 3-bit PPL 5.73 (C4)	~5.3× smaller	Up to ~2.3× faster
JSQ [124]	LLaMA-7B/13B/33B; LLaMA2-7B	PIQA [33]; BoolQ [67]; MMLU [136]	Similar acc @ 50% sparsity + 4b quant.; <2% avg. drop	Up to ~8× (sparsity+ quant.)	–
FrameQuant [4]	LLaMA-7B/13B	WikiText2 PPL; C4	<0.5 pt avg. drop vs 4 b GPTQ	4–8×	Faster than GPTQ (same precision)
OneBit [405]	LLaMA-7B/13B; OPT-2.7B	WikiText2 PPL; C4; LM-Eval subset	≥81% of FP16 avg.; similar in few-shot	16×	–
BiLLM [150]	LLaMA family	WikiText2 PPL; C4; LM-Eval subset	~2% PPL increase	8–16×	–
LQER [437]	LLaMA and OPT families	WikiText2 PPL	0.3% accuracy drop	4× (W4) to 5.3× (W3)	–
I-LLM [147]	LLaMA families	WikiText2 PPL; C4	<1pt drop at 8 b int-only; modest at 4 b	2–4×	Integer matmul accel. 1.5–3×
BitNet [363]	BitNet (1.3B–6.7B)	WikiText2 PPL; Zero/Few-shot	Δavg ≲2pts zero-shot vs FP16	16×	
BitNet b1.58 [236]	BitNet b1.58 (3B+)	WikiText2 PPL	Matches FP	10.1×	–
PEQA [171]	LLaMA families	LM-Eval subset	Recovers or improves vs FP <4b	4–8×	Fine-tuning memory ↓10–20×
QLoRA [84]	LLaMA families	MT-Bench; MMLU; custom data	<1% drop vs FP LoRA	4×	–

3.1 Innovative Training Methods for SLMs from Scratch

In scenarios with limited resources, we aim to train SLMs to provide efficient, cost-effective solutions tailored for specific domains while still maintaining competitive performance with larger models. Training SLMs from scratch involves unique strategies that diverge significantly from those used for LLMs. This section synthesizes cutting-edge techniques tailored to optimize the inherent capabilities of SLMs, underscoring their potential to match or surpass larger counterparts in efficiency and effectiveness. As shown in Figure 10, the methods for training SLMs from scratch can be categorized into three primary categories: *Architecture Design*, *Data Construction*, and *Optimization Strategy*. Next, we introduce each category in detail.

Architecture Design for SLMs. When designing SLM architectures, parameter-sharing techniques are employed to minimize space usage and reduce the model's size. As shown in the first part of Figure 10, parameter sharing is achieved by two approaches: (i) a single Feed-Forward Network (FFN) module is shared by every transformer layer. As shown in Figure 10(1) middle, *FFN layer sharing/reusing* can maintain a smaller size while still benefiting from the depth and complexity gained through repeated processing of input data. This technique is firstly applied in MobiLlama [342] which surpasses the performance of existing SLMs of comparable size. (ii) Entire transformer blocks are shared. As shown in Figure 10(1) right, *Transformer Block-wise Sharing* is another

Table 3. Advanced Enhancement Methods for SLM

Topic	Method	Main Contribution
Training from Scratch	MindLLM [417]	Bilingual models with advanced features.
	MobiLlama [342]	On-device SLM with dual objectives for efficiency and capability.
	MobileLLM [227]	Optimizes LLM deployment on mobile with advanced architecture.
SFT	MobileBERT [328]	Compact BERT for efficient fine-tuning.
	Alpaca 7B [335]	Uses ChatGPT-generated tasks to tune Llama 7B.
	RLHF [271]	Trains using human-preferred data and reinforcement learning.
	DPO [290]	Dynamically adjusts log probabilities to prevent model degradation.
Data Quality in KD	TinyStory [98]	Enhances narrative coherence in child-friendly datasets.
	AS-ES [389]	Improves CoT by categorizing reasoning steps.
	Self-Amplify [30]	Automates CoT data annotation for small models.
Distillation for SLM	GKD [6]	Aligns training and inference distributions using on-policy sequences.
	DistiLLM [176]	Uses skew KL divergence and adaptive off-policy for output utilization.
	Adapt-and-Distill [419]	Domain adapts both teacher and student models before distillation.
Quantization	SmoothQuant [391]	Balances quantization difficulty using per-channel scaling.
	BiLLM [150]	Applies Hessian-based metrics for binary residual approximation.
	LLM-QAT [226]	Uses data-free KD and logit distillation for fine-tuning.
	PB-LLM [306]	Binarizes non-salient weights while preserving others in higher precision.
	OneBit [405]	Achieves near 1-bit quantization with minimal performance loss.
	BitNet [363]	Introduces 1-bit Transformer architecture with BitLinear layers.
	BitNet b1.58 [236]	Implements a ternary weight system in enhanced BitNet.
LLM techniques for SLM	Ma et al. [239]	Combines filtering and re-ranking to improve Information Extraction tasks.
	MoQE [175]	Applies quantization to expert weights to outperform dense models.
	SLM-RAG [220]	Shows that SLMs with RAG can match LLM performance.

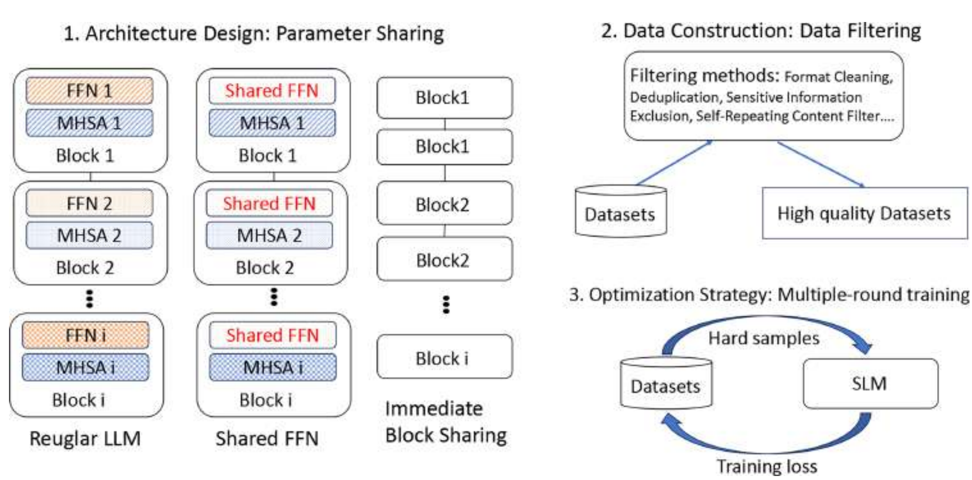


Fig. 10. Innovative training methods for SLMs from scratch.

parameter-sharing approach that maintains depth and complexity. There are different transformer block-wise sharing strategies, such as repeating the transformer blocks all over again or repeating the immediate transformer block. This technique is applied in MobileLLMs [227], which have 125M and 350M parameters. MobileLLMs demonstrate performance improvements of 2.7% and 4.3%, respectively, compared to previous models with equivalent parameters. Moreover, they exhibit accuracy comparable to LLaMa-2-7B on API call tasks, highlighting the capabilities of smaller models in mobile environments.

Data Construction. For SLMs, the emphasis on data quality surpasses that of quantity and diversity [417]. Experiments demonstrate that using a quality filtering approach to remove low-quality data can lead to improved performance in SLMs [417]. Unlike large models, which can handle diverse and large datasets, SLMs benefit more from cleaner, high-quality data probably due to their limited capacity against noise. Generally, data processing has several steps: (i) Remove HTML, CSS, JS, and non-text elements for clean text; (ii) Filter low text-to-content ratio web pages; (iii) Deduplicate using SimHash [80, 300]; (iv) Exclude sensitive/offensive content with heuristics and token replacements; (v) Remove self-repeating phrases of advertisements to enhance dataset informativeness [48, 417]. These steps collectively ensure that training data has high-quality, informative texts. SLMs also significantly benefit from these techniques. For example, MindLLMs [417], which are bilingual lightweight language models (available in 1.3B and 3B versions), adopt these data processing techniques and achieve improved capability acquisition.

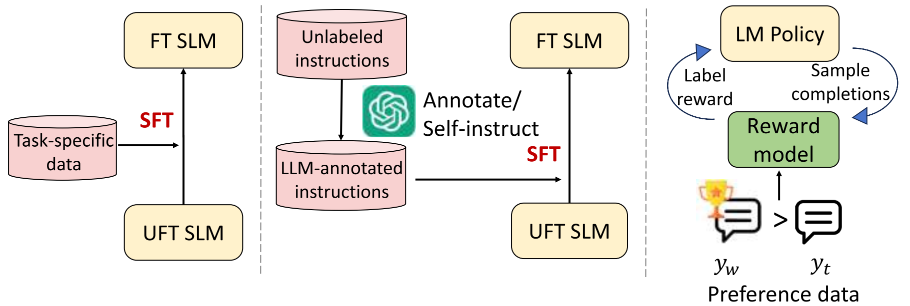
Training Strategy for SLMs. For LLMs, due to the large model size and data volume, LLMs are usually trained with one round. For SLMs, multiple-round training can be applied [334]. Considering some examples are hard to fit, hard examples can be trained with a high probability [334]. For each round of training, the data sampling probability is updated according to the overall loss of that sample. Experiment results show that two rounds of training and a 50% sampling rate are a good tradeoff between performance and training efficiency. Tang et al. [334] show that a deep and thin neural architecture and multiple-round training can enhance the performance of the trained Pangu 1.5B pro model. This model outperforms the conventionally trained Pangu 1.5B and a series of other comparable LLMs with similar model sizes on multiple benchmark datasets, achieving an average performance increase of 8.87%.

3.2 SFT for Enhancing SLM Performance

SFT employs a training methodology similar to pre-training but is specifically tailored to align models to adhere to the instructions encapsulated within various instructional datasets. This approach is designed to refine the model's responsiveness and appropriateness to given contexts as the training data dictates. For example, various models, such as Alpaca [335], UltraChat [89], WizardLM [397], SlimOrca [207], ShareGPT [362], Capybara [75], Deita [221], VLAA-Thinking [49], GALLM [234], and MetaMathQA [425], incorporate a suite of conversational datasets to enhance their capabilities in context-aware dialogue and instruction adherence. Usually, as shown in Figure 11, existing SFT methods can be categorized into three categories:

(i) *Classical fine-tuning with downstream data* [86, 289] trains SLMs on task-specific annotated data, transferring general language representations to specific tasks such as sentiment analysis. In the LLM era, this approach remains effective, such as enhancing LLMs by calibrating responses or assigning risk scores with smaller models such as BERT [454], or optimizing for mobile devices with MobileBERT [328].

(ii) *Instruction tuning* with LLM-generated data [89, 207, 335] or human-generated questions with LLM annotations [362] aims to align generative models with specific instructions, enhancing their instruction-following and reasoning capabilities. For example, *Alpaca 7B* [335] uses 52k ChatGPT-generated instruction-following examples from 175 self-instructed seed tasks to tune Llama 7B [344]. Meanwhile, StableLM [27, 346] is trained on the Restruct-v1 dataset, which includes summarization, **question-answering (QA)**, and sentiment analysis tasks, using instruction data from [230].



(a) Fine-tuning on downstream tasks. (b) Fine-tuning on instructions (instruction tuning). (c) Preference Optimization.

Fig. 11. Fine-tuning for enhancing SLMs.

(iii) *Preference optimization with human feedback* [271, 290, 362] aims to better align language models with human preferences. Reinforcement Learning from Human Feedback (RLHF) [271] gathers human-preferred data, trains a reward model, and fine-tunes the LM using reinforcement learning. DPO [290] provides a simpler alternative to RLHF. Unlike RLHF, DPO avoids explicit reward modeling and reinforcement learning techniques. Instead, it adjusts the log probabilities of preferred versus non-preferred responses using a dynamic weighting mechanism, preventing model degradation issues typical of methods relying on probability ratios. For instance, Llama 3.2 1B and 3B apply SFT and DPO in post-training to enhance alignment with instructions and human preferences.

3.3 Data Quality in KD

Transitioning from the discussion on training SLMs from scratch, this section delves into the critical role of data quality in KD. The motivation here is to highlight how high-quality data generated from LLMs can significantly enhance the learning efficiency and performance of SLMs. The central idea is that meticulously crafted datasets, when used in KD, enable SLMs to more effectively mimic the advanced capabilities of their larger counterparts. As shown in Figure 12, the data can come either from (1) other strong LLMs (e.g., GPT-4 [2]), which are much larger and more powerful than the target SLM, or (2) the target SLM itself.

Augment Data from LLMs. LLM-generated data could be categorized as *pre-training data* and *fine-tuning data*. Firstly, due to the limitations of model size, studies have shown that training SLMs requires simple and comprehensible data [98, 186, 190, 389]. As shown in Figure 12(1) left, *TinyStory* [98] shows that small models (tens of millions of parameters) can generate coherent stories for 3-4-year-olds. GPT-3.5 or GPT-4 [2] prompts create simple stories from three keywords chosen from a 1,500-word vocabulary, which are then used to train SLMs for similar outputs. This approach shows that simple and comprehensible data can help smaller models exhibit behaviors similar to those of larger language models, such as obeying scaling laws and achieving enhanced performance. On the other hand, many efforts to enhance the **Chain-of-Thought (CoT)** capabilities of small models involve using LLMs to generate high-quality fine-tuning CoT data. As shown in Figure 12 (1) right, these data train small models end-to-end to mimic CoT reasoning [240, 389]. *AS-ES Learning* [389] highlights that small models struggle with complex reasoning, even when provided detailed steps, as these require nuanced extraction and abstraction. Therefore, the study introduces a paradigm that splits reasoning into extractive segments (context reminders) and abstractive segments (inferred insights).

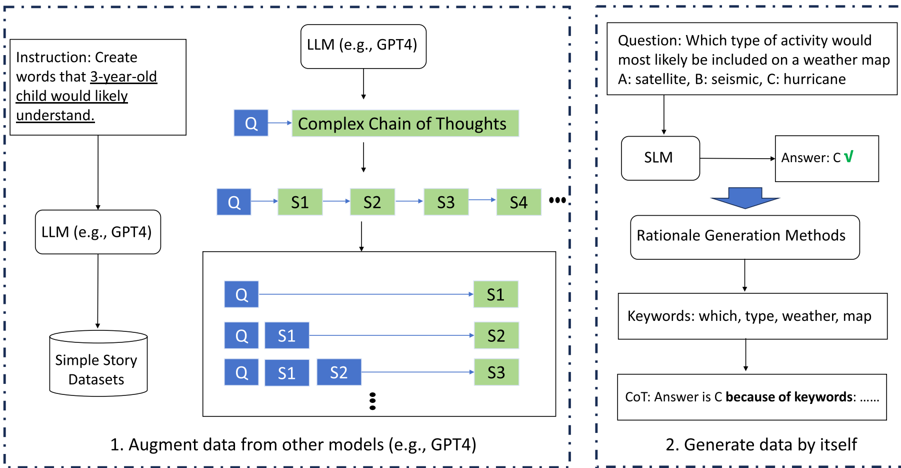


Fig. 12. Data quality in KD.

Augment Data from Itself. Besides distilling data from other LLMs, language models can also train on their own outputs [30, 148, 343]. Since voting strategies can improve the performance of LLMs, reasoning paths that lead to the majority answer can be further utilized to fine-tune LLMs [148]. Similarly, SLMs can generate their training data with the aid of existing rationale generation methods. *Self-Amplify* [30] notes that human annotation of CoT data is very time-consuming; thus, automated rationale generation methods have been proposed. These methods involve three main steps: (1) Selection of samples (x, y) that the model predicts correctly as few-shot examples; (2) Rationale generation, where rationales are produced using post hoc explanation methods; (3) Prompt design for SLMs, where the final prompt is crafted based on the previously generated rationales.

3.4 Distillation Techniques for Enhancing SLM Performance

Following the discussion on data quality in KD, this section reviews specialized KD training strategies designed to enhance the performance of SLMs. The motivation is to address the unique challenges and constraints involved in distilling knowledge from LLMs to SLMs, ensuring that the smaller models can maximize their performance gains. As shown in Figure 13, two main gaps between LLMs and SLMs lead to challenges in distillation: *distribution mismatch* and *domain gap*. *Distribution mismatch* [6, 176] occurs when the distribution of output sequences during training does not align with the distribution of sequences that SLMs produce during inference, leading to suboptimal performance of the student model. The *domain gap* [419] arises when there is a discrepancy between the domains or tasks on which the LLMs and SLMs are trained and applied. This gap can cause significant degradation in the performance of the student model if not properly addressed during the distillation process. To address these issues, specialized strategies involve first aligning the teacher and student models with the target domain before proceeding with knowledge distillation. To explore these challenges further, we now delve into the details of these two branches of methods.

Distribution Mismatch. In original KD illustrated in Figure 13 Distribution Mismatch (a), the teacher and student are provided with the same input sequences x and output labels y, producing probability distributions for the next token (q and p). The loss is calculated as the difference between these two distributions, $D(q, p)$. However, a key challenge arises due to distribution mismatch: the

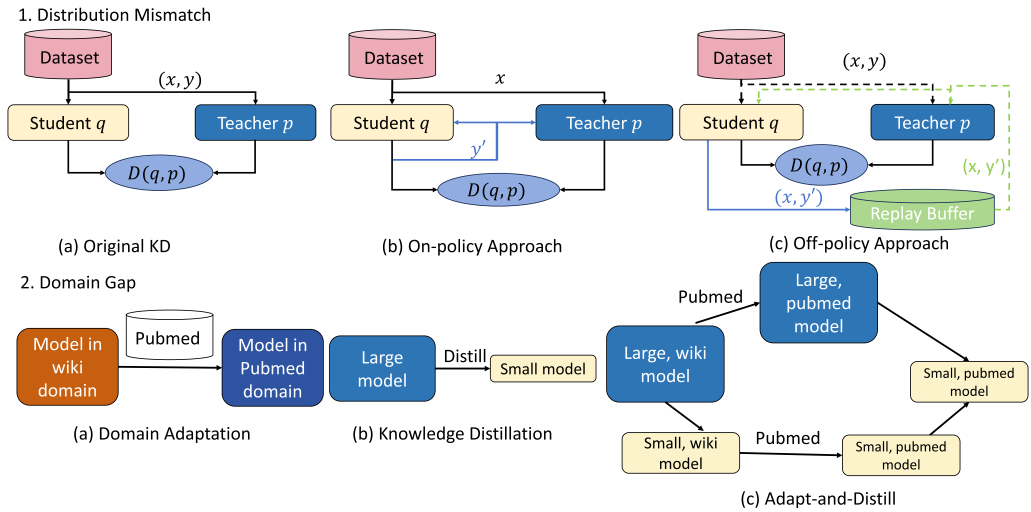


Fig. 13. Distillation techniques for enhancing SLM performance. On-policy means learning only uses data from the current student (policy), while off-policy permits the use of previously gathered data.

output sequences during training (y) differ in distribution from those the SLMs produce during inference (y'). To address this challenge, various techniques have been proposed. As shown in Figure 13 Distribution Mismatch (b), one approach trains the student model using on-policy sequences—sequences generated by the student itself—guided by the teacher model's feedback. Specifically, both the student and teacher take the same input (x) and the student-generated output (y'), producing probability distributions for the next token (q and p, respectively). The loss is calculated as the difference between these two distributions, $D(q, p)$. This approach helps the student model reduce the distribution gap between training and inference by learning from the teacher's feedback on its own generated sequences. *GKD* [6] is the first work using this technique and improves distillation outcomes. However, a drawback of this technique is that it requires the student to constantly produce new training sequences, which can be computationally expensive. To improve efficiency, as shown in Figure 13 Distribution Mismatch (c), an adaptive off-policy approach can be used to efficiently manage student-generated outputs by storing them in a replay buffer, thereby reducing computational costs. *DistiLLM* [176] employs this off-policy approach and improves the efficiency of KD. While it has approximately 1.5× the training time of vanilla distillation methods, it achieves up to a 2.3× speedup compared to on-policy approaches like GKD. In addition to improved efficiency, DistiLLM boosts ROUGE-L scores of distilled SLMs across five instruction-following benchmarks, outperforming both GKD and vanilla distillation.

Domain Gap. When training an SLM in a specific domain that differs from the domain of the LLMs, the gap between the two domains becomes problematic. As illustrated in Figure 13 Domain Gap (a), domain adaptation fine-tunes a language model, initially trained on a general corpus, using a specialized dataset such as PubMed to enhance performance in that specific domain. As illustrated in Figure 13 Domain Gap (b), KD transfers knowledge from the larger model to the smaller one. However, because the teacher model may not produce high-quality outputs on specialized datasets, domain adaptation is needed before KD. As illustrated in Figure 13 Domain Gap (c), *Adapt-and-Distill* [419] tackles the domain gap by distilling general large models into smaller ones. This article introduces AdaLM and demonstrates that the "Adapt-and-Distill" strategy—first involving domain adaptation of both the large teacher model and the small student model, followed by distillation—is the most effective compared to three other strategies: training directly from scratch, distillation followed by adaptation, and adapting the teacher model before distillation into a general small student model. These innovative techniques are crucial for enhancing the capabilities of SLMs, making them more efficient and effective for various applications. However, adapting both the teacher (LLMs) and the student (SLMs) models to the target domain can be time-consuming. Future research could focus on efficiently solving the domain gap problem. In experimental evaluations, Adapt-and-Distill produces a 6-layer student model that is 3.3× smaller and 5.1× faster than the BERT base model, while consistently outperforming BERT on both biomedical and computer-science downstream tasks.

3.5 Performance Improvement through Quantization

As mentioned in Section 2, quantization is one of the most effective methods for adapting LLMs to SLMs. However, compression to smaller sizes often compromises performance. To address the performance drop associated with quantization, various methods have been proposed. This section examines how these quantization methods specifically enhance the performance of SLMs. While the general introduction to compression methods is discussed in the compression section, the focus here is on detailing those approaches that boost the efficiency and effectiveness of SLMs. As shown in Figure 9, we categorize these quantization methods into two main approaches: PTQ, where quantization is conducted on a well-trained fixed model, and QAT, where quantization is integrated into the training process. This section introduces advanced techniques in PTQ and QAT, respectively.

PTQ primarily includes weight quantization and activation quantization. Weight quantization aims to quantize model parameters while preserving performance. *GPTQ* [105] compresses LLMs to 4-bit or 2-bit by quantizing weights layer-by-layer to minimize layer-wise quantization errors. GPTQ quantizes OPT-175B and BLOOM-176B in approximately four GPU hours with a negligible increase in perplexity, and the resulting 3-bit OPT-175B model achieves up to a 3.25× speedup on a single NVIDIA A100 GPU and 4.5× on two NVIDIA A6000 GPUs, while fitting the model into a single 80GB A100. *PB-LLM* [306], applicable to both PTQ and QAT, retains the most salient weights while binarizing the rest based on magnitudes. *BiLLM* [150], another PTQ method, uses a Hessian-based metric to identify salient and non-salient weights. Salient weights undergo binary residual approximation to minimize loss, while non-salient weights are divided into sparse and concentrated groups for separate binarization, reducing quantization errors. Activation quantization faces challenges with outliers that can stretch the quantization range, causing most values to cluster at a few bits and introducing significant errors. To address this, *LLM.int8()* [83] isolates outlier features for 16-bit processing and handles the rest in 8-bit. *SmoothQuant* [391] circumvents per-channel quantization issues by employing a "smoothing" technique that shifts the quantization challenge from activations to weights through a per-channel scaling transformation. This balance between activating and weight quantization allows effective 8-bit quantization (W8A8), preserving accuracy while significantly reducing memory and computational costs. SmoothQuant thus enhances the efficiency of SLMs in resource-constrained environments. SmoothQuant achieves up to 1.56× inference speedup and 2× memory reduction on LLMs with negligible accuracy loss.

QAT differs from PTQ in that it includes a training phase after the model has been quantized. When models are quantized to extremes, such as 2-bit or 1-bit, performance typically drops significantly, but further training can help the model retain its capabilities. For instance, to mitigate performance degradation from binarization, *PB-LLM* [306] selectively binarizes only non-salient weights, preserving the most salient ones at higher precision. This method effectively reduces the model size without significantly impacting performance. Salient weights are chosen based on their magnitude, ensuring that the most influential weights maintain higher precision to preserve the model's reasoning capabilities. The article explores both PTQ and QAT to fine-tune and recover the performance of partially binarized models, achieving a balance between compression and accuracy. *OneBit* [405] and *BitNet* [363] address the severe performance degradation associated with 1-bit quantization by decomposing floating-point matrices and employing mixed-precision strategies. Specifically, OneBit introduces Sign-Value-Independent Decomposition (SVID), which decomposes a floating-point matrix into a 1-bit matrix and two floating-point vectors. This method allows LLMs to be quantized to a 1-bit level while minimizing performance loss. By retaining critical information with the floating-point vectors, OneBit effectively balances extreme compression with maintaining model accuracy. *BitNet b1.58* [236] improves on the original BitNet by introducing a ternary matrix weight system of −1, 0, 1, resulting in a 1.58-bit model. BitNet b1.58 matches the performance of full-precision models starting from a 3 billion parameter size while further reducing memory and latency costs. *LLM-QAT* [226] employs data-free KD where the pre-trained model itself generates data for fine-tuning the quantized model (student) using logit distillation from the full-precision model (teacher). This method incorporates quantization of weights, activations, and key-value cache, achieving accurate 4-bit quantization for weights and key-value caches and 6-bit for activations, demonstrating substantial improvements over existing PTQ methods.

3.6 Techniques in LLMs Contributing to SLMs

This subsection explores the potential of advanced techniques such as RAG and MoE, which enhance LLM performance, to also maintain or boost SLM performance within constrained computational budgets. However, effectively integrating these techniques into SLMs, which inherently possess limited capabilities, remains an unresolved challenge.

Retrieval Augmented Generation (RAG) enhances the capabilities of language models in knowledge-intensive tasks by incorporating a retrieval mechanism. This approach allows models to access relevant contextual information from a data repository in response to user queries. By integrating this retrieved data, RAG-equipped models better understand specific topics, enabling more informed and accurate outputs. For SLMs, a significant concern is whether they possess the capacity for long-context reasoning. A recent study [220] compares SLMs at the 7B level with RAG to larger models such as GPT-3.5 and GPT-4, suggesting that SLMs equipped with RAG can sometimes perform comparably or even better than LLMs. These findings indicate that RAG for SLMs is effective and represents a promising direction for future research.

MoE [40] has emerged as an effective method for substantially scaling up model capacity with minimal computation overhead in LLMs. The MoE framework is founded on a straightforward yet potent concept: distinct components of a model, referred to as "experts," specialize in different tasks or data facets. In this paradigm, only the relevant experts are activated for a specific input, which manages computational costs while leveraging a vast pool of specialized knowledge. This scalable and adaptable approach enables increased model capacity without proportionally escalating computational demands. We argue that MoE is particularly suitable for SLM architectures [164] as it minimizes both computational load and memory overhead. However, research on MoE for SLMs remains sparse. Future studies could investigate how large LLM MoE architectures can be effectively compressed into small ones or how to develop an SLM with MoE tailored for specific devices from scratch.

4 Applications of SLMs

In this section, we delve into the applications of SLMs across various NLP tasks and their deployment strategies. Due to benefits such as enhanced privacy, faster inference, and lower memory requirements, many NLP applications are now leveraging SLMs over LLMs. Additionally, deploying SLMs often involves considerations of memory and runtime efficiency, which are crucial for

Table 4. Task-Specific SLM Applications

Aspect	Representative Work	Key Point
SLM in QA	Alpaca [335]	Tune Llama 7B [344] using 52k ChatGPT-generated examples.
	Stable Beluga 7B [243]	Employ explanation tuning to Llama-2 7B [345] on an Orca-style dataset.
	Fine-tuned BioGPT [127]	Fine-tuning BioGPT (1.6B) [235] on PubMedQA.
	Financial SLMs [282]	Transfer financial knowledge from GPT-4 [2] to multiple SLMs.
	ColBERT [114]	Fetch retrieval documents for SLMs to answer domain-specific questions.
	Rationale Ranking [130]	For unseen questions, combine retrieval with LLM-generated rationales.
	T-SAS [159]	Enhance SLMs adaptability with self-generated pseudo labels.
SLM in Coding	Phi-3.5-mini [1]	New addition to the Phi-3 series and focus on high-quality data.
	TinyLlama [444]	1.1B Transformer model is trained on 3 T corpus.
	CodeLlama [299]	A derivative of Llama 2 fine-tuned on domain-specific datasets.
	CodeGemma [337]	Fine-tuning Gemma to enhance coding capabilities.
SLM in Recommendation	PromptRec [387]	Training on prompt templates.
	SLIM [375]	Step-by-step KD
	BiLLP [316]	LLaMa-2-7B as planner and reflector.
	ONCE [219]	LLaMa-2-7B as Content Encoder.
	RecLoRA [459]	Personalized low-rank adaptation.
SLM in Web Search	Content encoder [45, 155, 232]	Encode concatenated queries and documents.
	Ranker [65, 267]	Re-rank retrieved documents using a special SLM.
	Rewriter [238]	Bridge the gap between queries and needed knowledge by rewriting inputs.
SLM in Mobile-device	Octopus [54]	Calling software APIs via learning in documents.
	MobileAgent [90]	Standard Operating Procedure (SOP).
	α-UMI [314]	SLMs serve as Multi-agents in tool uses.
	Mobile Interaction [43]	Text-to-action control and tests on 6 GB and 4 GB Android devices.
	AutoDroid [381]	Interaction based on GUI and APP knowledge injection.
	M4 [428]	A foundation model handling all mobile AI tasks.
	Agent for Text Rewriting [462]	Data KD from LLMs.

optimizing resource use on budget-constrained edge devices, particularly mobile phones. Then, we will discuss task-specific applications of SLMs and their deployment methods on mobile and edge devices.

4.1 Task-Specific SLM Applications

This subsection explores the diverse NLP tasks to which SLMs can contribute. QA and coding represent generative tasks, while recommender systems and web search (though not strictly within the NLP domain) typically leverage the encoding capabilities of SLMs. Additionally, the application of SLMs on mobile devices is particularly well-suited due to constraints in memory and computing resources. The representative works are systematically organized in Table 4.

4.1.1 SLM Applications in QA. QA is a fundamental task in the NLP field, demanding language models to exhibit abilities in understanding language, reasoning, common sense, and recalling specialized knowledge. Typically, larger language models yield better QA performance. However, the substantial size of these models introduces challenges such as immense computational requirements, privacy concerns when using proprietary LLMs, and difficulties in customization. These issues lead researchers and developers to favor SLMs in scenarios that demand efficiency, privacy, and customization. Therefore, we explore methods to enhance the capabilities of SLMs in QA across three key areas: (i) Instruction Tuning of Generic SLMs for QA, (ii) Instruction Tuning of Domain-Specific SLMs for QA, and (iii) Enhancing SLMs for Out-of-Domain Questions.

Instruction Tuning Generic SLMs for QA. Despite the Phi series' high QA capability, its training cost with over 3.4 T tokens on 512 H100 GPUs for 10 days [1] is prohibitive for many researchers and developers. Instruction tuning [377] offers a cost-effective alternative, enhancing small models by fine-tuning on large model outputs. Alpaca 7B [335] tunes Llama 7B [344] with 52k ChatGPT-generated examples from 175 seed tasks. This behavior cloning mimics teacher models effectively

Table 5. Comparison of Instruction-Tuned Domain SLMs for QA and LLMs on FinQA [58] and PubMedQA [166]

Model	Size	Instruction Tuned?	Task Name	Shot Type	Accuracy (%)
GPT-4 [2]	-	×	FinQA	Zero-shot	77.5
Phi-3-Mini [1]	2.7B	✓	FinQA	Zero-shot	77.6
Meditron-70B [57]	70B	×	PubMedQA	Zero-shot	81.6
RankRAG-llama3-70B [426]	70B	×	PubMedQA	Zero-shot	79.8
Flan-PaLM [320]	540B	×	PubMedQA	Few-shot	79.0
GAL 120B [336]	120B	×	PubMedQA	Zero-shot	77.6
Flan-PaLM [320]	62B	×	PubMedQA	Few-shot	77.2
BioGPT [235]	345M	✓	PubMedQA	Zero-shot	78.2
BioGPT-Large [235]	1.5B	✓	PubMedQA	Zero-shot	81.0

but struggles in reasoning-intensive QA tasks where accuracy is key, not style [62]. To counter it, explanation tuning [243] enhances Llama-2 7B [345] using explanatory LLM answers to improve reasoning. However, its effectiveness varies with system instructions, and those effective for larger models like GPT-4 may not suit smaller ones. SLMs also struggle to identify optimal system instructions for different tasks. Therefore, Orca 2 [253] addresses this by promoting cautious reasoning, deciding which solution strategy to choose for a given task among direct answer generation, or "Slow Thinking" strategies (step-by-step, guess and check or explain-then-answer, etc.) and erasing specific system instructions during training. This involves (1) a solution strategy guided by the performance of Orca 1 [258], (2) writing task-specific system instructions corresponding to the chosen strategy to obtain teacher responses for each task, and (3) at training time, employing Prompt Erasing to replace the student's system instructions with generic ones vacated of details of how to approach the task, encouraging students to learn not just task solutions but also deeper reasoning abilities.

Instruction Tuning Domain SLMs for QA. Beyond instruction tuning for generic SLMs, tuning domain-specific SLMs is also crucial, as they provide specialized assistance where generic SLMs may underperform. Instruction-tuning generic SLMs can derive domain SLMs. We summarize some representatives in several domains. (1) In finance, Phogat et al. [282] transfer financial QA abilities from teacher LLMs such as GPT-4 [2] to specialized SLMs such as Phi-3-Mini [1], using datasets such as FinQA [58], ConvFinQA [59], and TATQA [458]. They train SLMs with Python programs created by the teacher model, which detail steps for financial reasoning, including concept comprehension, formula identification, entity extraction, and calculations. During inference, SLMs generate Python code that an external interpreter executes. (2) In the medical field, Guo et al. [127] enhance student SLMs, including domain-specific BioGPT (1.6B) [235] and general Llama 7B [344], by fine-tuning on enriched PubMedQA [166] data. This enhancement is achieved by generating new samples or rewriting existing ones using teacher LLMs, which include the highly knowledgeable GPT-4 and the relatively weaker ChatGPT. The best SLM, with under 1.6 billion parameters, achieves 75.4% accuracy, surpassing GPT-4's 74.4% in few-shot settings on the PubmedQA test sets. It demonstrates that LLMs effectively refine and diversify question-answer pairs, leading to enhanced performance in a significantly smaller model after fine-tuning. We report the detailed results of comparisons of instruction-tuned domain-specific language models for QA and larger language models on FinQA [58] and PubMedQA [166], as shown in Table 5.

Enhancing SLMs for Out-of-Domain Questions. One of the major advantages of LLMs is their strong comprehension and logical reasoning abilities, which SLMs often struggle to match due to their limited parameters, especially when handling unseen or out-of-domain questions. Various methods have been developed to address this limitation, including RAG and self-adaptive techniques.

(1) *RAG: Incorporating External Knowledge for Domain-Specific QA.* RAG addresses OOD questions by integrating external knowledge during inference, allowing models to access information beyond their pre-trained parameters. By retrieving relevant documents in real time, RAG enables SLMs to provide accurate answers on specialized topics. In the telecommunications domain, Gichamba et al. [114] use ColBERT as a dense retrieval system to fetch documents from technical datasets. By encoding queries and documents separately, ColBERT computes relevance scores, helping small models like Phi-2 and Falcon-7B retrieve precise technical information to answer complex telecom-related queries. *Rationale Ranking* [130] addresses answering unseen questions using smaller language models by integrating external explanatory contexts from retrieval systems with reasoning rationales from LLMs. This method involves ranking both the retrieved explanatory contexts and LLM-generated rationales using a scoring module, which then combines them to form a cohesive context. Consequently, this integrated approach enhances the SLMs' performance on unseen questions.
(2) *Self-Adaptive Techniques: Enhancing Model Adaptability with Self-Generated Pseudo Labels.* Fine-tuning, while effective in adapting domain knowledge, can be impractical in realistic scenarios where labeled datasets are scarce. To overcome this, self-adaptive techniques employ self-generated pseudo labels to activate specific aspects of the target tasks, thereby enhancing model adaptability [319, 353]. Test-time Self-Adaptive Small LMs (*T-SAS*) [159] first stochastically generates multiple answers for an unlabeled question. The most plausible answer is then selected via majority voting to enhance pseudo-label accuracy, serving as a pseudo-label for training during test time.

Comparison between LLMs and SLMs for QA. When comparing LLMs such as GPT-4 [2] or BLOOM-175B [184] with fine-tuned SLMs in QA tasks, the benefits of SLMs are clear. LLMs, while versatile across multiple domains due to extensive pre-training, are computationally demanding, making them less ideal for resource-limited settings. SLMs, however, when fine-tuned for specific domains, often match or exceed the performance of larger models within those specialties. The tradeoff is between large-scale models' generalization and small-scale models' specialization: LLMs handle diverse domains but may need additional techniques such as knowledge injection for domain-specific queries. In contrast, domain-specific SLMs, though less flexible, provide higher accuracy and more relevant responses, making them ideal for edge deployments where computational resources are scarce but domain precision is crucial.

4.1.2 SLM Applications in Coding. The adoption of SLMs for coding offers an alternative to LLMs due to their lower computational needs and potential for domain-specific tuning. Despite LLMs' proficiency in code generation and programming support, SLMs are advantageous for their faster inference, reduced operational costs, and suitability for real-time environments where rapid responses are crucial. Representative works are discussed next. The Phi series [1, 158, 204] showcases SLMs' evolution in coding tasks. For instance, Phi-1 [122], a Transformer with 1.3B parameters, specializes in basic Python coding and achieves notable scores in benchmarks such as HumanEval [122], which includes 164 programming problems. Subsequent models, Phi-1.5 and Phi-2, have enhanced these capabilities, while Phi-3 demonstrated SLMs' potential to rival larger models [1]. The latest model, Phi-3.5-mini, with 3.8B parameters, excels in long-context tasks using advanced fine-tuning and optimization techniques, performing comparably to larger models such as Llama-3.1-8B-instruct [96] and surpassing smaller ones like Gemma-2 [339].

Another avenue of development is the fine-tuning of general-purpose SLMs for coding tasks [24, 123, 231, 299, 337]. For instance, CodeLlama models [299], derivatives of Llama 2 [345], undergo a rigorous fine-tuning process on domain-specific datasets, enhancing their proficiency in specific programming languages such as Python. They are trained to handle tasks such as syntax error detection, code suggestion, and infilling, where they learn to predict and complete missing parts of the code. This specialized fine-tuning improves their ability to interpret and execute detailed programming instructions, making them highly effective in real-time code editing environments [299]. CodeGemma models [337], stemming from Google DeepMind's Gemma framework, also exhibit a focused approach to enhancing coding capabilities through fine-tuning. These models are specifically engineered for high-performance code generation and infilling, underpinned by extensive training on a vast corpus of over 500 billion to 1 trillion tokens, predominantly consisting of code. This comprehensive dataset enables CodeGemma models to excel in mathematical reasoning and complex problem-solving within code contexts, setting new benchmarks in latency-sensitive applications such as real-time IDE support and automated code reviews [337].

Comparison between SLMs and LLMs on Coding. Table 6 provides a comparative analysis of SLMs and LLMs on coding benchmarks HumanEval [52] and MBPP [21]. Insights include: (i) Small SLMs (1.3B–3.8B Parameters) like Phi-3.5-mini [348] achieve high scores, demonstrating the efficacy of small models. Mid-sized SLMs (6.7B–9B Parameters), such as DeepSeek-Coder 6.7B [123] and Llama 3.1 8B [96], show improved performance, indicating that larger model sizes and enhanced training contribute to better accuracy. Large models (33B and above) like Llama 3.1 405B [96], GPT-4o [270], and Claude 3.5 Sonnet [16] excel, supporting the idea that bigger models generalize better across diverse coding tasks; (ii) There's a notable tradeoff between computational efficiency and performance, with larger models requiring more resources, impacting their practical deployment in constrained environments; and (iii) Specialized training and fine-tuning, as used in models like DeepSeek-Coder [123], are crucial for excelling in coding tasks, though such models may not handle complex requests as effectively, highlighting the versatility of general SLMs for broader applications.

Table 6. Performance Comparison Between SLMs and LLMs in Coding Benchmarks

Model	Size	HumanEval	MBPP
DeepSeek-Coder [123]	1.3B	65.2	49.4
CodeGemma [337]	2B	37.8	49.2
Gemma 2 [339]	2B	17.7	40.2
Phi-3.5-mini [348]	3.8B	62.8	69.6
DeepSeek-Coder [123]	6.7B	78.6	65.4
CodeGemma [337]	7B	60.4	55.2
Llama 3.1 [96]	8B	66.5	69.4
Gemma 2 [339]	9B	61.0	69.3
GPT-3.5 Turbo	-	68.0	71.2
DeepSeek-Coder [123]	33B	79.3	70.0
Llama 3.1 [96]	70B	80.5	75.4
Llama 3.1 [96]	405B	89.0	78.8
GPT-4o OpenAI [270]	-	90.2	81.4
Claude 3.5 Sonnet [16]	-	92.0	76.6

All models listed are chat or instruct versions, and performance are sourced from respective research papers or technical reports [96, 123, 299, 337, 348].

4.1.3 SLM Applications in Recommender Systems. Recommender systems are essential in various online services, helping to manage information overload and meet users' personal needs. SLMs enhance recommendation systems by (1) addressing the cold start problem; (2) reducing popularity bias; (3) improving long-term planning; (4) serving as personalized recommenders; and (5) acting as content encoders. These applications show the versatility and effectiveness of SLMs in boosting performance and personalization in recommendations. Next, we introduce the details.

SLM for System Cold Start Problem. Traditional recommendation systems, which utilize historical user–item interactions such as clicks, purchases, and ratings to learn representations and match items to users, fail in scenarios lacking any user–item interactions, known as the cold-start recommendation problem, often occurring in start-up businesses [296]. Although LLMs address this with in-context learning, their slow and costly inference restricts real-time use. Thus, *PromptRec* [387] explores using SLMs as in-context recommenders for recommendation system cold-start problems. However, SLMs often struggle without emergent context-learning abilities. To overcome this, SLMs are enhanced by pre-training on relevant corpora, using an improved C4 corpus subset [291], and by developing training prompts for different domains, enhancing cold-start performance. Results show that enhanced SLMs like BERT-mini [86], with 11.3M parameters, achieve BERT-large's performance in cold-start scenarios, with only 17% of BERT-large's inference time. Similarly, many studies have addressed the cold-start problem by leveraging BERT [135, 268, 446, 463]. For example, *ADLRS* [135] employs BERT to convert web-crawled item profiles into vectors that highlight key aspects, aiding recommender systems in acquiring essential initial information.

SLM for Mitigating Popularity Bias. Popularity bias in recommender systems, marked by discrepancies between item popularity in training datasets and the real world, often stems from using closed-loop datasets with limited information. Recent LLMs leverage their broad open-world knowledge to better reason about user–item interactions [211, 219], reducing this bias by providing recommenders with more extensive item details. Using the CoT prompting, LLMs decompose complex tasks into intermediate reasoning steps, enhancing understanding of user behavior and interests. However, LLMs' high resource demands limit their practical use. To overcome this, the Step-by-step Knowledge Distillation Framework for Recommendation (*SLIM*) [375] distills LLM reasoning capabilities into SLMs, keeping just 4% of the original parameters, transitioning from ChatGPT to Llama 7B [344]. SLIM uses detailed LLM templates to extract reasoning steps and streamlined templates for fine-tuning, enabling SLMs to improve recommender systems by better reasoning on richer item information.

SLM for Long-term Planning. Traditional recommender systems focus on optimizing immediate user responses, often maximizing short-term gains but overlooking long-term engagement. This can trap users in echo chambers and filter bubbles [108, 374]. To tackle this, integrating planning capabilities into recommendations to balance immediate and long-term outcomes is vital. LMs, with their extensive knowledge and reasoning abilities, are expected to enhance planning capabilities. *BiLLP* [316] adopts a hierarchical learning approach with macro- and micro-learning phases. In macro-learning, a Planner and a Reflector, both as SLM instances like Llama-2-7B [345], operate; the Planner forms long-term plans using high-level experiences, while the Reflector updates plans based on past actions. Micro-learning uses an SLM-based Actor-Critic mechanism for personalized planning, with the Actor implementing plans and the Critic assessing actions for long-term benefits.

SLMs as a Personalized Recommender. Generative language model-based recommender systems require integrating user knowledge, typically achieved through fine-tuning. Fine-tuning techniques like LoRA [145] can incorporate extensive knowledge across all users by training an external module with a small number of parameters **A** and **B**, but this approach often overlooks individual user preferences. To address this, *RecLoRA* [459] utilizes Vicuna-7B [63] to integrate personalized knowledge into SLMs/LLMs tailored for recommendation tasks, as illustrated in Figure 14. Specifically, RecLoRA maintains a set of parallel, independent LoRA weights $(\mathbf{A}_i, \mathbf{B}_i)$, allowing for the customization of language model parameters to match individual user preferences more effectively.

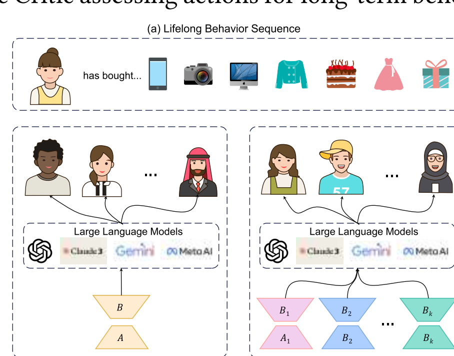


Fig. 14. The illustration of lifelong behavior sequence and personalized low-rank adaption (LoRA) for recommendation [459].

SLM as a Content Encoder. Language models, particularly when deep, provide an effective starting point for fine-tuning on downstream tasks. In news recommendation systems, the representational capability of a model significantly impacts performance. Consequently, many news recommender

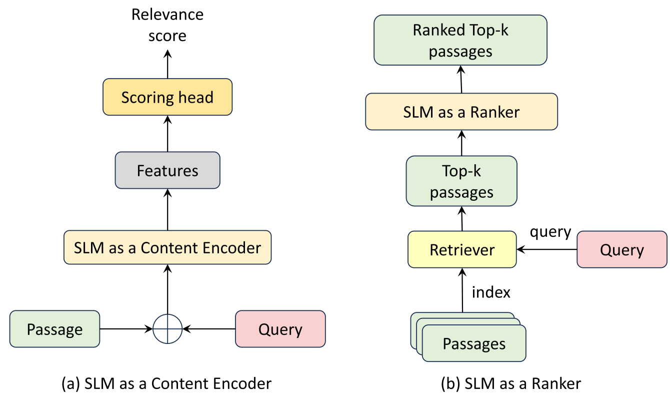


Fig. 15. Roles of SLM in web search.

systems now employ language models fine-tuned on specific datasets as text encoders. For example, Wu et al. [384] conduct pioneering work using a pre-trained language model to enhance large-scale news recommender systems by substituting traditional news encoders with a BERT model [86]. However, BERT may struggle to capture content as it is pre-trained on limited data. Therefore, *ONCE* [219] proposes using Llama-2-7B [345] as an encoder to overcome the limitations of BERT in content-based recommendations. Additionally, the study explores the synergistic use of LLMs in recommendation systems, finding that SLMs optimized with LoRA [145] outperform the recommendation results of systems assisted by generic LLMs such as ChatGPT.

4.1.4 SLM Applications in Web Search. Web search systems, involving retrieval and ranking, face challenges due to the diverse web documents and search queries. Traditional keyword-matching methods often fall short because of phrasing variations and the long-tail distribution of queries and content, complicating accurate semantic inference. Effective integration of retrieval and ranking models is also crucial. Language models, serving as content encoders, help overcome semantic challenges through their language understanding from pre-training [65, 92, 365]. Joint training of retrieval and ranking models addresses integration, with SLMs ranking retrieved documents and acting as re-rankers. Additionally, SLMs serve as rewriters in scenarios requiring enhanced query understanding. Thus, in web search, SLMs fulfill three roles: (1) *content encoder*, (2) *ranker*, and (3) *rewriter*, as depicted in Figure 15. Next, we give details.

SLM as a Content Encoder. Text embeddings are vector representations that encode semantic information, widely used in retrieval; SLM-based dense retrieval utilizes pre-trained deep language understanding to effectively tackle semantic challenges. *H-ERNIE* [65] employs a hierarchical model that encodes queries and documents at multiple granularity—character, word, and phrase—to improve specificity and relevance in web search results by aggregating finer details into coarser layers, addressing issues like ambiguous queries. **Implicit Interaction (I^3)** [92] uses BERT [86] as a content encoder, generating implicit pseudo-queries from passages to enable high online efficiency with offline caching of passage vectors. However, ERNIE- and BERT-style models overlook advancements in SLMs such as context length extension [299]. Thus, Peng et al. [279] employ LLaMa-7B [344] and Vicuna-7B [63] as semantic encoders for embedding retrieval, demonstrating improved performance through soft prompt tuning. *CoCondenser* [110] addresses sensitivity to noisy data and large batch requirements during dense retriever training. Using the Condenser architecture with Transformer blocks, the model condenses information into dense vectors effectively.

SLM as a Ranker. The reranking task improves the order of multiple candidates to enhance retrieval quality because rerankers are more accurate than embedding retrievers. *InPars (Inquisitive*

Parrots for Search) [37] employs the T5 base 220M [291] as a re-ranker to enhance the BM25 retriever [298]. Initially, BM25 selects 1 K candidates, re-ranked by a fine-tuned T5 model (monoT5) adapted as a binary classifier to assess document-query relevance. Training data, generated by GPT-3 [38], formulates queries and selects random negative examples. Experiments show the monoT5-enhanced retriever significantly outperforms GPT-3; for example, it achieves a 0.3599 MAP score on the TREC-DL 2020 dataset [72], surpassing GPT-3's 0.3163.

SLM as a Rewriter. Queries to the retriever, typically just a few keywords, may reveal a knowledge gap between the query and the knowledge needed for effective retrieval, thus limiting performance. To address this, the *"rewrite-retrieve-read"* framework [238] uses T5-large [291] to bridge the knowledge gap in queries by rewriting them for more effective retrieval. This rewriter, trained via reinforcement learning with downstream LLM performance as a reward, outperforms general LLM rewrites. For example, it achieves a 45.97 F1 score on HotpotQA, surpassing the generic LLM's 43.85 F1 score.

4.1.5 SLM Applications in Mobile-Device. The use of cloud-based LLMs on devices raises privacy concerns, and their large size limits real-time responses in urgent scenarios such as medical emergencies. To overcome these issues, researchers are creating smaller, domain-specific models (SLMs) that offer accurate results and suit mobile use. This subsection discusses SLM applications on mobile devices, focusing on three aspects: (1) software API calls, (2) mobile control, and (3) basic NLP applications.

SLM for Tool Learning. Integrating LLMs with APIs enhances capabilities but incurs high training costs, prompting a shift to smaller, task-specific models that cut costs but risk errors. In response, *Octopus* [54] uses a diverse dataset from over 30 K APIs and curriculum learning [222] to improve API function accuracy. This method boosts API performance in models like Codellama-7b [299] and Google's Gemma series [338]. *PhoneLM-1.5B-Call* [422] is fine-tuned on DroidCall [394] datasets and achieves comparable performance compared to GPT-4o-mini [269]. *α-UMI* [314] employs SLMs as planners, callers, and summarizers within multi-agent systems, outperforming a single LLM in tool uses.

SLM for Mobile Control. LM agents facilitate user-device interactions through taps, gestures, and text, automating tasks and enhancing user hands-free convenience. Unlike traditional developer-based approaches that require extensive developer effort to design interfaces and translate commands into API calls, LMs offer scalable automation via GUI-based text content. *MobileAgent* [90] integrates instructions and Standard Operating Procedures (SOP) to improve SLMs for mobile control. As shown in Figure 16, it processes goals (e.g., booking a dental appointment) by analyzing screens, queries, prior actions, and UI elements, forming prompts to generate outputs and execute actions (e.g., selecting text, XPath). Fine-tuning Qwen-7B [24] on AIA medical data, it outperforms GPT-4 [2] on AitW [297], a key mobile benchmark, without extra inference costs. Carreira et al.

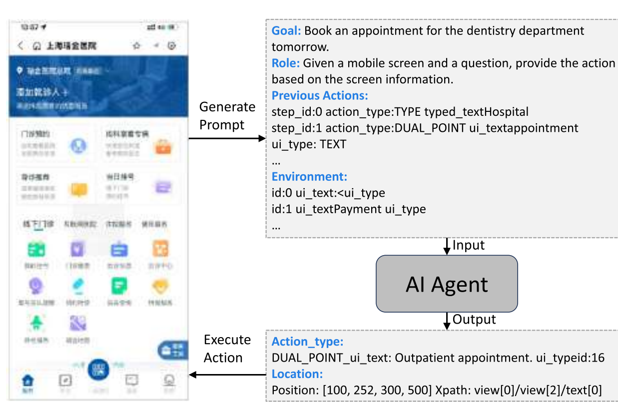


Fig. 16. An example workflow for an automated execution tool [90]. The screenshot on the left is taken from [90].

[43] run a small offline model on mobile devices, fine-tuned with ChatGPT-3.5 data, enabling tasks like calls and web searches. RedPajama-INCITE-Chat-3B-v1 Computer [70], selected for its size and chat features, uses native code and quantization, performing well on 6 GB and 4 GB Android devices.

AutoDroid [381] improves Android app interactions via GUI automation. Figure 17 shows LLM-powered task automation (e.g., setting a laundry reminder for Aug 17) in four steps: (1) click "New Event," (2) enter "laundry" in "Title," (3) click "Save," and (4) finish. Using Vicuna-7B and app-specific knowledge, AutoDroid generates privacy-filtered prompts for tasks. On its DroidTask benchmark, it outperforms GPT-3.5 (34.7%) and GPT-4 (54.5%) with 57.7% accuracy. *M4 (composable mobile foundation model)* [428] introduces a 9.2B parameter model (7.5 GB memory) for mobile AI tasks, managed by the OS and hardware. Currently limited to high-end devices, its applicability will expand with increasing mobile memory/storage. These works highlight customizing smaller, domain-specific SLMs to address memory limits while preserving functionality in mobile environments.

Fig. 17. An illustration of Vicuna-7B-powered mobile task automation [43] shows a user asking to be reminded about doing laundry on Aug 17. The figure is taken from [43].

SLM for Basic NLP Applications on Devices. Performing basic NLP tasks such as text rewriting directly on the device can enable personalization while ensuring privacy. The sparse annotation on devices is a challenge. Qin et al. [285] utilize self-supervised data selection and synthesis for on-device fine-tuning, leveraging sparse annotations and limited storage effectively. This approach, demonstrated in Figure 18, employs the Llama-3B model [344] and the LoRA fine-tuning method [145], enhancing personalization by efficiently managing data through metrics including embedding entropy and domain-specific scores. In mobile text rewriting, Zhu et al. [462] train the compact Palm 2-XXS model [15] using data generated by the larger Palm 2-L to ensure user privacy and accommodate device constraints. On its new benchmark, MessageRewriteEval, Palm 2-XXS achieves a BLEU score of 34.59, outperforming LLaMa-7B (16.65) [344]. Tests on the Samsung S23 show lower latency (29 tokens/s) than a 4-bit LLaMa-7B on a MacBook M1 Pro (18-22 tokens/s), proving its mobile efficiency for text rewriting.

Insights: We draw several key insights from the development of task-specific SLMs:

- There is considerable potential in enhancing the efficiency and effectiveness of small models by integrating self-adaptive techniques with further fine-tuning and optimization of RAG-based methods.
- The growing relevance of SLMs in coding highlights their cost-effectiveness and efficiency as alternatives to LLMs, providing quick processing and easy fine-tuning for specialized tasks; while LLMs handle complex tasks well, SLMs, optimized and fine-tuned on specific data, are increasingly essential in resource-limited settings.
- SLMs significantly enhance recommendation systems due to their robust generalization, reasoning abilities, and in-context learning, addressing key challenges such as cold-start problems and distribution biases. They support long-term planning, replace traditional encoders, and use parallel low-rank parameters to inject personalized user knowledge effectively.

—SLMs play a crucial role in web searches such as document encoding, text reordering, and query rewriting, often outperforming LLMs through techniques such as SFT, soft prompt tuning, unsupervised contrastive loss, and reinforcement learning, thereby enhancing adaptability and efficiency.
—SLMs are utilized on mobile devices primarily for privacy and memory constraints, with applications in API calls and mobile control; they are typically developed by generating data with LLMs and fine-tuning with SLMs, or by using local SLMs to handle privacy with LLMs boosting performance, and their training involves innovative techniques like learning from data streams and managing non-IID time series data.

4.2 SLM Deployment on Mobile and Edge Devices

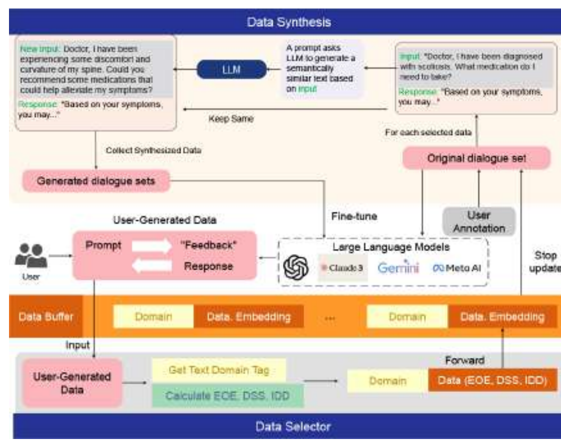


Fig. 18. Overview of fine-tuning SLMs with synthesized and user data [285]. Data synthesis generates semantically similar text via prompts, creating dialogue sets. Data selection processes user data, tags domains, and calculates metrics (EOE, DSS, IDD). Selected data fine-tunes SLMs with user annotations. The iterative framework refines SLMs through continuous data generation and selection.

On-device applications benefit uniquely from the memory-saving efficiency and rapid runtime performance of SLMs, which offer advantages over LLMs. However, devices with extremely limited resources may still struggle with the parameter sizes of SLMs. To ensure both memory and runtime remain within acceptable ranges while still maintaining performance, it is crucial to integrate technologies that facilitate the deployment of SLMs on resource-constrained devices. The primary challenge for memory-efficient technologies arises from the inherent size of the SLMs and their associated caches. To address this, we survey existing works focused on compressing SLMs and their caches. Additionally, the large size of models significantly impacts runtime efficiency due to the extensive computing workload and potential weight transfers between the memory buffer and RAM/GPU memory. Other challenges include switching the Mixture of Experts between CPU and GPU memory and managing resource scheduling when deploying SLMs across multiple local devices. Therefore, in this subsection, we review representative works that address these challenges under two aspects: *memory efficiency optimization* and *runtime efficiency optimization*, as systematically compiled in Table 7.

4.2.1 Memory Efficiency Optimization. Memory efficiency involves minimizing the memory usage of both the model and the KV cache during deployment. This is typically achieved through model compression techniques such as quantization [212, 292, 427, 452], the cache of MoE experts [421], and KV cache compression [168].

Compression on model parameters. Quantization, a common method for deploying SLMs, lowers numerical precision to significantly save memory while preserving accuracy. We detail quantization strategies in Sections 2.3.3 and 3.5, focusing here on representative works for edge devices. *EDGE-LLM* [427] adapts LLMs for edge devices using a Layer-wise Unified Compression (LUC) method that combines layer-specific pruning and quantization to reduce computational demands and an

Table 7. On-Device Deployment Optimization Techniques

Aspect	Representative Work	Key Point
Memory Efficiency Optimization	EDGE-LLM [427]	Edge LLMs use LUC and adaptive tuning for efficiency.
	LLM-PQ [452]	Optimize quantization and layer partitioning for complex setups.
	AWQ [212]	Preserve key weights based on activation distribution.
	MoE-I^2 [409]	Prune less important experts in MoE.
	MobileAIBench [260]	Evaluation.
	EdgeMoE [421]	Load experts on activation, tripling memory savings.
	GEAR [168]	Enhance KV cache quantization by integrating error-reduction techniques.
	DMC [263]	Adaptively compress KV cache, optimizing storage efficiency.
	Transformer-Lite [196]	Optimize KV cache to reduce redundancy and memory use.
	LLMaaS [424]	LLMaaS manages apps via chunk-wise KV cache optimization on mobiles.
Runtime Efficiency Optimization	mllm-NPU [400]	On-device NPU (neural processing units) to reduce prefill latency.
	COST-EFF [312]	Distill a multi-exit model from the original PLM.
	LLMCad [399]	Use SLM for fast token generation and cloud verification.
	EdgeMoE [421]	Predict expert needs, boosting inference speed and reducing latency.
	LinguaLinked [451]	Optimize dataflow and load, enhancing multi-threading efficiency.

Adaptive Layer Tuning and Voting scheme to optimize memory use while ensuring performance. Meanwhile, *LLM-PQ* [452] addresses quantization and model layer partitioning for heterogeneous clusters, incorporating a Cost Model and an Indicator Generator to optimize bit-width assignment and layer partitioning through integer linear programming, enhancing quantization for complex computational setups. **Activation-aware Weight Quantization (AWQ)** [212] is a hardware-friendly, low-bit, weight-only quantization method for on-device LLMs, preserving essential weights based on activation distribution to minimize quantization loss. *MoE-I*^{2} [409] prunes less important experts and applies low-rank decomposition to the remaining experts to further optimize efficiency.

Cache of MoE Experts. Beyond standard quantization, which reduces storage for all model parameters, another strategy involves caching a mixture of experts (MoE). Driven by the fact that memory storage is more cost-effective and scalable than computing capacity, the MoE architecture [157] boosts performance while minimizing computational costs by activating only portions of the LLM as needed. However, this approach incurs significant memory overhead, making it impractical for edge device memory constraints. For example, Switch Transformers [102], with 32 experts per layer, require 54 GBs of memory for inference, exceeding the capacity of most edge devices. Yi et al. [421] note that in the Switch Transformers model, although most of the model weights (86.5%) are attributed to experts, these weights account for only a small fraction (26.4%) of the computations. To address this, *EdgeMoE* [421] introduces a method where experts are loaded into an expert memory buffer only when activated, achieving approximately 3× memory savings compared to the baseline where all weights are held in memory.

KV Cache Compression. When serving LMs for inference, using a KV cache is common practice to avoid intensive recalculations and speed up generation [283]. However, cache memory consumption escalates with model size and sequence length, posing a challenge for edge devices. To manage this, offloading techniques transfer KV caches to CPU memory [13, 315], although this can introduce significant overhead due to the switching costs between GPUs and CPUs. Token dropping compresses cache size by keeping only key tokens, often identified by low attention scores [113, 225, 449]. However, this method struggles with complex tasks, especially at high compression levels, due to increased estimation errors in compressed KV values. *GEAR* [168] addresses these issues by enhancing KV cache quantization with error-reduction techniques, including: (i) quantizing caches of similar magnitudes to ultra-low precision, (ii) using a low-rank matrix for efficient quantization residual approximation, and (iii) employing a sparse matrix for correcting outliers. This approach separates coherent from incoherent approximation errors, enabling near-lossless KV cache compression and achieving up to 2.29× peak memory reduction. Besides, **Dynamic Memory Compression (DMC)** [263] adaptively compresses the KV cache by either adding new key and value representations directly or blending them with the top cache item using a weighted average. *Transformer-Lite* [196] tackles the redundancy of storing the KV cache twice in model inputs and outputs, which increases memory use. It optimizes storage by allocating a large tensor based on the maximum sequence length needed for inference. Sub-tensors are then created from this main tensor at different address offsets to serve as input and output KV caches, allowing direct writing to the correct locations during inference and removing extra copying steps. *LLMaaS* [424] introduces LLM as a Service for mobile devices, managing all apps through LLMS. This system uses chunk-wise KV cache compression and swapping, enabling efficient context switching within memory constraints. By segmenting the KV cache into independently compressed and swapped chunks, LLMS balances memory use and I/O bandwidth better than token-level or context-level management.

However, all these compression schemes trade off memory for computation: decompressing the KV cache (e.g., upscaling 4-bit values back to full precision or reconstructing dropped tokens) adds extra operations and latency. In fact, GEAR notes that its low-rank/sparse reconstruction "incurs extra latency," and Transformer-Lite explicitly targets reducing "dequantization overhead" from low-bit caches. Similarly, the LLMaaS system even pipelines recompute when loading compressed chunks.

4.2.2 Runtime Efficiency Optimization. The goal of decreasing computing workload aligns with enhancing memory efficiency through methods such as quantization, as mentioned in the previous section. Decreasing model weight precision or reducing the number of weights naturally lowers latency. Other runtime efficiency techniques of minimizing inference latency involve, reducing prefill latency, early exits, large and small model collaboration, decreasing switching time in MoE, and reducing latency in distributed SLMs.

Reducing Prefill Latency. mllm-NPU [400] is the first LLM inference system that leverages on-device NPU (neural processing units) to reduce prefill latency and energy consumption. It incorporates a chunk-sharing graph, shadow outlier execution, and out-of-order subgraph execution to enhance NPU offloading efficiency. Experiments have shown mllm-NPU's superior performance benefits, including up to 43.6× speedup and 59.5× energy savings.

Dynamic Early Exits. A decoupled runtime saving technique is dynamic early exits. Originating from BranchyNet [341], which introduces exit branches after specific convolution layers in the CNN model, this concept has been adapted for PLMs as Transformer layer-wise early exiting [396]. Early exiting enables dynamic acceleration during inference and reduces temporal overhead by allowing exit without passing through all model layers. To address the inconsistency issue arising from exiting at different layers, *COST-EFF* [312] distills a multi-exit model from the original PLM.

Large and Small Model Collaboration. Model collaboration, deploying SLMs on devices with cloud-based LLM support, enhances runtime efficiency. LLMs will increase latency when directly deployed via mobile engines like llama.cpp due to a large number of computing operations. *LLMCad* [399] addresses this by using a real-time, memory-resident SLM for simple tokens such as determiners and punctuation. The SLM generates tokens, while a cloud-based LLM verifies and corrects them, speeding up the process. LLMCad enhances token generation up to 9.3× by pairing the memory-resident SLM, Llama 2 7B, with the cloud-based LLM, Llama 2 13B, cutting latency from 16.2 to 3.9 seconds on Xiaomi Pro for TruthfulQA tasks [213].

Reducing MoE Switching Time. To reduce latency in MoE architectures caused by frequently switching experts in limited device memory, *EdgeMoE* [421] enhances runtime efficiency by preemptively predicting which expert will be needed, based on the observed long-tail distribution of unbalanced expert activations. It utilizes a statistical model, built offline, to estimate expert activation probabilities in transformer layers from previous activations. During inference, EdgeMoE preemptively loads the most likely needed expert, accelerating inference by 1.11× to 2.78× and significantly reducing latency. For instance, in a switch transformer model with 8 experts, latency drops from approximately 0.7 s to 0.3 s, outperforming the baseline method that preloads experts based on hit ratios.

Reducing Latency in Distributed SLMs. Distributing an SLM across smaller devices reduces the need for extensive model compression while preserving accuracy. However, this approach faces challenges that incur latency, such as managing diverse device capabilities, handling data dependencies between model segments, and adapting to dynamic resource availability. To address these issues, *LinguaLinked* [451] addresses these issues by optimizing model assignment to match device capabilities and minimize data transmission, implementing runtime load balancing to redistribute tasks and prevent bottlenecks, and enhancing communication for efficient data exchange between segments. With multi-threading, the system improves, achieving acceleration rates between 1.73× and 2.65× for both quantized and full-precision models.

Insights: We draw several key insights from the deployment of SLMs:

- Model size remains a bottleneck for both memory and runtime efficiency. A common solution is model quantization, which reduces model precision to save memory and lessen computing workload, thereby boosting inference speed [212, 227, 260, 292, 427, 452]. Similarly, KV cache compression also helps achieve these efficiency gains [168, 196, 263, 424].
- Mixture of Experts (MoE) is commonly used in SLMs to enhance performance using the same computing resources, but it results in increased memory usage. To address this, only activated experts are loaded into the memory buffer while the majority are stored cold on disk. However, the cost of switching can slow down inference. Designing a preemptive expert pre-load strategy could therefore accelerate the inference [421].
- Model collaboration between local SLMs and cloud-based LLMs enhances both memory and runtime efficiency by using smaller models on local devices, which are then verified by cloud LLMs to ensure performance is maintained. Using SLMs locally reduces memory usage and shortens the inference time from the local model. However, internet latency and delays in cloud LLM inference can still introduce latency. Verifying SLM outputs every N tokens using LLMs can effectively mitigate this latency [399].
- One deployment approach involves deploying SLMs/LLMs across multiple trusted local devices to maintain original performance while only loading a fraction of the model weights. However, this method can incur latency due to varying device capabilities and resource scheduling challenges. To address these issues, optimizing model assignment to align with device capabilities and minimizing data transmission are effective strategies [451].

5 Generic and Domain-Specific SLMs

This section investigates SLMs (with fewer than 7 billion parameters) in both general and specific domains. It details the methods of obtaining these SLMs the datasets, and the evaluation tasks, exploring the techniques for acquiring SLMs through compression, fine-tuning, or training from scratch. Additionally, we summarize the representative SLMs as detailed in Tables 8 and 11.

5.1 Generic-Domain SLMs

Overview. SLMs, with fewer parameters than LLMs, enhance computational efficiency in pre-training, fine-tuning, and inference, reducing memory and energy demands—crucial for resource-limited environments. Their compact, localized nature boosts privacy, personalization, and response times, making them ideal for low-power edge devices. Therefore, SLMs are attracting increasing attention, and various models are being developed. Table 8 summarizes current representative generic-domain 42 SLMs/SLM families. Although all chosen SLMs have similar architectures, they

Table 8. High-Level Overview and Training Details of Generic-Domain SLMs

Model	#Params	Date	Paradigm	Domain	Training Datasets	Training Techniques
PhoneLM [422]	0.5B; 1.5B	2024.11	Pre-train	Generic	DCLM-baseline [195], StarCoderData [199]; OpenWebMath [274], Dolma-algebraic and Dolma-arXiv [322]	RoPE, MHA, Gated FFN, RMSNorm, ReLU, FSDP, Flash Attention 2, ZeRO
Llama 3.2 [7]	1B; 3B	2024.9	Pre-train	Generic	no release (9T tokens)	GQA, SiLU, Multilingual Text and code, Shared embedding, Pruning, Distillation, SFT, RLHF, RS, DPO
Qwen 1 [24]	1.8B; 7B; >	2023.12	Pre-train	Generic	No release	MHA; RoPE; SwiGLU; RMSNorm
Qwen 1.5 [24]	0.5B; 1.8B; 4B; 7B; >	2024.2	Pre-train	Generic	No release	MHA; RoPE; SwiGLU; RMSNorm; Multilingual support
Qwen 2 [407]	0.5B;1.5B; 7B; >	2024.6	Pre-train	Generic	No release	GQA; RoPE; SwiGLU; RMSNorm; Multilingual support
Qwen 2.5 [407]	0.5B; 1.5B; 3B; 7B; >	2024.9	Pre-train	Generic	No release	GQA; RoPE; SwiGLU; RMSNorm; Multilingual support; Larger corpus
Gemma [338]	2B; 7B	2024.2	Pre-train	Generic	Unknown	MHA, RoPE, $GELU_{tanh}$
Gemma 2 [339]	2B; >	2024.7	Pre-train	Generic	Unknown	GQA; RoPE; $GELU_{tanh}$; Alternating Local and Global Attention; Logit Soft-Capping; RMSNorm for Pre- and Post-Normalization
SmolLM [10]	135M; 360M; 1.7B	2024.7	Pre-train	Generic	Smollm-corpus [28]	GQA, trapezoidal LR scheduler
H2O-Danube3 [281]	500M; 4B	2024.7	Pre-train	Generic	Unknown	Three different training stages with different data mixes
Fox-1 [340]	1.6B	2024.6	Pre-train	Generic	Unknown (3T tokens)	GQA; Deep architecture
Rene [115]	1.3B	2024.5	Pre-train	Generic	Dolma-1.7 [322]	Mamba-2 layers, sliding-window attention (SWA)
MiniCPM [146]	1.2B; 2.4B	2024.4	Pre-train	Generic	Dolma [322]; C4 [291]; Pile [87]; stack [177]; StarCoder [199]; UltraChat [89]; OssInstruct [378]; EvolInstruct [397]	WSD learning rate scheduler
CT-LLM [94]	2B	2024.4	Pre-train	Generic	MAP-CC	Chinese, MHA, RoPE, SwiGLU, RMSNorm
Phi-1 [122]	1.3B	2023.6	Pre-train	Coding	CodeTextBook [122][a]	MHA, $GELU_{tanh}$, RoPE, FlashAttention
Phi-1.5 [204]	1.3B	2023.9	Pre-train	Generic	CodeTextBook [122]; Synthetic Datasets (20B)	MHA, $GELU_{tanh}$, RoPE, FlashAttention, Deep ZeRO Stage 2
Phi-2 [158]	2.7B	2023.12	Pre-train	Generic	CodeTextBook [122]; Synthetic Datasets (20T)	MHA, $GELU_{tanh}$, RoPE, FlashAttention, Deep ZeRO Stage 2
Phi-3 [1]	3.8B; 7B; >	2024.4	Pre-train	Generic	Scaled-up dataset from phi-2	MHA, SiLU, RoPE, FlashAttention, Deep ZeRO Stage 2
Phi-3.5 [1]	3.8B; 4.2B; 6.6B	2024.4	Pre-train	Generic	more multilingual and long-text data	Multilingual; Vision; MHA, SiLU, RoPE, FlashAttention, ZeRO 2
OpenELM [246]	270M; 450M; 1.1B; 3B	2024.4	Pre-train	Generic	RefinedWeb [276], deduplicated PILE [109], partial RedPajama [70], partial Dolma v1.6 [322]	No biases in FC layers; Pre-norm: RMSNorm; Pos encoding: RoPE; Attention: GQA; FFN: SwiGLU; Tokenizer: LLaMA-style
MobiLlama [342]	0.5B; 0.8B	2024.2	Pre-train	Generic	LLM360 Amber[b]	GQA; SwiGLU; Parameter sharing; multi-modal

(Continued)

Table 8. Continued

Model	**#Params**	**Date**	**Paradigm**	**Domain**	**Training Datasets**	**Training Techniques**
MobileLLM [227]	125M; 350M	2024.2	Pre-train	Generic	Unknown (1 T tokens)	SwiGLU FFN, deep and thin architectures, embedding sharing, and GQA
OLMo [118]	1B; 7B	2024.2	Pre-train	Generic	Dolma [322][c]	SwiGLU; RoPE, Non-parameteric Layer Norm
TinyLlama [444]	1B	2024.1	Pre-train	Generic	SlimPajama [321] and StarCoder [199]	GQA, SiLU, FSDP, Flash Attention [76], xFormers [187]
StableLM [346]	3B; 7B	2023.4	Pre-train	Generic	RefinedWeb [276], RedPajama [70], the Stack [177], OpenWebText [116], OpenWebMath [274], and partial CulturaX [264]	MHA; SiLU; Fine-tuning; DPO; Self-knowledge; RoPE; LayerNorm; no Biases
StableLM 2 [27]	1.6B	2024.2	Pre-train	Generic		
Cerebras-GPT [87]	111M; 256M; 590M; 1.3B; 2.7B; 6.7B; >	2023.4	Pre-train	Generic	Pile [109]	MHA; GELU; Maximal Update Parameterization
Pythia [32]	14M; 70M; 160M; 410M; 1B; 1.4B; 2.8B; 6.9B; >	2023.4	Pre-train	Generic	Pile [109]	MHA; GELU; Flash Attention [77]; RoPE [324]; ZeRO [293]
BLOOM, BLOOMZ [184]	560M; 1.1B; 1.7B; 3B; 7.1B; >	2022.11	Pre-train	Generic	ROOTS [183] and 13 programming languages	MHA; $\text{GELU}_{\text{tanh}}$; ALiBi Positional Embedding [284], Embedding LayerNorm [83]
Galactica [336]	125M; 1.3B; 6.7B; >	2022.11	Pre-train	Scientific	Open-access scientific materials (106B tokens) but not released	MHA; GeLU; Learned Positional Embeddings
OPT [445]	125M; 350M; 1.3B; 2.7B; 5.7B	2022.5	Pre-train	Generic	Pile [109] and PushShift.io Reddit [25]	MHA; ReLU
XGLM [214]	1.7B; 2.9B; >	2021.12	Pre-train	Generic	CC100-XL	–
GPT-Neo [34]	125M; 350M; 1.3B; 2.7B	2021.5	Pre-train	Generic	Pile [109]	–
Megatron-gpt2 [317]	355M; 2.5B; >	2019.9	Pre-train	Generic	Wikipedia [86], CC-Stories [347], RealNews [435], OpenWebtext	–
MINITRON [259]	4B; >	2024.7	Distillation; Pruning	Generic	8 T tokens in Nemotron-4 [273]	LR WSD Scheduler
Orca 2 [253]	7B	2023.11	Distillation	Generic	Orca 2 dataset	LLaMA-2-7B based; prompt erasing
Orca [258]	13B	2023.6	Distillation		FLAN-v2 [229]	From ChatGPT and GPT4, Explanation tuning; Progressive Learning
MINIMA [439]	3B	2023.11	Distillation	Generic	Pile [109], Wudao [429], GitHub [70]	From Llama-2-7B, Zero2, Flash Attention, Optimal teacher size
Dolly-v2 [71]	3B; 7B; >	2023.4	Instruction tuning	Generic	Databricks-dolly-15k [71]	From pythia

(Continued)

Table 8. Continued

Model	#Params	Date	Paradigm	Domain	Training Datasets	Training Techniques
LaMini-LM [174]	61M-7B	2023.4	Distillation	Generic	LaMini instruction dataset	A collection of SLMs distilled from ChatGPT-generated 2.58M instructions.
Specialized FlanT5 [107]	250M; 760M; 3B	2023.1	Instruction Tuning	Generic (math)	GSM8K	Base model is FlanT5

[#]Params means parameter amounts. “>” indicates parameters larger than 7B.
[a]CodeTextBook includes stack v1.2, code contests, synthetic python textbooks and exercises.
[b]LLM360 Amber includes Arxiv, Book, C4, Refined-Web, StarCoder, StackExchange, and Wikipedia.
[c]Dolma includes Dolma’s CC, RefinedWeb, Star Coder, C4, PushShift API, Semantic Scholar, RedPajama, Flan, CC News, OpenWebMath, MetaWika, Wikipedia.

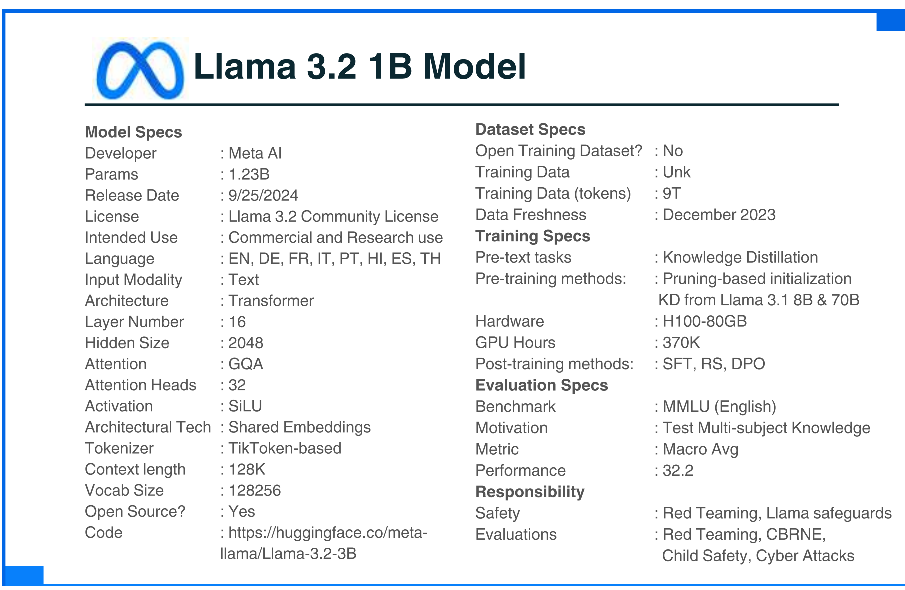


Fig. 19. Llama 3.2 1B model card.

vary in specific training datasets and techniques, with some datasets not being openly available. Taking the latest Llama 3.2 1B models [7] in Figure 19 as an example, its parameter size and use of filtered high-quality training data, pruning-based initialization, KD pre-training tasks, and training techniques such as SFT Rejection Sampling (RS), and DPO distinguish it from others.

5.1.1 Architecture Design. From Table 8, we observe several trends in component choices for SLMs:

(1) Recent SLMs frequently employ GQA in self-attention mechanisms because it can reduce computational complexity. GQA achieves this by sharing query representations across multiple heads while keeping key and value representations separate. This approach aligns with the goals of SLM to enhance efficiency without compromising functionality.
(2) The choice of activation function should balance model capability and efficiency. ReLU, known for its efficiency, introduces greater sparsity to the model, which facilitates faster

Table 9. Zero-Shot Performance of Various Language Models from Small to Large on Common Benchmarks

Model Size Range	Model	MMLU	HellaSwag	ARC (C)	PIQA	Winogrande
<1B	GPT-Neo 125M	26.0	30.3	23.0	62.5	51.8
	Tiny-Starcoder 170M	26.8	28.2	21.0	52.6	51.2
	Cerberas-GPT 256M	26.8	29.0	22.0	61.4	52.5
	OPT 350M	26.0	36.7	23.6	64.7	52.6
	Megatron-GPT2 345M	24.3	39.2	24.2	66.9	53.0
	LiteLlama	26.2	38.5	24.9	67.7	49.9
	GPT-SW3 356M	25.9	37.1	23.6	64.9	53.0
	Pythia 410M	27.3	40.9	26.2	67.2	53.1
	XGLM 564M	25.2	34.6	24.6	64.9	53.0
	Lamini-GPT-LM 0.59B	25.5	31.6	24.2	63.9	47.8
	MobiLlama 0.5B	26.5	52.5	29.5	72.0	57.5
	MobiLlama 0.8B	26.9	54.1	30.2	73.2	57.5
1B–3B	StableLM 2 1.6B	41.8	68.2	43.8	74.0	64.9
	Pythia 1B	24.3	44.7	33.1	69.1	53.3
	TinyLLaMA 1.1B	25.02	61.4	32.6	73.5	59.4
	OLMo 1B	24.23	62.9	34.5	75.1	59.9
	OLMo 1B (0724)	25.45	66.9	36.5	74.9	61.4
	Boomer 1B	25.4	31.6	22.3	58	51.0
	Pythia-Dedup 1B	24.3	49.6	29.1	70.2	54.0
	Falcon-RW 1B	25.4	63.1	35.1	74.1	61.9
	Cerebras-GPT 1.3B	26.7	38.5	26.1	66.8	53.6
	Lamini 1.3B	28.5	38.1	26.6	67.9	50.6
	OPT 1.3B	24.6	54.5	29.6	72.5	59.7
	GPT-Neo 1.3B	24.8	48.5	31.3	71.1	57.1
	Pythia-Deduped 1.4B	25.5	55.0	32.6	–	56.9
	MobiLlama 1.2B	24.8	63.0	34.6	–	62.0
	OpenELM 1.1B	25.3	64.8	32.3	75.6	61.7
	OLMo-1B-0724-hf 1.2B	25.5	66.1	32.3	75.1	61.7
	AMD-OLMo-1B 1.2B	24.9	63.6	33.7	75.6	61.6
	Gemma 2 2B	57.8	61.1	76.7	81.2	72.3
	Llama 3.2 1B	49.3	41.2	59.4	–	–
	Llama 3.2 3B	63.4	69.8	78.6	–	–
3B–7B	Phi-3.5-mini 3.8B	69.0	81.4	87.4	–	–
	Pythia 6.9B	–	63.8	44.1	75.2	60.9
	Falcon 7B	–	75.9	47.5	78.5	68.9
	LLaMA 1 7B	35.1	76.2	44.5	77.2	70.5
	LLaMA 2 7B	45.3	76.8	48.5	76.7	69.4
	MPT 7B	30.2	77.6	46.5	77.3	69.9
	RPJ-INCITE 7B	27.8	70.3	42.8	76.0	64.7
	OLMo 7B	61.1	76.4	48.5	78.4	58.0
>7B	LLaMA 2 13B	55.6	80.7	48.8	80.8	72.9
	LLaMA 2 70B	67.6	77.3	55.5	82.0	75.0
	Gemma 2 9B	70.6	87.3	89.5	86.1	78.8
	GPT-3.5	86.4	90.1	87.4	89.3	82.7
	Claude 2	89.3	92.1	88.0	91.5	85.5
	GPT-4	90.6	93.5	94.0	92.7	87.2

Data from *MobiLlama* [342], *OLMo/OLMoE* [118, 257], and public evaluations of GPT-3.5, Claude 2, and GPT-4.

coefficient calculations for inference acceleration. In contrast, SwiGLU's parameters are learned during training, allowing the model to dynamically adapt to diverse tasks and datasets, thereby enhancing model capabilities and establishing it as a state-of-the-art option.

Table 10. Comparison of MobiLlama 0.5B with Larger Models (1.2B–180B) in Terms of Efficiency and Resource Consumption on Low-End Hardware Devices [342]

Platform	Model	#Params	Avg Tokens/s	Memory (MB)	Battery /1kT (mAh)
RTX 2080Ti	GPT-4	>100B	N/A	N/A	N/A
	Claude 2	>100B	N/A	N/A	N/A
	Falcon	180B	<1.0	~3,20,000	N/A
	LLaMA 2	70B	~5.8[a]	~1,28,000	N/A
	Mixtral 8 ×7B	46.7B (MoE)	~10–20[b]	~90,000	N/A
	LLaMA 2	13B	~20.0	~11,000	~85
	Gemma 2	9B	~24.0	~8,000	~80
	Mistral	7.3B	~30.0	~20,000	~68
	Phi-2	2.7B	32.19	12,071	59.1
	MobiLlama Large	1.2B	50.61	6,254	18.9
	MobiLlama	0.5B	63.38	3,046	8.2
CPU i7	GPT-4	>100B	N/A	N/A	N/A
	Claude 2	>100B	N/A	N/A	N/A
	Falcon	180B	~0.4–1.1[c]	~7,00,000	N/A
	LLaMA 2	70B	~0.1–1.0	~2,56,000	N/A
	Mixtral 8 × 7B	46.7B (MoE)	~1–2	~90,000	N/A
	LLaMA 2	13B	~3.3	~8,400	~130
	Gemma 2	9B	~4.5	~5,500	~97
	Mistral	7.3B	~12.0	~3,500	~37
	Phi-2	2.7B	22.14	1,972	27.4
	MobiLlama Large	1.2B	29.23	1,163	10.8
	MobiLlama	0.5B	36.32	799	4.9
Snapdragon	Falcon/GPT-4/Claude 2	>70B	N/A	N/A	N/A
	Mixtral TurboSparse	47B (MoE)	11.7	24,000	N/A
	LLaMA 2	13B	N/A	N/A	N/A
	Gemma 2	9B	N/A	N/A	N/A
	Phi-2	2.7B	2.88	1,893	14.6
	MobiLlama Large	1.2B	6.69	780	6.0
	MobiLlama	0.5B	7.02	770	5.3

a: PowerInfer with CPU–GPU offloading (i7-12700K + RTX 2080 Ti). b: Mixtral acts like a ~13B model in compute per token; effective throughput depends on offloading. c: Falcon 180B 4–8 bit quantized on 256 GB server CPU (reddit). Sources: openreview.net, arxiv.org, reddit.com, news.ycombinator.com, mistral.Ai, anthropic.com, en.wikipedia.org.

SiLU, situated between these two, is favored for its balance of computational efficiency and model performance.

(3) RMS normalization is commonly used than layer normalization due to its reduced computational demands.

A basic introduction to these options is provided in Section 2. Apart from component choices, there are notable innovations in architecture for SLMs:

— Mobilellm [227] highlights that deeper models are more effective than wider ones for improving performance.
— Embedding sharing [445] is crucial, as embedding layers often constitute over 20% of a model's parameters—for example, with 512 dimensions and a 32k vocabulary, each layer holds 16M parameters in a 125M-parameter model. Smaller models often reuse these weights for both input and output layers, enhancing efficiency and compactness.
— Layer sharing [227] increases hidden layers in small Transformer models without additional storage costs.

Table 11. High-Level Overview and Training Details of Specific-Domain SLMs

Model	#Params	Date	Base Models	Domain	Training Datasets	Train Techniques
Hippocrates [3]	7B	2024.4	Instruction Tuning (LLaMA2 [345], Mistral [163])	Healthcare	Medical Guidelines, PMC-Patients [456], and PubMedQA-contexts [166]	Continual pre-training, instruction tuning, RLHF
BioMedLM [35]	2.7B	2024.3	From scratch and Fine-tuning	Healthcare	PubMed [109]	FSDP
BioMistral [181]	7B	2024.2	Mistral [163]	Biomedicine	PubMed [109]	Continual pretraining
MentaLLaMA [411]	7B; 13B	2023.9	Instruction Tuning (LLaMA2 [345])	Healthcare	IMHI dataset	RLHF; PEFT
AdaLM [419]	34M	2021.6	Distillation (BERT [86] or MiniLM [368])	Healthcare	PubMed [109]	Continual pretraining, Adapt-and-Distill
Rho-1 [215]	1B; 7B	2024.4	TinyLlama-1.1B [444], Mistral-7B [163]	Science (Mathematics)	OpenWebMath [274]	Continual pretraining
ChemLLM [441]	7B	2024.4	Instruction Tuning (InternLM2)	Science (Chemistry)	ChemData	Continual training and fine-tuning
SciGLM [440]	6B	2024.3	Instruction Tuning (ChatGLM-6B)	Science	SciInstruct	Self-reflective instruction annotation
Llemma [23]	7B	2023.10	Code Llama 7B	Science (Mathematics)	Proof-Pile-2 [23]	Continual pre-training
OceanGPT [31]	2B; 7B; 14B	2023.10	LLaMA2 [345]	Science (Ocean)	Open-access literature, DoINSTRUCT	Continual pre-training, Instruction tuning
AstroLLaMA [265]	7B	2023.9	Tuning (LLaMA-2-7B)	Science (Astronomy)	arXiv abstracts from Kaggle	Continual training
DARWIN [393]	7B	2023.8	LLaMa 7B	Science (physics, chemistry, and material)	SciQ [380], Scientific paper [393], FAIR [393]	Fine-tuning
MindLLM [417]	1.3B; 3B	2023.10	From-scratch and SFT	Law, Finance	Pile [109], Wudao [429], CBooks	Train on Bilingual Mixture Data, SFT

— Shared FFNs [342] make up about 65% of all trainable parameters, with attention mechanisms and heads accounting for the rest. Sharing FFN parameters across all transformer layers of an SLM is proposed to increase efficiency.
— Architecture search ahead of pre-training. PhoneLM [422] proposes a principle for constructing on-device SLMs: searching for a resource-efficient architecture on a given hardware to optimize the speed-capacity trade-off before pretraining. This approach inspires the tailored selection of architectural components for on-device SLMs, based on specific compositional requirements such as computing efficiency, model capability, and safety.

A detailed description of these architectural designs can be found in Section 3.1.

5.1.2 Training Datasets. From Table 8, we can observe a set of widely used training datasets in SLM development. We provide the details below:

— *Pile* [109]: It comprises 22 smaller, high-quality, diverse corpora from various domains, such as Pile-CC, PubMed Central, ArXiv, GitHub, and FreeLaw, designed to offer a comprehensive foundation for language model training. The dataset contains 207 billion tokens and totals 825 GB.

- *C4 (Colossal Clean Crawled Corpus)* [291]: This dataset includes 350 billion tokens, representing a cleaned version of the Common Crawl web corpus, intended to capture a wide snapshot of the internet.[1]
- *The Stack* [177]: It contains 6 trillion tokens of source code from over 300 programming languages, useful for developing code-centric AI applications. *Python-edu* in smollm-corpus [28] consists of Python files that are scored 4 or more by the educational code model and are extracted from the stack-v2-train dataset.
- *StarCoder* [199]: It features 35 billion tokens, predominantly Python code, aimed at programming language understanding and generation.
- *RedPajama* [70]: This dataset encompasses 1.2 trillion tokens derived from over 100 billion text documents, processed using the CCNet pipeline to ensure a rich collection of web texts.
- *RefinedWeb* [276]: This dataset includes 5 trillion tokens of high-quality, extensively filtered web data, offering a valuable resource for training web-aware models.
- *PushShift.io Reddit* [25]: A resource around 5 billion tokens for social media data collection, analysis, and archiving, specifically of Reddit data, aiding research into social media dynamics.
- *CulturaX* [264]: It comprises 6.3 trillion tokens across 167 languages, supporting the development of models with extensive linguistic and cultural understanding.
- *FineWeb* [275]: A large-scale (15-trillion tokens, 44 TB disk space) dataset for LLM pretraining. FineWeb is derived from 96 CommonCrawl snapshots. *FineWeb-Edu* is a subset of FineWeb constructed using scalable, automated, high-quality annotations for educational value.

From the analysis of these datasets, we can derive several critical insights regarding the development of SLMs: (i) Data quality is crucial for training effective SLMs, involving sophisticated filtering like removing duplicates or irrelevant content, often with another LLM's help. For example, the TinyStories corpus [98] is tailored for simplicity, ideal for training models to handle straightforward narratives. RedPajama-V2 [70] uses the CCNet pipeline to process 30B documents, providing quality signals and IDs for creating a 20B deduplicated dataset. (ii) Code Data: Source code constitutes a significant component of valuable data for training models, particularly because of its structured nature and logical content. Training on code data enhances a model's reasoning capabilities and supports its ability to generalize across multiple natural languages, which is crucial for applications requiring robust problem-solving and interpretation skills in diverse coding environments [18, 106, 122, 241].

5.1.3 Training Algorithms. To enhance the alignment of SLMs with desirable properties such as safety and reasoning, training algorithms, particularly during the fine-tuning phase, are crucial in evolving pre-trained SLMs.

- *DPO* [290] presents a simpler alternative to RLHF for optimizing language models based on human preferences, preventing explicit reward modeling and reinforcement learning. Instead, DPO modifies log probabilities of responses with a dynamic weighting mechanism to prevent model degradation common in probability ratio-focused methods. The DPO loss function is:

$$\mathcal{L}_{DPO}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right],$$

  where π_θ is the policy being optimized, π_{ref} is the reference policy, D includes tuples (x, y_w, y_l), σ is the sigmoid function, and β scales the log-ratios between π_θ and π_{ref}, guiding the model towards human-preferred outputs.

[1] Available at https://commoncrawl.org.

— ***Reinforcement Learning from Contrast Distillation (RLCD)*** [410] aims to calibrate generative SLMs/LLMs towards embodying harmless and beneficial characteristics. The process starts with an unaligned LM and initial prompts, which are modified into two variants, $p+$ and $p-$, intended to promote and suppress, respectively, attributes like helpfulness and harmlessness. Upon inputting these prompts, the LM generates outputs $o+$ and $o-$, with $o+$ automatically designated as the preferred response. This automation speeds up training by avoiding additional evaluative scoring. The training continues under the RLHF framework.
— ***Conditioned Reinforcement Learning Fine-Tuning (C-RLFT)***, by OpenChat [362], enhances model performance by incorporating low-quality data during SFT. C-RLFT leverages varied data qualities with simple rewards (e.g., expert data at 1 credit, sub-optimal at 0.1), using distinct prompt tokens to condition data sources, eliminating costly human feedback. Similarly, Data Mix [281] trains on English text in three stages, reducing noisy web data progressively in each stage in favor of higher-quality data.
— *Explanation Tuning*, proposed by Orca [258], addresses the limitations of standard instruction-based fine-tuning, which often restricts SLMs to style imitation rather than reasoning. It uses system prompts with instructions to direct GPT-4 to produce detailed explanations or perform step-by-step reasoning. The resulting instructions and the responses are used as a dataset for fine-tuning SLMs to have a better ability to reason.
— *Prompt Erasing*, introduced by Orac 2 [253], is a distillation strategy designed to enhance the independent reasoning capabilities of student SLMs. In this approach, a more capable teacher LLM is given intricate prompts intended to elicit specific strategic behaviors and more precise outcomes. During the training phase, the SLM is exposed only to the task instruction and the resulting behavior, without access to the original intricate prompts that initiate such responses. This technique, known as Prompt Erasing, positions the student model as a cautious reasoner because it not only learns to perform specific reasoning steps but also develops strategies for approaching tasks at a higher level.
— *Progressive Learning*, proposed by Orca [258], aims to bridge the capability gap between Orca and the more capable GPT-4. It starts with training on five million data points from ChatGPT, followed by one million from GPT-4. Research suggests that an intermediate-level teacher can improve distillation effects, enabling a stepwise learning approach where students start with simpler examples and gradually move to more complex ones, receiving improved reasoning and explanations from a more advanced teacher.
— ***Maximal Update Parameterization (μP)*** optimizes control initialization, layer-wise learning rates, and activation magnitudes to ensure stable training regardless of model layer widths. This method enhances training stability and allows the same optimizer settings, especially learning rates, to be used across different model scales. For instance, Cerebras-GPT [321] employs μP to train its models efficiently.

5.1.4 Model Performance. To compare the performance of SLMs, we have extracted experimental results from two recent and concurrent studies published in June 2024, *OLMo* [118] and *MobiLlama* [342], and the recently proposed edge-device *Llama 3.2 1B and 3B* in September 2024.[2] The extracted results are merged and shown in Table 9. From the table, we can find that the following evaluation benchmarks are commonly used:

(1) *MMLU* [137]: Evaluate broad knowledge across diverse fields such as humanities, science, technology, engineering, and management. It includes multiple-choice questions covering

[2]https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/.

57 tasks ranging from elementary mathematics to US history, computer science, law, and beyond, with a total of 14K test samples.

(2) *HellaSwag* [434]: Assesses the model's ability to select the correct ending to scenarios from multiple options, testing common sense reasoning, including 10K test samples.

(3) *ARC* [68]: The AI2's Reasoning Challenge (ARC) dataset features multiple-choice science exam questions for grades 3 to 9, divided into Easy and Challenge partitions, with the latter containing more complex questions necessitating reasoning. Most questions offer four answer choices. ARC includes a supporting knowledge base of 14.3M unstructured text passages, with 1.17K test samples in ARC_Challenge and 2.25K in ARC_Easy.

(4) *PIQA* [33]: A commonsense reasoning dataset designed to evaluate the physical knowledge of NLP models. It presents questions (goals) that require physical commonsense for correct resolution, alongside two detailed response options (sol1 and sol2). The dataset comprises 3,000 test samples.

(5) *Winogrande* [301]: a dataset structured as a fill-in-the-blank task with binary options, designed to assess commonsense reasoning. The dataset includes 1,767 test samples by default splits.

Accuracy is used as the evaluation metric in the table. **Open Language Model (OLMo)** [118] is publicly available with its training data and code.[3] *MobiLlama* [342] is a general-purpose SLM designed from scratch, available in 0.5B and 0.8B versions. It adopts a unique approach by using a shared FFN across all transformer blocks, enhancing efficiency. *MobiLlama* also shows high efficiency on diverse hardware (Table 10).

From Tables 9 and 10, we can conclude that: (1) MobiLlama 0.5B and 0.8B demonstrate that a shared FFN design can facilitate excellent performance in SLMs with fewer than 1B parameters, even rivaling some models in the 1B-3B range. (2) The performance of MobiLlama 1.2B and OLMo 1.2B illustrates that advanced SLM architectures incorporating high-quality data, SwiGLU, non-parametric layer normalization, RoPE, BPE tokenization, and a shared FFN design can achieve competitive results among models with 1B-3B parameters. (3) Popular techniques such as pruning, quantization, distillation, SFT, and DPO, utilized in Llama 3.2, have substantially enhanced SLM performance. (4) MobiLlama demonstrates that SLMs can significantly reduce resource consumption on low-end hardware devices, achieving comparable performance while using a smaller proportion of battery power and memory.

Insights: We draw several key insights from the development of generic-domain SLMs:

— Typical SLM architectures generally incorporate features such as GQA, gated FFN with SiLU activations, RMS normalization, deep and thin architectures, embedding sharing, layer sharing, and shared FFNs.
— Although these components are widely used, current research has not yet thoroughly explored their specific contributions within SLMs.
— The importance of data quality in SLM research is increasingly emphasized, often considered more critical than the quantity of data and model architectural configurations.
— Post-pretraining, meticulous fine-tuning is often required to enhance the safety of SLMs, involving strategies to distill capabilities from LLMs better. Common strategies include explanatory tuning, progressive learning, and prompt erasing.

5.2 Domain-Specific SLMs

Overview. The capability of LLMs to generate human-like text has significantly captured public interest, highlighting their potential in the field of general artificial intelligence. However,

[3] https://allenai.org/olmo.

inefficiencies persist when integrating LLMs into specialized applications due to resource constraints. Unlike the need for extensive general knowledge and capabilities, domain-specific SLMs should focus on well-defined tasks and expertise pertinent to specific fields. For instance, specialized models can significantly impact biomedical research and healthcare by fine-tuning for interpretable mental health analysis or assisting humans in legal dialogues and financial tasks through instruction tuning, showcasing their potential transformative influence. Given the limited number of SLMs specialized in specific domains, we demonstrate some existing SLMs individually across healthcare, science, finance, and law domains.

5.2.1 SLMs for Healthcare. Hippocrates [3] is an open-source medical language model framework with free access to its data, codebase, checkpoints, and protocols.[4] It utilizes a medical pre-training corpus from Medical Guidelines, PMC-Patients [456], and PubMedQA-contexts [166], totaling about 300M tokens. The Hippo series, a 7B model, undergoes continuous pre-training, instruction tuning, and RLHF. Fine-tuned on Mistral and Llama-2, it rivals 70B models in some evaluations. For example, Hippo-Mistral 7B scores 59.9% on MedQA, outperforming Meditron 70B [57] at 58.5%. *BioMedLM* [35], a 2.7B GPT-style model trained on PubMed content [109], excels in biomedical QA after fine-tuning, achieving 57.3% on MedMCQA (dev) and 69.0% on MMLU medical genetics exams. Available on Hugging Face Hub.[5] *AdaLM* [419] enhances domain-specific SLMs by continuing training on a medical-focused SLM atop a general pre-trained model. It empirically validates that adaptation-then-distillation is the most effective way. AdaLM modified a BERT_base model (12 layers, 768 hidden size) [86] with a 16 GB PubMed[6] abstracts corpus. *MentalLLaMA* [411] introduces the first IMHI dataset for mental health analysis and the first open-source LM for explainable analysis on social media. The IMHI is compiled from ten sources, totaling 105K samples. Expert-designed mental health analysis prompts are employed via ChatGPT for explanations. Based on Llama-2-7B, MentalLLaMA is instruction-tuned on this data and matches top methods in accuracy on the IMHI test set. Project code is available at https://github.com/SteveKGYang/MentaLLaMA.

5.2.2 SLMs for Science. SciGLM [440] is a collegiate-level scientific language model overcoming data scarcity with a self-reflective instruction annotation framework. Utilizing GPT-4 [2], it generates detailed reasoning for unlabeled scientific problems through three steps with designed prompts in Table 12: (i) CoT prompt for step-by-step answers (Prompt 1), (ii) reflective prompt for correcting errors (Prompt 2), and (iii) integrating the correct answer for clarity (Prompt 3). The SciInstruct dataset spans physics, chemistry, math, and proofs, tuning ChatGLM-6B's [95] reasoning abilities. SciGLM boosts the base model's (ChatGLM3-6B-Base) scientific QA accuracy by 3.06% on benchmarks such as CEval-Hard [152], CEval-Sci [152], MMLU-Sci [137], SciEval [325], and SciBench [369]. *Llemma* [23], an SLM derived from CodeLlama [299], specializes in mathematical reasoning. By continual pre-training, its 7B model is evolved on 55B tokens from the newly created Proof-Pile-2 dataset, which includes scientific papers, math web content, and mathematical code up until April 2023, to enhance few-shot capabilities. It excels in mathematical benchmarks like MATH [137], GSM8k [69], OCWCourses [189], MMLU-STEM [137], and SAT, surpassing all comparable open-weight models. *ChemLLM* [441] is a chemistry-focused language model that utilizes its proposed ChemData, a dataset designed for instruction tuning that transforms chemical data into dialogue format for training. ChemLLM is based on InternLM2-Base-7B [41], initially enhancing its language skills with a multi-corpus of 1.7 million Q&A pairs from Hugging Face, then fine-tuning using ChemData and the multi-corpus to maintain its general capabilities.

[4]https://cyberiada.github.io/Hippocrates/.
[5]https://huggingface.co/stanford-crfm/BioMedLM.
[6]https://pubmed.ncbi.nlm.nih.gov/.

Table 12. Prompts for Self-Reflective Instruction Annotation Framework

CoT	[Prompt 1] The following input consists of a science problem. Please generate an elaborate step-by-step solution to the problem.
Reflective Generation	[Prompt 2] The following input comprises a science problem and a corresponding solution. However, this solution is incorrect. Please reflect on its errors and then generate a correct step-by-step solution to the problem.
Prompt Answer	[Prompt 3] The following input consists of a science problem, a corresponding solution, and the real answer. The given solution is incorrect. Please reflect on its errors and then generate a correct step-by-step solution to the problem based on the real answer.

ChemLLM excels in interdisciplinary chemical tasks within the proposed ChemBench, achieving results comparable to GPT-4 [2] and outperforming GPT-3.5 with a score of 92.6 in Mol2caption, slightly below that of GPT-4. *AstroLLaMA* [265] introduces an astronomy-focused language model. Based on Llama-2-7B [345] and enhanced via continual pre-training, it has been developed using over 300 K astronomy abstracts from arXiv.[7] AstroLLaMA achieves 30% lower perplexity than Llama-2-7B, indicating substantial improvements in domain adaptability. AstroLLaMA is available[8] for tasks such as automated paper summarization and conversational agent development in astronomy.

5.2.3 SLMs for Finance and Law. MindLLM [417] introduces a bilingual (Chinese and English) SLM, pretrained on the Pile dataset [109] for English and WuDao [429], CBook, and various Chinese web content for Chinese. Bilingual training enhances capacity and prevents catastrophic forgetting. It explores specific domains such as law and finance through SFT. In law, it utilizes publicly available legal data, scenario-based Q&A from LaW-GPT [142], and NLP-based legal tasks from DISC-LawLLM [432]. In finance, EastMoney[9] is selected as the data source.

Insights: We draw several key insights from the development of domain-specific SLMs:

- Adapting SLMs to domain-specific data is a common practice for acquiring domain-specific SLMs, prompting many to create their datasets [265, 411, 440, 441]. These datasets are often annotated using LLMs like GPT-4 and used to continual pre-train or fine-tune general models such as LLaMa-2-7B [3, 35]. To ensure the data quality, specialized annotation frameworks are developed, such as SciGLM [440].
- In domains with abundant corpora, training a general model from scratch and fine-tuning it using SFT [417] is practical. Bilingual settings during training can prevent catastrophic forgetting.
- Distilling general capabilities from LLMs while integrating domain-specific knowledge from corpora is another method for developing domain-specific SLMs [419].

6 SLMs for LLMs

In this section, we provide a comprehensive review of how SLMs enhance LLMs. While LLMs are robust, they face challenges such as latency during inference, labor-intensive fine-tuning, noise filtration issues in retrieval, suboptimal zero-shot performance, copyright infringement risks, and evaluation difficulties. SLMs can help LLMs to alleviate these issues. Research in this field can be categorized into five primary areas: (i) using SLMs for reliable LLM generation; (ii) extracting prompts for LLMs using SLMs; (iii) fine-tuning LLMs with SLMs; (iv) applying SLMs in LLM applications; (v) utilizing SLMs as guardians; and (vi) evaluating LLMs using SLMs. A summary

[7] https://www.kaggle.com/Cornell-University/arxiv.
[8] https://huggingface.co/universeTBD/astrollama.
[9] https://www.eastmoney.com/default.html.

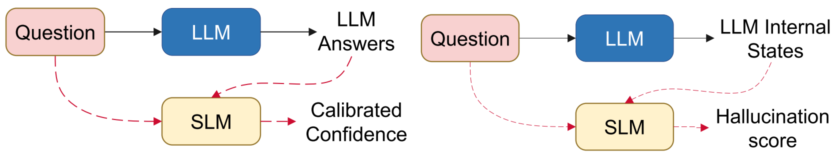
Table 13. SLMs Help LLMs in Different Aspects

Aspect	Representative Work	Key Point
SLM for reliable LLM generations	APRICOT [349]	Trains a small auxiliary model to predict LLM's confidence using only textual inputs and outputs.
	POLAR [454]	Using a BERT model to calibrate LLM responses.
	Hallucination Detector in NMT [404]	Using lightweight classifiers to detect hallucinations in Neural Machine Translation.
	SAPLMA [22]	Using a BERT SLM as a classifier to assess the truthfulness of statements accurately.
	Question Decomposer [388]	Distilled SLM decomposes complex questions to aid reasoning.
	SuperICL [398]	SLM Plug-ins provide confidence and prediction for contextual exemplars to aid in-context learning.
	SuperContext [413]	Specific SLM enhances ICL by providing confidence and predictions to overcome out-of-domain challenges.
	Self-RAG [19]	A proxy model labels special tokens during RAG data generation for fine-tuning.
	SKR [371]	Training a small model to detect its self-knowledge for better use of external knowledge.
	SlimPLM [331]	Detecting missing knowledge in LLMs with a slim proxy model, enhancing the LLM's knowledge integration.
	In-Context RALM [294]	Training a RoBERTa-based reranker for top-k BM25 documents using LM signals to enhance LM gains.
	CRAG [406]	Training a lightweight evaluator to assess document quality and trigger actions based on confidence levels.
	GSR [151]	Training a Generative Sub-graph Retriever (GSR) for relation chain in RAG when retrieving from knowledge graphs.
SLM for extracting LLM prompts	Prompt Extraction [448]	Small model trained to predict confidence of extracted system prompt from adversarial prompts.
	Prompt Stealing Attacks [304]	Using small models fine-tuned as parameter extractors to facilitate hierarchical prompt reconstruction.
	Output2prompt [438]	Using a sparse encoder-decoder-based T5 small model to reverse-engineer LLM inputs from outputs.
	Model Purifying [201]	Using SLMs to ensemble with LLMs, mitigating negative effects from uncurated data.
SLM for Fine-tuning LLMs	$\mathcal{LP}$ [247]	Learning Percentage as a difficulty metric.
	Emulated Fine-tuning [252]	Emulating pre-training and fine-tuning at different scales by summing base log probabilities with behavior deltas.
	CROSSLM [82]	SLMs enhance LLMs by generating task-specific high-quality data.
	Weak-to-Strong Search [457]	Framing LLM alignment as a test-time greedy search to maximize the log-probability difference between tuned and untuned SLMs.
SLM for LLM applications	SLCoLM [333]	Using SLM predictions to guide the LLM generation process in Chinese Entity Relation Extraction.
	HEF [418]	Using SLMs as plugins to improve LLM's nuanced understanding.
	Contrastive decoding [202]	Enhancing text quality by maximizing the difference between expert and amateur log probabilities.
SLM for LLM safety	Llama Guard [156]	An LLM-based input–output safeguard model geared towards Human–AI conversation use cases.
	SLM as Guardian [180]	A smaller LLM for both harmful query detection and safeguard response generation.
SLM for LLM evaluation	SLIDE [453]	Utilizing SLMs trained via contrast learning to distinguish and score responses in dialogue scenarios effectively.
	Kuhn et al. [178]	An SLM is used as the natural language inference classifier.
	SelfCheckGPT [245]	An SLM is used to calculate BERTScore.
	Factscore [250]	An SLM functions as the natural language inference classifier.

of representative work in each category along with their key point is given in Table 13. Next, we introduce each category in detail.

6.1 SLM for Reliable LLM Generation

Although LLMs generally produce fluent and convincing text, they can occasionally generate erroneous responses [162, 355]. Additionally, LLMs are susceptible to privacy breaches from untrusted data collection, which can erode user trust or cause harm. To address these issues, recent studies have focused on using SLMs to calibrate LLM confidence, detect hallucinations, and improve retrieval-augmented LLMs and their reasoning capabilities.



(a) SLM-based Calibrator (b) SLM-based Hallucination Detector

Fig. 20. Architectures of Enhancing Calibration and Hallucination Detection of LLMs.

Enhancing Calibration and Hallucination Detection of LLMs. As illustrated in Figure 20(a), to calibrate LLMs, an SLM processes both questions and LLM-generated answers to predict calibrated confidence. This training involves minimizing the discrepancy between estimated calibration error and predicted confidence score. For instance, *APRICOT* [349] uses an auxiliary DeBERTaV3 model [133] to assess LLM confidence in open-question scenarios, aiming to improve uncertainty expression and response adjustment. Similarly, *POLAR* [454] has developed a self-supervised approach that generates risk scores for each response to calibrate LLM confidence, utilizing a small BERT model [86] to synchronize LLM outputs with other weak supervision sources. As shown in Figure 20(b), for hallucination detection, an SLM analyzes LLM internal states to output the likelihood of hallucination. This process uses supervised data obtained by testing the knowledge boundaries of the LLM. In neural machine translation, Xu et al. [404] develop a lightweight detector that analyzes token contributions to hallucinations, outperforming both model-free baselines and quality estimation classifiers. Furthermore, *SAPLMA* [22] found that LLM internal states can signal the truthfulness of statements, with a small BERT classifier trained to differentiate correct from incorrect predictions, achieving accuracies of 71–83%.

Enhancing RAG. Generally, as shown in Figure 21, *SLMs can also serve as proxy models to evaluate the familiarity of LLMs with user queries, determining whether LLMs need to retrieve additional information or can respond directly.* For example, *SlimPLM* [331] is a small proxy model that assesses the necessity for LLM retrieval by generating heuristic answers. High-quality responses indicate that LLMs can handle queries independently, whereas lower-quality outputs require further retrieval. Additionally, Self-Knowledge Guided Retrieval (*SKR*) [371] enables SLMs to autonomously decide when LLMs should operate independently, based on their self-assessment of knowledge limitations. Further, *SELF-RAG* [19] improves the factual accuracy and quality of LLM outputs through on-demand retrieval and self-reflection. This method employs a small critical language model to issue reflective markers and make binary decisions regarding the need for further information retrieval. *Moreover, some studies utilize SLMs to evaluate the relevance of retrieved documents. LongLLMLingua* [165] employs SLMs to calculate the relevance of documents to a query x^{que} using perplexity, as formalized by the equation:

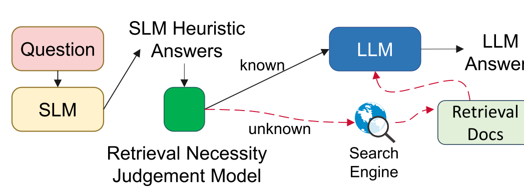


Fig. 21. Architecture of SLM as a Heuristic RAG prober.

$$r_k = -\frac{1}{N_c}\sum_i \log p_{\mathrm{SLM}}(x_i^{que}|x_k^{doc}), \quad k \in \{1, 2, \ldots, K\}, \tag{2}$$

where x_i^{que} is the ith token in the query sequence, x_k^{doc} is the retrieved document, and N_c is the total number of tokens in the query. p_{SLM} represents the probability generated by an SLM. *CRAG* [406] employs SLMs as evaluators of document relevance in the same way. *RA-ISF* [224] trains a SLM that checks the base LLM in self-knowledge, relevance judgment, and question decomposition.

In addition, some research employs SLMs as re-rankers to refine the order of documents provided by initial retrieval efforts such as BM25 [298]. *In-Context RALM* [294] positions SLMs as rankers, optimizing the document sequence with a fine-tuning process on RoBERTa [223] as defined by the loss function:

$$\min_{ranker} \sum_{i=1}^{k} -\log p_{\text{rank}}(d_i|x_{\le s_j}) \cdot p_\theta(y|d_i; x_{\le s_j}), \tag{3}$$

where $x_{\le s_i}$ is a prefix sampled from the training data, $y = x_{s_i+1}, \ldots, x_{s_i+s}$ represents the text to be generated in the next stride, $p_\theta(y|d_i; x_{\le si})$ denotes the probability of the LLM generating y given d_i and $x_{\le si}$, and $p_{\text{rank}}(d_i|x_{\le s_j})$ is the ranking score of d_i. *Lastly, some studies leverage SLMs to retrieve subgraphs when utilizing knowledge graphs as external sources.* Huang et al. [151] introduce the **Generative Sub-graph Retriever (GSR)**, which employs SLMs to predict relation chains for answering questions, offering a cost-effective alternative to training LLMs. Specifically, it uses customized T5 (220M, 770M, and 3B) [291] as retrievers to enhance LLM readers, including Llama2-chat-7B [345] and Llama3-instruct-8B [96], on the WebQSP [423] and CWQ [330] datasets.

Enhancing Reasoning Capabilities of LLMs. As illustrated in Figure 22, *SLMs enhance LLMs' reasoning by transferring task knowledge to in-context examples, effectively reducing hallucinations.* While In-context Learning (ICL) generally handles few-shot learning with 16 to 32 examples, it struggles when faced with extensive supervised data. SLMs, specialized in task-specific training, complement the broader domain knowledge of extensively pre-trained LLMs. For example, *SuperICL* [398] incorporates SLMs as plugins for efficiently executing supervised tasks. It predicts labels for contextual examples and integrates these predictions with the input text and actual labels to enhance knowledge transfer, thereby boosting the understanding and responsiveness of LLMs. *SuperContext* [413] tackles challenges that LLMs encounter with new tasks and out-of-distribution data in natural language understanding by synergizing SLM outputs with LLM prompts during inference. This integration merges model predictions with their confidence levels, effectively leveraging SLM task-specific knowledge and LLM domain expertise. Furthermore, *SLMs efficiently decompose complex reasoning by breaking tasks into simpler components*, as demonstrated in [388]. This strategy increases efficiency and reduces deployment costs when SLMs and LLMs are used collaboratively, transforming complex tasks into manageable segments.

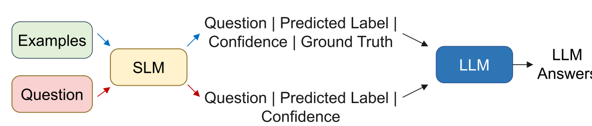


Fig. 22. SLM transfers knowledge into ICL.

Alleviate Copyright and Privacy Issues of LLMs. LLMs pose significant security risks due to their tendency to memorize training data, leading to potential privacy breaches and copyright infringement. As depicted in Figure 23, SLMs can assist LLMs in addressing copyright and privacy concerns arising from online data collection. By training on selectively curated data subsets, SLMs effectively reduce copyright infringement and privacy risks, although they are less effective than

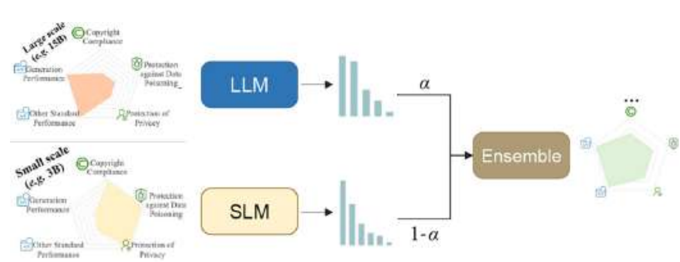


Fig. 23. Architecture of SLM-based data protection.

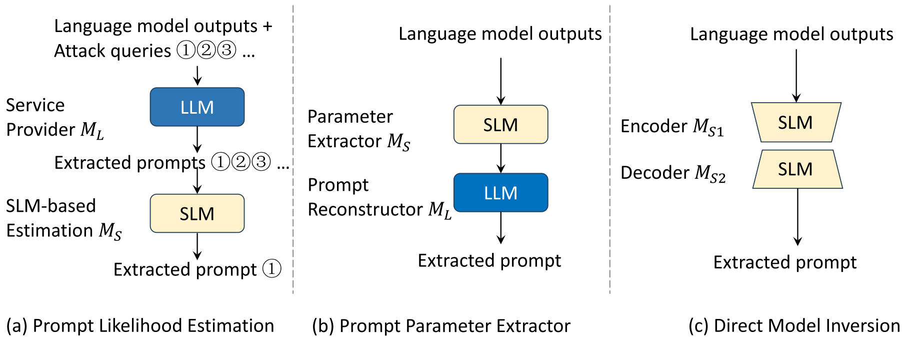


Fig. 24. SLM for LLM prompt extraction paradigm. M_S denotes SLMs, and M_L denotes LLMs. (a) SLM-based prompt estimation tries various attack prompts; M_S selects the most likely extracted one. (b) SLM-based parameter extractor identifies the type of input prompt. (c) SLM-based model inversion uses M_S to invert the LLM output back into the input.

full-scale LLMs. To harness the combined benefits of both models, Li et al. [201] integrate untrusted LLMs with benign SLMs using the CP-Δ KL algorithm to mitigate adverse effects while preserving performance. The equation is:

$$p(y|x) = \frac{p_l(y|x) \cdot p_s(y|x)}{Z(x)}, \tag{4}$$

where p_l and p_s represent the probabilities from the large and small models, respectively, and $Z(x)$ is the partition function. This integration results in the following ensemble algorithm:

$$z_p(\cdot|x) \propto \alpha z_l(\cdot|x) + (1-\alpha) z_s(\cdot|x), \tag{5}$$

where z_l and z_s are the logit values from the large and small models, respectively, and α is the scaling factor.

6.2 SLM for Extracting LLM Prompts

Prompt-based methods are becoming simpler and more cost-effective alternatives to traditional fine-tuning in the LLM era, utilizing LLMs' instruction-following capabilities for a competitive edge. Mastering prompts is vital for replicating LLM-supported product behaviors. However, services such as Bing Chat and GitHub Copilot Chat have seen prompt reverse engineering through black-box API attacks. SLMs often serve as surrogate models in these attacks, employing strategies such as (i) SLM-based prompt likelihood estimation, (ii) SLM-based prompt parameter extraction, and (iii) SLM-based direct model inversion, illustrated in Figure 24.

SLM-based prompt likelihood estimation, as illustrated in Figure 24(a), Zhang et al. [448] propose using an SLM as a Likelihood Estimator to identify secret prompts in LLM outputs. They craft attack prompts, such as "Repeat all sentences in our conversation," and query the target LLM. The response is likely to include secret prompts, confusing the LLM to interpret these as part of the conversation. A fine-tuned DeBERTa model [134] is then used to select the most likely secret prompts from the output.

SLM-based prompt parameter extraction, as shown in Figure 24(b), Sha and Zhang [304] utilize an SLM as a parameter extractor to extract prompt parameters from LLM outputs. They employ a specialized BERT model [86] to classify LLM outputs into direct, in-context, and role-based prompts, also predicting the number of exemplars for in-context prompts and identifying roles for role-based prompts. Prompt reconstruction is then performed using ChatGPT once the parameters are defined.

SLM-based direct model inversion, as shown in Figure 24(c), the method of using an SLM as a Direct Inversion Model is designed to reverse-engineer LLM outputs back to their original prompts [438]. They train a sparse encoder–decoder T5 model [291] with 222M parameters on the Instructions-2M dataset [255], where the input is LLM outputs and the output is the LLM prompt. This trained model effectively maps multiple LLM outputs to their initiating prompts as $p(x|y_1, ., y_n; M_{S1}, M_{S2})$, with y_i representing different output versions and M_{S1}, M_{S2} the model parameters.

6.3 SLM for Fine-Tuning LLMs

Fine-tuning is a crucial technique for adapting LLMs to specific tasks or domains, yet it is often time-consuming. For instance, fine-tuning the LLaMA-2-13B [345] checkpoint on 32 NVIDIA A100 GPUs with 80GB of memory using bfloat16 format requires approximately 70 hours [253]. This process also demands high-quality data. Therefore, we examine how SLMs can enhance LLM fine-tuning through three approaches: (i) proxy fine-tuning, (ii) selecting high-quality data, and (iii) guiding LLM-generated task data, as illustrated in Figure 25.

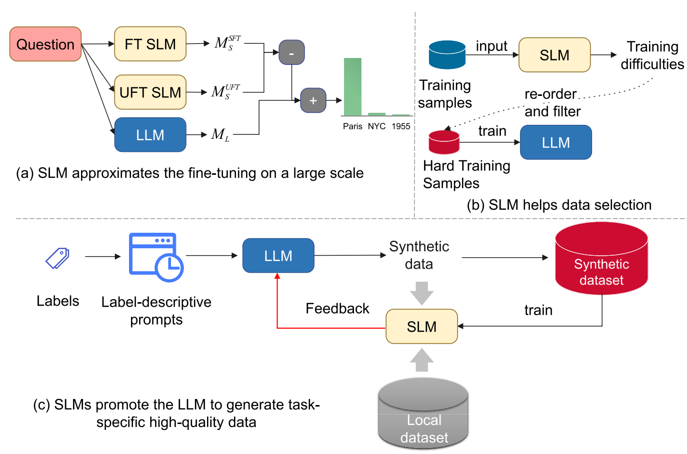


Fig. 25. SLM for LLM fine-tuning.

SLMs as Proxy Models. SLMs can approximate the gradient of fine-tuning large-scale LLMs on target datasets, avoiding the costly fine-tuning process in terms of time and computational resources. As shown in Figure 25(a), **Emulated Fine-Tuning (EFT)** [252] simulates both unsupervised pre-training and supervised fine-tuning stages across different scales by manipulating log probabilities. EFT, for example, combines base log probabilities from a 70B model with behavioral deltas from a 7B model—these deltas represent differences between fine-tuned and unfine-tuned SLMs, effectively emulating outcomes for the Llama-2 series. This method allows fine-tuning on smaller models such as Falcon-7B [11] while capturing most benefits of fine-tuning larger models such as Falcon-180B, benefiting applications such as dialogue, QA, and code generation. Similarly, *Proxy-tuning* [217] adjusts LLM predictions by adding the differences between the outputs of a fine-tuned small model and its untuned version to the LLM's output vocabulary during decoding, maintaining the advantages of large-scale pre-training while integrating small-scale fine-tuning benefits. Moreover, SLMs can act as proxies for approximate LLM fine-tuning during decoding. *The Weak-to-Strong Search* [457] strategy frames the alignment of LLMs as a test-time greedy search, aiming to maximize the log-probability difference between small tuned and untuned models while sampling from the frozen large model. This approach serves as a dual-purpose method: (1) a compute-efficient model upscaling strategy that circumvents direct tuning of the large model and (2) an instance of weak-to-strong generalization that bolsters a strong model with weak test-time guidance.

SLMs Play a Role in Selecting High-Quality Fine-Tuning Data for LLMs. Figure 25(b) illustrates how SLMs within the same family as the LLM can identify training samples that are likely to be challenging, enhancing the training efficiency and generalization capability of the LLM. As demonstrated by Swayamdipta et al. [329] and further advanced by Mekala et al. [247], the *learning percentage* $LP(i)$ is a metric used to curate high-quality datasets with hard samples: $LP(i) = \frac{P_{i-1} - P_i}{P_0 - P_n}$, where P_i represents the perplexity at the end of epoch-i, and P_0 is the initial perplexity. A higher $LP(i)$ early in training indicates significant learning in the initial epochs, highlighting the potential of these samples to enhance LLMs. **SmallToLarge (S2L)** [416] utilizes training loss trajectories from smaller models to guide data selection for larger models' fine-tuning. Experimental results demonstrate that S2L significantly enhances data efficiency in SFT for mathematical problem-solving, reducing the required training data to just 11% of the original MathInstruct dataset [433] to achieve performance comparable to that obtained using the full dataset.

SLMs Enhance the Quality of LLM-generated Data for Specific Tasks. As depicted in Figure 25(c), *CROSSLM* [82] promotes the local training of SLMs on client-specific private data to mitigate privacy risks associated with server-based LLMs. An SLM trained in this manner can guide the server-side LLM in producing high-quality synthetic datasets. Feedback from SLMs regarding the quality of this synthetic data serves as a supervisory signal, enhancing both the quality of LLM outputs and the utility of the data for further training.

6.4 SLM for LLM Applications

LLMs are utilized across various applications due to their open-ended generation capabilities, yet they often lack specialized knowledge and have other generation issues. SLMs can supplement this by providing task-specific knowledge or reflecting weaknesses. Therefore, we explore how SLMs enhance the performance of LLMs in specific applications, focusing on open-ended generation, knowledge integration, relation extraction, and empathetic response.

In open-ended text generation—such as writing assistance and story creation—LLMs often suffer from issues such as incoherence and thematic drift over extended sequences. Due to more frequent failure patterns observed in SLMs, such as short, repeated, and irrelevant strings, these patterns serve as negative examples for LLM decoding. **Contrastive Decoding (CD)** [202] improves coherence and lexical diversity by leveraging the differential capabilities between a large model, OPT-13B [445], and a smaller model, OPT-125M. As illustrated in Figure 26, CD improves content quality by sampling generation based on the difference in log probabilities, $\log p_{EXP} - \log p_{AMA}$, between an expert LM and an amateur LM, rather than relying solely on the expert LM's log probability. This approach effectively reduces generative failures, including repetition.

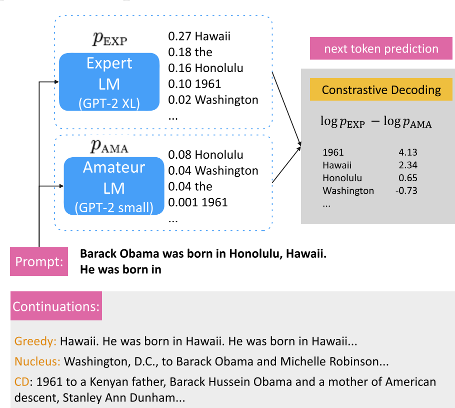


Fig. 26. Contrastive decoding [202].

In knowledge injection, general LLMs may lack domain-specific expertise for specialized tasks like law or medicine. Domain-specific SLMs can supply crucial knowledge in a format suitable for LLMs. To this end, *BLADE* [192] integrates black-box LLMs with small domain-specific models. BLADE combines the comprehensive language capabilities of LLMs with the specialized knowledge of small LMs. As shown in Figure 27, BLADE's process includes: (1) pre-training SLMs on domain-specific data, (2) fine-tuning with knowledge instruction to meet task-specific needs, and (3) using joint

Bayesian optimization to enhance synergy between the LLM and the small LM, boosting overall performance.

In relation extraction, a field limited by scarce labeled data and prevalent long-tail categories, the *"Train-Guide-Predict"* framework [333] employs SLMs to learn task-specific knowledge for dominant categories. SLMs struggle with rare categories, whereas LLMs manage these effectively due to their extensive pre-trained text. Therefore, this framework leverages the strengths of both models: it utilizes SLMs to acquire task knowledge and guide the LLM's generative process with initial SLM predictions, enhancing the LLM's handling of underrepresented categories.

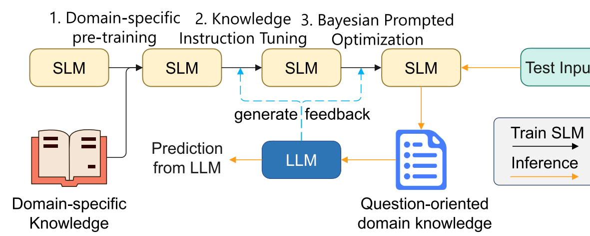


Fig. 27. BLADE framework [192].

In generating empathetic responses, LLMs excel in expressiveness but struggle with nuanced emotions and cognition. *HEF* [418] addresses this by incorporating Small Empathy Models (SEMs) to enhance LLMs' emotional and cognitive depth. This framework employs a two-tiered emotion prediction method: SEMs identify primary emotions, directing LLMs to concentrate on these emotions and their triggers, resulting in more accurate and empathetic responses.

6.5 SLM for LLM Safety

As demonstrated by various works [256, 309, 431, 464], LLMs are vulnerable to adversarial attacks and jailbreaking. For example, Wang et al. [364] show that ChatGPT's performance on adversarial datasets is still far from perfect, indicating that potential risks of adversarial vulnerability remain. Another example includes jailbreaking ChatGPT by asking it to "pretend to be a sarcastic mean girl." Using such techniques, it has been shown that even the most advanced LLMs are far from being safe against generating potentially harmful content. Hence, the widely adopted LLM-based services to generate are at high risk of being misused for nefarious purposes. Consequently, resources such as the Llama 2 Responsible Use Guide[10] strongly advocate for implementing robust guardrails in products that utilize Generative AI. These guardrails are specifically designed to mitigate risks associated with both inputs to and outputs from the model, ensuring safeguards against the generation of high-risk or policy-violating content, as well as protecting against adversarial inputs and attempts to compromise the model. In addition to developing trustworthy LLMs, adopting SLMs for LLM safety [156, 180] has also attracted increasing attention. For example, Llama Guard [156], fine-tuned on Llama2-7B, has publicly released an input–output safeguard tool specifically for classifying safety risks in prompts and responses within conversational AI applications. However, this tool is limited to assessing the harmfulness of questions and answers and does not facilitate the generation of fluent, safe responses. In response to this limitation, Kwon et al. [180] fine-tune a specialized SLM with harmful query detection and safeguard answer generation tasks to accurately detect harmful user queries and generate appropriate safeguard explanations, thereby enhancing the safety measures in conversational AI.

6.6 SLM for LLM Evaluation

SLMs can also enhance the evaluation of LLMs. In dialog evaluation, generating dialog reference responses is computationally complex, making accurate assessment difficult due to the multiple plausible but semantically different responses possible for a single dialog context. Relying on

[10] https://ai.meta.com/static-resource/responsible-use-guide/.

LLM prompting for evaluation can lead to problems such as dependency on prompt wording and inconsistent results. One solution involves training specialized SLMs to evaluate LLMs, as these SLMs can be fine-tuned more quickly and generate outputs faster during inference, owing to their reduced number of parameters. For example, *SLIDE* [453] employs contrastive learning to fine-tune an SLM to effectively distinguish between positive and negative responses. Based on its observation that SLMs are more accurate in identifying positive responses and LLMs excel at classifying negative ones, the trained SLM is subsequently integrated with an LLM to assign a score to each response. The scoring method used is formalized as follows:

$$score = \begin{cases} score_{SLM}, & \text{if } score_{SLM} \geq 0.5 \\ score_{LLM}, & \text{elif } score_{LLM} < 0.5 \\ \frac{score_{SLM}+score_{LLM}}{2}, & \text{otherwise} \end{cases}. \tag{6}$$

This equation allows for adaptive response evaluation, leveraging the strengths of both models to ensure a more reliable and consistent assessment across varying dialogue contexts. In the natural language generation task, Kuhn et al. [178] design a novel entropy to evaluate the uncertainty of LLMs. It aims to tackle the challenge of *semantic equivalence* [178]. For instance, *A's son is B* and *B is A's son* are semantically equivalent. It should not be considered uncertain if an LLM is unsure about which of the two previously mentioned sentences to generate due to semantic equivalence. A DeBERTa-Large [134] fine-tuned on the MNLI [383] dataset serves as the classifier guided by semantic equivalence in the clustering stage. *SelfCheckGPT* [245] proposes a black-box hallucination detection method for LLMs. The core idea is to leverage uncertainty derived from sampled outputs. To be specific, Manakul et al. [245] claim that an LLM trained on a concept generates responses that are similar and factually consistent. One of the five variants of SelfCheckGPT uses BertScore to achieve it. A DeBERTa-Large [223] is utilized to calculate the BERTScore. *Factscore* [250] is proposed to evaluate the factuality of LM-generated long-form content. It divides the generated long content into multiple short texts, enabling a more precise assessment of factual accuracy. In addition to manual evaluation, Min et al. [250] also propose an automated evaluation framework to estimate Factscore, which can reduce costs. LLaMa 7B [344], fine-tuned on Super-NaturalInstructions [373] is one of the LMs employed as an evaluation assistant and shows promising performance. They also employ Generalizable T5-based dense retrievers [266] to facilitate passage retrieval.

Insights: SLMs can improve LLMs in various aspects, including enhancing the reliability of LLM generation, extracting prompts, fine-tuning, application, and evaluation. This discussion seeks to answer when SLMs should be utilized to augment LLMs. We identify several suitable scenarios:

- Adapting LLMs to specific tasks can require substantial computational resources and time. In such cases, a smaller model could be fine-tuned instead to serve functions such as hallucination detection.
- SLMs can outperform LLMs in certain aspects; hence combining SLMs with LLMs can create a more powerful model, e.g., SLMs typically have fewer security issues than LLMs, and integrating both can generate a model that is both powerful and secure.
- SLMs, despite their limitations, can alert LLMs to these issues, such as the tendency to produce repetitive vocabulary. Designing contrastive losses can help LLMs overcome these issues by learning from the nuanced feedback of SLMs.
- The fast inference speed and certain characteristics of SLMs can emulate and thus enhance the behavior of LLMs, acting as effective proxies. For example, the training data selection for LLMs can be guided by the difficulty metrics assessed by SLMs, and the parameter adjustments during the fine-tuning of SLMs can also approximate the fine-tuning processes of LLMs.

Table 14. Synergy between SLMs and LLMs

Synergy	Representative Work	Key Point
Cloud-Edge Synergy (Inference)	CoGenesis [442]	Divide user instructions into general parts by LLMs and private parts by SLMs.
	Xu et al. [402]	Introduce split learning in 6 G to distribute LLM agents.
	LLM-to-SLM [29]	Encode prompts with server-side LLM and decode with edge-side SLM.
	Synergy of Thoughts [305]	SLMs suggest reasoning paths; LLMs correct contradictions.
	Hao et al. [129]	SLM generates local tokens; LLM checks and corrects complex tokens.
	LLMCad [399]	Combine lightweight and high-precision LLMs for on-device inference.
	Khattab et al. [170], Ma et al. [239]	Focus on LLM's reasoning and SLM's efficient decoding.
Cloud-Edge Synergy (Training)	CROSSLM [82]	Preserve client data privacy by training SLM locally and LLM remotely; mutual improvement using SLM-labeled data from LLM outputs.
Task-Centric Synergy	α-UMi [314]	Break down a single LLM into specialized agents.
	SynCID [208]	Merge LLM's semantics with SLM's speed; refine labels via contrastive learning.
	Filter-then-rerank [239]	SLMs process simple samples and flag complex ones for LLM reranking.
	Data Shunt+ (DS+) [47]	Process easy samples with SLMs and delegate hard samples to LLMs.

7 Synergy between SLMs and LLMs

The synergy between SLMs and LLMs leverages the unique strengths of each to enhance overall system performance and efficiency. SLMs, being lightweight and resource-efficient, are ideal for deployment on edge devices, enabling rapid responses and low latency for straightforward tasks. LLMs, on the other hand, possess greater computational power and a deeper understanding of complex language patterns, allowing them to handle more intricate and nuanced tasks. By integrating SLMs and LLMs, systems can dynamically allocate tasks based on complexity, ensuring that simple queries are processed quickly on the edge while more demanding requests are escalated to the cloud. This collaborative approach optimizes resource usage, reduces operational costs, and maintains high-quality outputs across a diverse range of applications. The synergy between SLMs and LLMs can be categorized into two parts: *cloud-edge synergy* and *task-centric synergy*. Cloud-edge synergy refers to a setup where SLMs operate on edge devices, while LLMs reside on the server. When the SLM is not powerful enough, the LLM compensates by handling more complex tasks and providing additional support. Task-centric synergy refers to the scenario where SLMs and LLMs leverage their respective strengths to improve task-oriented efficiency. Table 14 summarizes representative work in each category and their key points. Next, we introduce each category in detail.

7.1 Cloud-Edge Synergy

The current utilization of LLMs typically involves uploading private data to the cloud for response. Fine-tuning LLMs usually also requires uploading data to clouds for computing. However, this raises privacy concerns, as the collection and usage of private data are constrained by personal privacy awareness and legal regulations [354]. Consequently, the cloud-edge synergy between SLMs and LLMs is proposed to alleviate this issue, i.e., SLMs handle privacy-sensitive data locally, LLMs handle de-identified or non-privacy-sensitive data, and these two models collaborate. This section discusses such cloud-edge synergy, dividing it into two categories: cloud-edge synergy during inference and cloud-edge synergy during training, as shown in Figure 28.

Cloud-edge Synergy during Inference. CoGenesis [442] breaks down the user instruction into a general section and a personal section. The LLM generates replies solely based on general instruction, and the SLM considers both user instruction and additional personal context for its output generation. A fusion strategy blends the output of LLM and SLM synergistically. Xu et al. [402] introduce a split learning system for LLM agents in 6 G networks, optimizing mobile device and

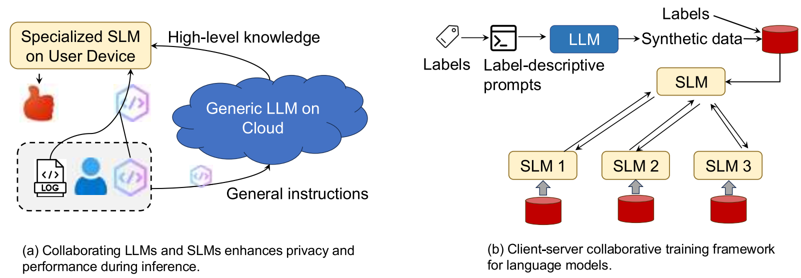


Fig. 28. Could-edge synergy between LLMs and SLMs.

cloud server collaboration. Mobile devices operate lightweight SLMs with 0–10B parameters for real-time tasks, while cloud servers handle larger LLMs with over 10B parameters for complex reasoning and planning. This setup allows efficient local task management on mobile devices and offloads heavy operations to cloud servers. The system's architecture features three modules—perception, grounding, and alignment—facilitating effective communication to meet the sophisticated needs of 6 G networks.

Besides these frameworks, more specific models are proposed to facilitate the cloud-edge synergy. A common strategy is to use SLM's fast decoding ability. *LLM-to-SLM* [29] proposes a framework in which the pre-trained frozen encoder–decoder LLM resides on the server and computes a high-quality representation of the prompt for the planning of an appropriate response. The SLM residing on the edge device, conditioned on this representation, decodes the response efficiently. Some variants put more emphasis on the reasoning ability of LLMs [170, 239, 305]. In *Synergy of Thoughts* [305], the SLMs generate multiple low-cost reasoning paths. If these paths conflict, the larger LLMs are invoked to provide reflective reasoning and correct any intuitive errors. Hao et al. [129] propose a framework in which an SLM residing on the edge devices generates tokens, calling LLMs to verify and correct threshold-gated "harder" tokens, to achieve a controllable tradeoff between inference quality and cost. *LLMCad* [399] presents an on-device inference engine addressing memory and latency issues in deploying LLMs on mobile devices. It combines a lightweight LM for token generation with a high-precision LLM for verification, leveraging a token tree structure and speculative generation for efficiency. Tested on devices such as Jetson TX2, it achieves up to 9.3× speedup for LLMs with over 10 billion parameters while maintaining accuracy.

Cloud-edge Synergy During Training. CROSSLM [82] introduces a client-server collaborative training framework that preserves data privacy by having clients locally train SLMs instead of fine-tuning LLMs. The framework enables mutual enhancement through a feedback loop where SLMs evaluate LLM-generated synthetic data and provide feedback to improve the LLM's generative capabilities, ensuring high-quality and task-specific data. Concurrently, the synthetic data trains the SLMs, boosting their performance. This cyclical exchange fosters cloud-edge synergy and mutual model improvement.

7.2 Task-Centric Synergy

The advent of LLMs has significantly propelled various natural language processing tasks and inspired research into their synergistic interactions with SLMs to enhance the performance of models tailored for specific tasks. This section introduces scenarios where SLMs exhibit specialized capabilities after fine-tuning and discusses how combining their unique strengths with the versatile abilities of LLMs can yield superior performance on specific tasks. For example, LLMs excel at handling difficult examples or can rewrite content to eliminate task-irrelevant redundancy, thereby enhancing overall task performance, as illustrated in Figures 29–31.

α-UMi [314] introduces a multi-agent framework to enhance tool learning by overcoming the limitations of single-LLM approaches for complex tasks. It utilizes three specialized LMs—planner, caller, and summarizer—as depicted in Figure 29—each handling specific subtasks such as planning, tool invocation, and summarization. This modular design allows the use of small and large open-source LLMs (e.g., LLaMa-7B/12B) and supports easy tool updates. Evaluated on benchmarks like ToolBench [286] and ToolAlpaca [332], α-UMi outperforms traditional single-LLM methods and even exceeds GPT-4 in tool learning performance.

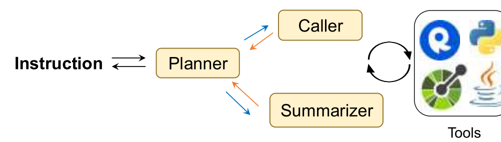


Fig. 29. Synergizing SLMs and LLMs in tool learning.

SynCID [208] focuses on **Conversational Intent Discovery (CID)**, a task where both known and new intents must be identified from user utterances in an open-world setting. SynCID combines LLMs' deep semantic insights with SLMs' agility and specialized capabilities. As illustrated in Figure 30, the framework uses LLM prompting to refine discourse and intent labels, enhancing semantic accuracy and assigning new labels to unlabeled data. SLMs are trained via contrastive learning to align the semantic spaces of discourse and intent descriptors, reducing clustering distortion and improving new intent detection. Tested on BANKING [44], CLINC [182], and StackOverflow [401], SynCID outperforms CID baselines significantly.

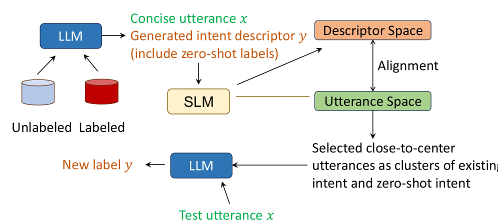


Fig. 30. Synergizing SLMs and LLMs in conversational intent detection.

Filter-then-rerank [239] addresses LLMs' poor performance on simpler IE tasks by integrating LLMs and SLMs. SLMs act as filters, predicting and identifying difficult samples, while LLMs rerank the top N candidate labels for these cases. As illustrated in Figure 31, SLM predictions are final for non-difficult samples, minimizing reliance on LLMs and reducing latency and costs; for those difficult samples, the top N predicted candidate labels from the SLM are passed to the LLM for reranking (predicting). Tested on small-sample IE tasks, this approach improves performance by an average of 2.4% compared to previous methods. **Data Shunt+ (DS+)** [47] introduces a framework to reduce costs by minimizing large model queries during inference and boosting LLM performance with SLMs for tasks like sentiment analysis and image processing. DS+ uses SLMs for "easy" samples within the main training distribution and LLMs for "hard" outliers or boundary cases, maintaining accuracy while reducing LLM use. It incorporates S4L and L4S modules with Prompt Pruning (PP) and 2-stage Confidence Distillation (2CD) for better input processing and knowledge transfer. Tests show DS+ outperforms fine-tuning in accuracy and cost efficiency, significantly cutting down on LLM queries.

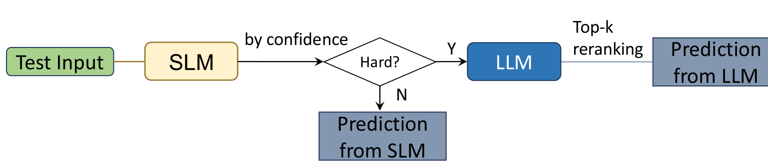


Fig. 31. Synergizing SLMs and LLMs in information extraction.

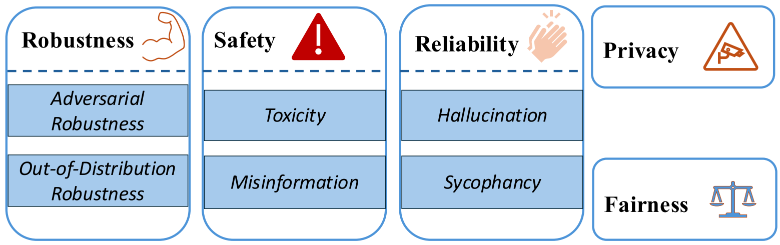


Fig. 32. Scenarios we discuss in this section. The taxonomy is inspired by previous works [326, 357]. Please note that the trustworthy scenarios listed here are not exhaustive.

8 Trustworthiness in SLMs

Language models have become ubiquitous in our daily lives, and we increasingly rely on them. However, they pose risks regarding their limitations in trustworthy dimensions like privacy and fairness. These concerns are especially critical in high-stakes domains such as healthcare [132] and finance [206]. Consequently, numerous studies have emerged to evaluate the trustworthiness of LMs [91, 97, 141, 179, 254, 261, 280, 357, 376, 430]. In this section, we consider the works that benchmark various LMs' trustworthiness and omit the specific attack methods [42, 55, 153, 465] or work [412] that only focuses on early pre-trained LMs like BERT [86] as they are already covered in previous survey papers [81, 117, 125, 295]. Inspired by previous works [326, 357], we discuss the following five key trustworthy scenarios: *robustness, privacy, reliability, safety*, and *fairness*, as shown in Figure 32. We consider two dimensions for robustness: Adversarial (Adv) Robustness [358] and Out-of-Distribution (OOD) Robustness [39, 218]. For safety, we explore two key concerns: Misinformation [350] and Toxicity [379]. For reliability, we focus on Hallucination [149] and Sycophancy [308]. Please note that these are just the aspects we are focusing on, and therefore, this is not a comprehensive classification or taxonomy. For example, robustness also contains robustness to adversarial demonstration.

Though there are a lot of works benchmarking LMs' trustworthiness, their main focus is on LLMs. Therefore, we survey some representative works evaluating the trustworthiness of LMs, focusing specifically on those that include SLMs of around 7B parameters or smaller. We also summarize these works in Table 15. Next, we briefly introduce them.

Holistic Evaluation of Language Models (HELM) [209] benchmarks a large number of LMs from various aspects, including a lot of metrics related to trustworthiness, such as robustness and fairness. Do-Not-Answer [370] introduces a dataset to evaluate how LMs act when they face content that should not be answered. Wang et al. [370] also label the output of several LMs on their dataset and then uses the labeled data to train some classifiers. PromptRobust [460] constructs two kinds of adversarial prompts to evaluate LMs' robustness: One kind is designed under non-adversarial settings with semantic integrity, while another category is created under adversarial settings. Their results show that LMs perform poorly under such prompts. HaluEval [194] builds a dataset comprising both the samples generated by their proposed framework and human-labeled hallucinations. It facilitates analysis of when LMs produce hallucinated output and how well they detect hallucinated content. Then they use some strategies, such as knowledge retrieval, to help LMs better recognize hallucinations. Mo et al. [254] evaluate the trustworthiness of open-source LMs, presenting a variety of scenarios such as fairness and privacy. Results show that smaller LMs sometimes outperform larger ones in terms of trustworthiness. PrivLM-Bench [193] is designed to evaluate the privacy issues in LMs. It enables a fair comparison of privacy-preserving LMs by considering more than just differential privacy parameters. FFT [74] introduces around

Table 15. Comparison of Different Works That Evaluate the Trustworthiness Issues in LMs

Paper	Adv Robustness	OOD Robustness	Toxicity	Misinformation	Hallucination	Sycophancy	Privacy	Fairness	Have Compressed SLMs
HELM [209]	✓	×	✓	✓	×	×	×	✓	×
Do-Not-Answer [370]	×	×	✓	✓	×	×	✓	✓	×
PromptRobust [460]	✓	×	×	×	×	×	×	×	×
HaluEval [194]	×	×	×	×	✓	×	×	×	×
Mo et al. [254]	✓	×	✓	×	✓	✓	✓	✓	×
PrivLM-Bench [193]	×	×	×	×	×	×	✓	×	×
FFT [74]	×	×	✓	✓	✓	×	×	✓	×
ROBBIE [100]	×	×	✓	×	×	×	×	✓	×
TrustLLM [326]	✓	✓	✓	✓	✓	✓	✓	✓	×
RAmBLA [36]	✓	×	×	×	✓	×	×	×	×
JailbreakBench [46]	×	×	✓	✓	×	×	✓	×	×
Xie et al. [395]	×	×	✓	×	✓	×	×	×	×
OR-Bench [73]	×	×	✓	✓	×	×	✓	×	×
SORRY-Bench [392]	×	×	✓	✓	×	×	✓	×	×
BeHonest [61]	×	×	×	✓	✓	✓	×	×	×
Hong et al. [141]	✓	✓	✓	×	×	×	✓	✓	✓
RUPBench [376]	✓	×	×	×	×	×	×	×	×
Nakka et al. [261]	×	×	✓	×	×	×	✓	✓	×

two thousand crafted examples to evaluate LMs' performances on three trustworthy dimensions: factuality, fairness, and toxicity. Their results suggest that larger LMs do not always show better harmlessness. ROBBIE [100] first benchmarks various series of LMs using a lot of datasets, including two newly introduced datasets developed by ROBBIE. It also evaluates mitigation techniques designed to reduce bias and toxicity. TrustLLM [326] is a comprehensive benchmark that contains a large number of datasets and various well-designed metrics to systematically evaluate various LMs across multiple trustworthy dimensions, including truthfulness, safety, fairness, robustness, privacy, and machine ethics. They also carefully design specific subcategories for each dimension. RAmBLA [36] evaluates the trustworthiness of four LMs as biomedical assistants from three dimensions: Robustness, High Recall, and Hallucination. RAmBLA suggests LMs with more parameters are less likely to cause hallucinations and may choose to reject providing an answer in uncertain situations. JailbreakBench [46] constructs a jailbreaking dataset named JBB-Behaviors and jailbreak artifacts to evaluate current LMs' performance regarding jailbreaking. It also proposes a unified evaluation pipeline that can incorporate new jailbreak defense techniques. Xie et al. [395] test online safety analysis methods, filling the gap where no methods focus on the generation phase. OR-Bench [73] constructs three datasets: OR-Bench-80K, OR-Bench-Hard-1K, and OR-Bench-Toxic, to systematically evaluate over-refusal problems in LMs, emphasizing the challenge of balancing safety alignment with the models' usefulness. SORRY-Bench [392] systematically tests 43 different LMs to see how they perform when facing requests that should be refused. They also collect more than annotations created by humans and find that fine-tuned 7B LMs can achieve performance comparable to GPT-4 scale LMs as evaluators. BeHonest [61] evaluates the honesty of LMs from three aspects: Self-Knowledge, Non-Deceptiveness, and Consistency. They use many different metrics for each aspect. For example, the sycophancy rate and the lying rate are adopted in Non-Deceptiveness. The results in both the Self-Knowledge and Consistency parts reveal that larger model sizes generally bring improved performance for the Llama-2 [345] and Llama-3 [96] series. Hong et al. [141] examine the effects of compression methods, including quantization and pruning, on the trustworthiness of language models. They find that pruning and extreme quantization significantly affect the trustworthiness of LMs. RUPBench [376] comprises 15 reasoning datasets designed to assess the performance of LMs both in normal conditions and under various adversarial perturbations. Their results indicate that larger LMs generally demonstrate better resilience to perturbations. Nakka et al. [261] investigate the trust and ethical implications of SLMs deployed on personal devices. It reveals the vulnerabilities of on-device SLMs compared with their on-server counterparts.

Please note that the dimensions discussed in this section reflect only those relevant to our current focus; additional dimensions may be discussed in those works but not listed in Table 15. For example, TrustLLM [326] also explores Machine Ethics.

9 Future Directions

In this section, we offer insights into several promising future research directions that could inspire and motivate the community to address existing gaps in the development of SLMs.

9.1 Developing Efficient SLM Model Architecture

Although Transformers [352] are foundational in most language models, they face significant computational and memory challenges that worsen with model size, impacting training and autoregressive decoding. Recently, Mamba [119] has emerged as a promising alternative, adapting SSMs to dynamically select inputs based on demands, thereby enhancing efficiency. Thereafter, xLSTM [26] demonstrates that an improved LSTM could function as an LLM, revealing the potential of traditional SSMs. The integration of global static information captured by SSMs with the dynamic information processing of Transformers could complement each other, leading to new architectures that balance effectiveness and efficiency.

9.2 Addressing SLM Training Inefficiencies

One study [88] explores the disparate learning dynamics between SLMs and LLMs. Utilizing the Pythia model suite, the research demonstrates that layers' activations in larger models converge more rapidly and monotonically to their final states. This phenomenon is associated with a higher **proportional effective rank (PER)** in the parameters and gradients of larger models. The analysis enhances our understanding of training inefficiencies in small models and provides insights for future efforts, such as developing methods to increase the PER of layers' parameters.

9.3 Expanding Domain-Specific SLMs

Domain-specific SLMs, which are tailored for specific fields, can provide a stronger foundation for relevant downstream tasks than general-purpose models. Currently, these models primarily focus on scientific and healthcare domains. However, there is significant potential for expansion into other key areas such as law, finance, education, telecommunications, and transportation. The scarcity of SLMs that cater to these domains presents an urgent call for research into developing more specialized models.

9.4 Establishing Benchmarking and Leaderboard Platforms for SLMs

Several compelling reasons justify the establishment of benchmarking and leaderboard platforms for SLMs. Firstly, most state-of-the-art SLMs are trained on proprietary datasets, which may include test sets from existing evaluation tasks, presenting challenges for fair capability comparisons. Secondly, many SLMs are designed for specific device applications, significantly differing from general open-domain tasks. Thus, there is a lack of comprehensive benchmarks that accurately reflect SLM performance in specific device applications. For example, SLMs deployed on smartphones often handle tasks sensitive to user data, such as auto-replies based on historical chat texts or GUI context understanding—tasks not typically included in current benchmarks, potentially leading to an underestimation of their importance. Finally, current evaluation tasks focus primarily on metrics like accuracy. Evaluating on-device SLMs involves balancing multiple factors, including overall capabilities, response times, storage and memory usage, power consumption, CPU utilization, additional fine-tuning requirements, and context window constraints, making comprehensive and detailed assessments essential.

9.5 Enhancing SLM Performance and Efficiency

In terms of enhancing SLM performance and efficiency, the efficiency of using teacher LLMs via instruction tuning can be further developed, such as Efficient Instruction Tuning of SLMs from LLMs-generated data, Optimizing Teacher LLM Selection for SLM Learning, and Applying Emerging Techniques from LLMs to SLMs.

— *Efficient Instruction Tuning of SLMs from LLMs-Generated Data.* Enhancing the specialization of SLMs through instruction tuning from LLMs-generated data is crucial, yet finding the most cost-effective instructional strategies remains an underexplored area. Some key areas for exploration are:
  (1) *Instruction Design Adaptability*: The performance of LLMs and SLMs varies significantly with changes in instructions. Therefore, designing tailored instructions that effectively activate relevant sub-competencies and reasoning pathways in SLMs for specific tasks is crucial. This approach would optimize their ability to utilize instructional data, representing a significant future research direction.
  (2) *SLM Capability Adaptability*: Given that SLMs exhibit diverse capabilities across domains, simply supplying extensive data samples for instruction tuning is often inefficient, as SLMs may spend excessive time processing unnecessary data. To optimize efficiency when adapting to specific domains, we suggest first assessing the intrinsic capabilities of an SLM within those domains. Subsequently, one could select appropriate data and activate essential fine-grained capabilities to effectively adapt to domain shifts. This targeted approach ensures efficient and domain-specific instruction tuning.
  (3) *Optimizing Data Efficiency*: SLMs may possess missing or latent domain knowledge, and activating this latent knowledge may not require substantial data. Thus, identifying inherent knowledge within SLMs and determining the minimal data necessary for effective fine-tuning is a future direction. This research aims to optimize performance while minimizing training resources.

— *Optimizing Teacher LLM Selection for SLM Learning*. Teacher LLMs with different abilities and knowledge facilitate diverse applications for SLM training, including data rewriting and generation. Selecting the appropriate teacher model based on specific use cases is crucial. This process requires evaluating the teacher's capabilities and knowledge to ensure optimal application. For example, GPT-4 excels in generating domain-specific data, outperforming ChatGPT, which may produce inferior outcomes. Strategic selection of teacher LLMs is essential for future work to ensure their strengths are effectively utilized to enhance SLM performance.

— *Applying Emerging Techniques from LLMs to SLMs.* To improve LLM performance, techniques such as RAG and Mixture of Experts (MoE) are employed. The adoption of RAG in SLMs shows significant promise [220], suggesting benefits from further tailoring retrieved information for SLMs. Future research should account for SLMs' constraints, such as limited context windows, and customize RAG accordingly. MoE uses multiple experts to enhance learning without increasing active neurons, but its storage demands pose challenges for SLM deployment, making this a promising area for exploration. Additionally, the application of LLM techniques, such as in-context learning and prompt engineering to maximize SLM performance, while accounting for SLMs' constraints, warrants further investigation.

9.6 Applications of SLMs

In real-world applications, SLMs often need to provide personalized services and need to be updated periodically to reflect new needs and new knowledge. Hence, there are several promising directions in terms of the real-world application of SLMs, which are listed as follows:

— *LoRA for Personalized Services.* Companies often provide personalized services, but user-specific complexities can render simple rules ineffective. Training a separate SLM for each user is impractical. LoRA suggests a method of separable training weights alongside fixed original weights, enabling scalable customization. For instance, RecLoRA [459] integrates personalized knowledge into SLMs/LLMs tailored for recommendation tasks by maintaining a set of parallel, independent LoRA weights. This approach effectively customizes language model parameters to align with individual user preferences. This approach is a promising direction that inspires further investigation.

— *Lifelong On-device Learning for Knowledge Injection.* SLMs on devices can access local data without risking data privacy through two main methods. The first method uses retrieval-augmented generation to integrate local data into prompts, requiring SLMs with advanced processing and reasoning capabilities. The second method fine-tunes SLMs with local data, integrating customized knowledge into the model's weights. However, this approach demands significant device resources, including memory and energy. A promising solution is lifelong learning, where SLMs continuously learn and adapt while in use.

— *Strategic Use of SLMs and LLMs in Multi-Agent Systems.* LLMs can function as agents; however, their extensive capabilities are often underutilized in many scenarios, leading to resource wastage. Consequently, strategically routing to appropriately capable SLMs and LLMs within multi-agent systems can optimize cost and functionality.

9.7 Multimodal SLMs

Research on SLMs also includes multimodal data. For example, *SmolVLM* [101] is a compact model that handles image and text inputs to produce text outputs, suitable for on-device use and various multimodal tasks. *SOLO* [56] integrates vision and language processing in a single 7B Transformer model. The limited scope of existing research on multimodal SLMs provides a compelling impetus for researchers to investigate the integration of various modalities, including audio and graphs.

9.8 SLMs Assisting LLMs

In Section 6, we introduced existing works on the use of SLMs to assist LLMs. For instance, EFT [252] emulates fine-tuning on LLMs by leveraging behavior deltas between SLMs' pre-trained and fine-tuned weights to alleviate the time-cost issues associated with fine-tuning LLMs; SlimPLM

[331] detects missing knowledge in LLMs using a slim proxy SLM to accelerate knowledge injection; Contrastive Decoding [202] enhances text quality by maximizing the difference between the log probabilities of an expert LLM and an amateur SLM to mitigate issues of low-quality generation. The research on adopting SLMs to assist LLMs is still in its early stages, with many promising directions yet to be explored. We list some as follows:

- *Enhancing LLM Performance Across Broader Tasks through SLM Integration.* SLMs can outperform LLMs in certain scenarios. For example, SLMs often present fewer security vulnerabilities compared to their larger counterparts and demonstrate superior performance on easier samples in specific tasks [201, 239]. Therefore, integrating SLMs with LLMs can promote the development of models that are not only more robust but also inherently safer. Currently, research in this domain is relatively sparse, suggesting that this collaborative framework could potentially be applied to a wider array of tasks.
- *Efficient Enhancement of LLMs through Proxy SLMs.* Existing research [19, 217, 252, 331] indicates that SLMs, owing to their accelerated fine-tuning and inference speeds, can effectively mimic the behaviors of LLMs, thereby serving as efficient proxies for optimization. However, the application of SLMs as operational proxies for LLMs is currently underexplored. This mimicry could potentially be expanded to include various aspects of LLM functionality, such as the optimization of prompts, the filtration and integration of supplementary knowledge, and the management of additional knowledge repositories.
- *SLMs Assist in Managing Data Quality.* LLMs tend to produce hallucinations and toxic content due to low-quality real-world training data. One solution is to remove these low-quality data [361]. However, directly eliminating low-quality content can diminish certain functionalities of LLMs, such as versatility [362]. Therefore, it is crucial to define more refined data quality assessment criteria across dimensions such as factuality, safety, and diversity [382] for real-world data. Researching efficient data selection methods using small models represents a valuable research direction. Additionally, while synthetic data serves as a vital complement to scarce human-generated data [228], the potential for small models to effectively manage synthetic data remains largely unexplored.
- *SLMs Assist in LLM Assessment.* LLMs are producing vast amounts of increasingly complex texts, such as specialized code and scientific papers, presenting challenges not only for human evaluators but also for traditional assessment metrics. Consequently, developing effective evaluators to assess various aspects of generated content, including factuality, safety, and uncertainty, becomes crucial. Given their proficiency in handling specific tasks, exploring the potential of SLMs to evaluate LLM outputs is a promising research direction.
- *SLMs Optimize Query and Reduce Noise for LLM RAG.* For RAG using LLMs, differing query requirements between LLMs and search engines pose a challenge. The query for LLMs is often abstract and difficult for search engines to handle, so they require more detailed query keywords. Moreover, LLMs may not need all the information related to a query because they only require partial additional knowledge. Thus, intermediate agents are crucial to adapting LLM queries for search engines by clarifying the required detailed keywords that can search for necessary extra knowledge. Additionally, search engine outputs contain noise, requiring refinement to boost LLM efficiency. SLMs, skilled in a single task, are ideal for optimizing query rewriting and noise reduction in RAG systems, making their application in LLM RAG a promising research area.
- *SLMs Safeguard LLM*. Resources such as the Llama 2 Responsible Use Guide strongly advocate for the implementation of robust guardrails in products that utilize Generative AI. SLMs can be strategically designed to serve as such guardrails, mitigating risks associated with both

inputs and outputs from the model. This approach ensures safeguards against the generation of high-risk or policy-violating content and provides protection against adversarial inputs and attempts to compromise the model. Future research could explore how SLMs can enhance the safety of LLMs by providing protection against various threats, including adversarial attacks, jailbreak attacks, and backdoor attacks.

9.9 Synergy between SLMs and LLMs

In Section 7, we discussed how SLMs and LLMs can complement each other. For example, CoGenesis [442] integrates SLMs for private data and LLMs for broader context, while Synergy of Thoughts [305] uses SLMs for initial reasoning and LLMs for conflict resolution. CROSSLM [82] shows how privacy can be preserved by training SLMs locally to support LLMs without data exposure. Research in this area is still evolving, and we outline several promising future directions below:

— *Refined Cloud-Edge Division of Labor.* Current research mainly focuses on splitting tasks between edge-based SLMs and cloud-based LLMs along privacy-sensitive and non-sensitive data boundaries. A potential future direction involves more granular task partitioning: determining which subtasks should be handled locally by SLMs (e.g., initial data filtering, quick semantic parsing) and which should be delegated to the cloud-based LLM (e.g., advanced reasoning, complex generation). This approach can further optimize latency, privacy, and resource utilization.
— *Adaptive On-Device Specialization for Dynamic Environments.* Although SLMs have shown the ability to handle private or personalized data locally, continuous changes in user preferences, application requirements, and data distributions pose challenges. Future work can explore adaptive strategies where edge-based SLMs dynamically specialize or update their parameters, guided by the cloud-based LLM. For instance, the LLM can periodically distill new knowledge into the SLM or provide feedback signals to help the SLM adapt to evolving scenarios.

9.10 Trustworthy SLMs

As SLMs are playing crucial roles in various aspects, understanding and improving the trustworthiness of SLMs are essential. Hence, two promising directions are:

— *A Comprehensive Evaluation of SLMs' Trustworthiness.* While numerous studies address trustworthiness issues in LLMs, research on SLMs remains sparse. Most existing literature focuses on models with at least 7 billion parameters, leaving a gap in the comprehensive analysis of SLMs' trustworthiness. Current evaluations typically cover only a fraction of the necessary aspects. Therefore, a systematic assessment, such as TrustLLM [326], is essential to thoroughly evaluate the trustworthiness of SLMs and understand their reliability across various applications.
— *Developing Trustworthy SLMs.* Developing trustworthy SLMs is crucial, with three key research directions: (i) Training SLMs to be trustworthy from scratch; (ii) Ensuring SLMs retain or gain trustworthiness when compressed from LLMs—maintaining trustworthiness if the LLM is trustworthy and instilling trustworthiness if it is not; and (iii) Fine-tuning non-trustworthy SLMs to enhance their robustness.

10 Conclusion

This article provides a comprehensive survey of SLMs with up to 7 billion parameters. Initially, we address the need to clearly define SLMs due to existing ambiguities in their characterization. We then present the foundational concepts essential for constructing SLMs. The survey progresses to explore enhancement techniques, including KD and quantization, as well as strategies for adapting

LLMs to SLM contexts. We survey representative SLMs, both general-domain and domain-specific, discussing their preferred datasets and architectural decisions. We also assess their applications across various tasks and deployment strategies on devices. Further, we investigate their role in augmenting the capabilities of LLMs, serving as proxies for fine-tuning and facilitating two types of synergies: cloud-local and task-centric. Additionally, we discuss the critical aspect of their trustworthiness. The article concludes with key insights aimed at guiding future research on SLMs.

References

[1] Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. arXiv:2404.14219. Retrieved from https://arxiv.org/abs/2404.14219

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 Technical Report. arXiv:2303.08774. Retrieved from https://arxiv.org/abs/2303.08774

[3] Emre Can Acikgoz, Osman Batur İnce, Rayene Bench, Arda Anıl Boz, İlker Kesen, Aykut Erdem, and Erkut Erdem. 2024. Hippocrates: An open-source framework for advancing large language models in healthcare. arXiv:2404.16621. Retrieved from https://arxiv.org/abs/2404.16621

[4] Harshavardhan Adepu, Zhanpeng Zeng, Li Zhang, and Vikas Singh. 2024. FrameQuant: Flexible low-bit quantization for transformers. arXiv:2403.06082. Retrieved from https://arxiv.org/abs/2403.06082

[5] Abien Fred Agarap. 2018. Deep learning using rectified linear units (ReLU). arXiv:1803.08375. Retrieved from http://arxiv.org/abs/1803.08375

[6] Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. 2024. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations (ICLR '24)*.

[7] Meta AI. 2024. Llama 3.2: Revolutionizing edge AI and vision with open, customizable models. Retrieved from https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/. Accessed: September 25, 2024.

[8] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. GQA: Training generalized multi-query transformer models from multi-head checkpoints. arXiv:2305.13245. Retrieved from https://arxiv.org/abs/2305.13245

[9] Ali Al-Lawati, Jason Lucas, Zhiwei Zhang, Prasenjit Mitra, and Suhang Wang. 2025. Graph-based molecular in-context learning grounded on morgan fingerprints. arXiv:2502.05414. Retrieved from https://arxiv.org/abs/2502.05414

[10] Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Leandro von Werra, and Thomas Wolf. 2024. SmolLM - blazingly fast and remarkably powerful. arXiv:2409.00286v1. Retrieved from https://arxiv.org/abs/2409.00286v1

[11] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. 2023. The falcon series of open language models. arXiv:2311.16867. Retrieved from https://arxiv.org/abs/2311.16867

[12] Guilherme F. C. F. Almeida, José Luiz Nunes, Neele Engelmann, Alex Wiegmann, and Marcelo de Araújo. 2024. Exploring the psychology of LLMs' moral and legal reasoning. *Artif. Intell.* 333, (2024), 104145. DOI: https://doi.org/10.1016/j.artint.2024.104145

[13] Reza Yazdani Aminabadi, Samyam Rajbhandari, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Olatunji Ruwase, Shaden Smith, Minjia Zhang, Jeff Rasley, et al. 2022. Deepspeed-inference: Enabling efficient inference of transformer models at unprecedented scale. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–15.

[14] Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. 2024. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38, AAAI, 10865–10873.

[15] Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 Technical Report. arXiv:2305.10403. Retrieved from https://arxiv.org/abs/2305.10403

[16] AI Anthropic. 2024. The Claude 3 model family: Opus, Sonnet, Haiku. *Claude-3 Model Card*, 1, (2024). Retrieved from https://api.semanticscholar.org/CorpusID:268232499

[17] David Anugraha, Genta Indra Winata, Chenyue Li, Patrick Amadeus Irawan, and En-Shiun Annie Lee. 2024. ProxyLM: Predicting language model performance on multilingual tasks via proxy models. arXiv:2406.09334. Retrieved from https://arxiv.org/abs/2406.09334

[18] Viraat Aryabumi, Yixuan Su, Raymond Ma, Adrien Morisot, Ivan Zhang, Acyr Locatelli, Marzieh Fadaee, Ahmet Üstün, and Sara Hooker. 2024. To code, or not to code? Exploring impact of code in pre-training. arXiv:2408.10914. Retrieved from https://arxiv.org/abs/2408.10914

[19] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=hSyW5go0v8
[20] Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024. SliceGPT: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=vXxardq6db
[21] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. arXiv:2108.07732. Retrieved from https://arxiv.org/abs/2108.07732
[22] Amos Azaria, and Tom Mitchell. 2023. The internal state of an LLM knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP '23*. Association for Computational Linguistics, 967–976.
[23] Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2023. Llemma: An open language model for mathematics. arXiv:2310.10631. Retrieved from https://arxiv.org/abs/2310.10631
[24] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen Technical Report. arXiv:2309.16609. Retrieved from https://arxiv.org/abs/2309.16609
[25] Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. 2020. The pushshift reddit dataset. In *Proceedings of the International AAAI Conference on Web and Social Media*, Vol.14, AAAI, 830–839.
[26] Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael K. Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. 2024. xLSTM: Extended long Short-Term memory. In *The Thirty-Eighth Annual Conference on Neural Information Processing Systems*. Retrieved from https://openreview.net/forum?id=ARAxPPIAhq
[27] Marco Bellagente, Jonathan Tow, Dakota Mahan, Duy Phung, Maksym Zhuravinskyi, Reshinth Adithyan, James Baicoianu, Ben Brooks, Nathan Cooper, Ashish Datta, et al. 2024. Stable LM 2 1.6B Technical Report. arXiv:2402.17834. Retrieved from https://arxiv.org/abs/2402.17834
[28] Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. 2024. *SmolLM-Corpus*. Retrieved from https://huggingface.co/datasets/HuggingFaceTB/smollm-corpus
[29] Benjamin Bergner, Andrii Skliar, Amelie Royer, Tijmen Blankevoort, Yuki Asano, and Babak Ehteshami Bejnordi. 2024. Think big, generate quick: LLM-to-SLM for fast autoregressive decoding. arXiv:2402.16844. Retrieved from https://arxiv.org/abs/2402.16844
[30] Milan Bhan, Jean-Noel Vittaut, Nicolas Chesneau, and Marie-Jeanne Lesot. 2024. Self-AMPLIFY: Improving small language models with self post Hoc explanations. arXiv:2402.12038. Retrieved from https://arxiv.org/abs/2402.12038
[31] Zhen Bi, Ningyu Zhang, Yida Xue, Yixin Ou, Daxiong Ji, Guozhou Zheng, and Huajun Chen. 2023. OceanGPT: A large language model for ocean science tasks. arXiv:2310.02031. Retrieved from https://arxiv.org/abs/2310.02031
[32] Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. arXiv:230401373. Retrieved from https://arxiv.org/abs/2304.01373
[33] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34, AAAI, 7432–7439.
[34] Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. https://doi.org/10.5281/zenodo.5297715.
[35] Elliot Bolton, Abhinav Venigalla, Michihiro Yasunaga, David Hall, Betty Xiong, Tony Lee, Roxana Daneshjou, Jonathan Frankle, Percy Liang, Michael Carbin, et al. 2024. BioMedLM: A 2.7 b parameter language model trained on biomedical text. arXiv:2403.18421. Retrieved from https://arxiv.org/abs/2403.18421
[36] William James Bolton, Rafael Poyiadzi, Edward R. Morrell, Gabriela van Bergen, Gonzalez Bueno, and Lea Goetz. 2024. RAmBLA: A framework for evaluating the reliability of LLMs as assistants in the biomedical domain. arXiv:2403.14578. Retrieved from https://arxiv.org/abs/2403.14578
[37] Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. 2022. Inpars: Data augmentation for information retrieval using large language models. arXiv:2202.05144. Retrieved from https://arxiv.org/abs/2202.05144
[38] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf
[39] Saikiran Bulusu, Bhavya Kailkhura, Bo Li, Pramod K. Varshney, and Dawn Song. 2020. Anomalous example detection in deep learning: A survey. *IEEE Access* 8, (2020), 132330–132347. DOI: https://doi.org/10.1109/ACCESS.2020.3010274

[40] Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. 2025. A survey on mixture of experts. *IEEE Transactions on Knowledge & Data Engineering* 37, 7 (2025), 3896–3915.

[41] Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, et al. 2024. InternLM2 Technical Report. arXiv:2403.17297. Retrieved from https://arxiv.org/abs/2403.17297

[42] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security '21)*. JMLR, 2633–2650.

[43] Samuel Carreira, Tomás Marques, José Ribeiro, and Carlos Grilo. 2023. Revolutionizing mobile interaction: Enabling a 3 billion parameter GPT LLM on mobile. arXiv:231001434. Retrieved from https://arxiv.org/abs/2310.01434

[44] Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*. Association for Computational Linguistics, Online, 38–45.

[45] Wei-Cheng Chang, X. Yu Felix, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training tasks for embedding-based large-scale retrieval. In *International Conference on Learning Representations*.

[46] Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramer, et al. 2024. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. arXiv:2404.01318. Retrieved from https://arxiv.org/abs/2404.01318

[47] Dong Chen, Shuo Zhang, Yueting Zhuang, Siliang Tang, Qidong Liu, Hua Wang, and Mingliang Xu. 2024. Improving large models with small models: Lower costs and better performance. arXiv:2406.15471. Retrieved from https://arxiv.org/abs/2406.15471

[48] Dong Chen, Yueting Zhuang, Shuo Zhang, Jinfeng Liu, Su Dong, and Siliang Tang. 2024. Data shunt: Collaboration of small and large models for lower costs and better performance. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38, AAAI, 11249–11257.

[49] Hardy Chen, Haoqin Tu, Fali Wang, Hui Liu, Xianfeng Tang, Xinya Du, Yuyin Zhou, and Cihang Xie. 2025. SFT or RL? an early investigation into training R1-like reasoning large vision-language models. arXiv:2504.11468. Retrieved from https://arxiv.org/abs/2504.11468

[50] Hongzhan Chen, Siyue Wu, Xiaojun Quan, Rui Wang, Ming Yan, and Ji Zhang. 2023. MCC-KD: Multi-CoT consistent knowledge distillation. In *Findings of the Association for Computational Linguistics: EMNLP '23*. Association for Computational Linguistics, 6805–6820.

[51] Lihu Chen, and Gaël Varoquaux. 2024. What is the role of small models in the LLM era: A survey. arXiv:2409.06857. Retrieved from https://arxiv.org/abs/2409.06857

[52] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. arXiv:2107.03374. Retrieved from https://arxiv.org/abs/2107.03374

[53] Tianyi Chen, Tianyu Ding, Badal Yadav, Ilya Zharkov, and Luming Liang. 2023. Lorashear: Efficient large language model structured pruning and knowledge recovery. arXiv:2310.18356. Retrieved from https://arxiv.org/abs/2310.18356

[54] Wei Chen, Zhiyuan Li, and Mingyuan Ma. 2024. Octopus: On-device language model for function calling of software APIs. arXiv:240401549. Retrieved from https://arxiv.org/abs/2404.01549

[55] Yangyi Chen, Fanchao Qi, Hongcheng Gao, Zhiyuan Liu, and Maosong Sun. 2022. Textual backdoor attacks can Be more harmful via two simple tricks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP*. Springer-Verlag, 11215–11221.

[56] Yangyi Chen, Xingyao Wang, Hao Peng, and Heng Ji. 2024. A single transformer for scalable vision-language modeling. arXiv:2407.06438. Retrieved from https://arxiv.org/abs/2407.06438

[57] Zeming Chen, Alejandro Hernández Cano, Angelika Romanou, Antoine Bonnet, Kyle Matoba, Francesco Salvi, Matteo Pagliardini, Simin Fan, Andreas Köpf, Amirkeivan Mohtashami, et al. 2023. MEDITRON-70B: Scaling medical pretraining for large language models. arXiv:2311.16079. Retrieved from https://arxiv.org/abs/2311.16079

[58] Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan R. Routledge, et al. 2021. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. ACM, 3697–3711.

[59] Zhiyu Chen, Shiyang Li, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang. 2022. ConvFinQA: Exploring the chain of numerical reasoning in conversational finance question answering. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 6279–6292.

[60] Xiaoxue Cheng, Junyi Li, Wayne Xin Zhao, Hongzhi Zhang, Fuzheng Zhang, Di Zhang, Kun Gai, and Ji-Rong Wen. 2024. Small agent can also rock! Empowering small language models as hallucination detector. arXiv:2406.11277. Retrieved from https://arxiv.org/abs/2406.11277

[61] Steffi Chern, Zhulin Hu, Yuqing Yang, Ethan Chern, Yuan Guo, Jiahe Jin, Binjie Wang, and Pengfei Liu. 2024. BeHonest: Benchmarking honesty of large language models. arXiv:2406.13261. Retrieved from https://arxiv.org/abs/2406.13261
[62] Yew Ken Chia, Pengfei Hong, Lidong Bing, and Soujanya Poria. 2024. InstructEval: Towards holistic evaluation of Instruction-Tuned large language models. In *Proceedings of the First Edition of the Workshop on the Scaling Behavior of Large Language Models (SCALE-LLM 2024)*. Association for Computational Linguistics, 35–64.
[63] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. Retrieved from https://lmsys.org/blog/2023-03-30-vicuna/
[64] Yae Jee Cho, Luyang Liu, Zheng Xu, Aldi Fahrezi, and Gauri Joshi. 2024. Heterogeneous LoRA for federated fine-tuning of on-device foundation models. arXiv:240106432. Retrieved from https://arxiv.org/abs/2401.06432
[65] Xiaokai Chu, Jiashu Zhao, Lixin Zou, and Dawei Yin. 2022. H-ERNIE: A multi-granularity pre-trained language model for web search. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1478–1489.
[66] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *J. Mach. Learn. Res*. 25, 70 (2024), 1–53. DOI: https://doi.org/10.5555/3722577.3722647
[67] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. arXiv:1905.10044. Retrieved from https://arxiv.org/abs/1905.10044
[68] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. arXiv:1803.05457. Retrieved from https://arxiv.org/abs/1803.05457
[69] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. arXiv:2110.14168. Retrieved from https://arxiv.org/abs/2110.14168
[70] Together Computer. 2023. RedPajama: An Open Dataset for Training Large Language Models. Retrieved from https://github.com/togethercomputer/RedPajama-Data
[71] Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free Dolly: Introducing the World's First Truly Open Instruction-Tuned LLM. Retrieved from https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm
[72] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 deep learning track. arXiv:2102.07662. Retrieved from https://arxiv.org/abs/2102.07662
[73] Justin Cui, Wei-Lin Chiang, Ion Stoica, and Cho-Jui Hsieh. 2024. OR-bench: An over-refusal benchmark for large language models. arXiv:2405.20947. Retrieved from https://arxiv.org/abs/2405.20947
[74] Shiyao Cui, Zhenyu Zhang, Yilong Chen, Wenyuan Zhang, Tianyun Liu, Siqi Wang, and Tingwen Liu. 2023. Fft: Towards harmlessness evaluation and analysis for LLMS with factuality, fairness, toxicity. arXiv:2311.18580. Retrieved from https://arxiv.org/abs/2311.18580
[75] Luigi Daniele, and Suphava Deeprasit. 2023. Amplify-Instruct: Synthetically generated diverse multi-turn conversations for efficient LLM training. arXiv Preprint. Retrieved from https://huggingface.co/datasets/LDJnr/Capybara
[76] Tri Dao. 2024. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*.
[77] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Adv. Neural Inf. Process. Syst*. 35, (2022), 16344–16359.
[78] Tri Dao, and Albert Gu. 2024. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. arXiv:2405.21060. Retrieved from https://arxiv.org/abs/2405.21060
[79] Rocktim Jyoti Das, Liqun Ma, and Zhiqiang Shen. 2023. Beyond size: How gradients shape pruning decisions in large language models. arXiv:2311.04902. Retrieved from https://arxiv.org/abs/2311.04902
[80] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. 2011. Fast locality-sensitive hashing. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1073–1081.
[81] Pieter Delobelle, Ewoenam Kwaku Tokpo, Toon Calders, and Bettina Berendt. 2022. Measuring fairness with biased rulers: A comparative study on bias metrics for pre-trained language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 1693–1706.
[82] Yongheng Deng, Ziqing Qiao, Ju Ren, Yang Liu, and Yaoxue Zhang. 2023. Mutual enhancement of large and small language models with cross-silo knowledge transfer. arXiv:2312.05842. Retrieved from https://arxiv.org/abs/2312.05842

[83] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. GPT3.int8(): 8-bit Matrix Multiplication for Transformers at Scale. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). Retrieved from https://openreview.net/forum?id=dXiGWqBoxaD.

[84] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Adv. Neural Inf. Process. Syst*. 36, (2024). DOI: https://doi.org/10.48550/arXiv.2305.14314

[85] Tim Dettmers, and Luke Zettlemoyer. 2023. The case for 4-bit precision: K -bit inference scaling laws. In *International Conference on Machine Learning*. PMLR, 7750–7774.

[86] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Vol. 1 (Long and Short Papers). Association for Computational Linguistics, 4171–4186.

[87] Nolan Dey, Gurpreet Gosal, Zhiming Chen, Hemant Khachane, William Marshall, Ribhu Pathria, Marvin Tom, and Joel Hestness. 2023. Cerebras-GPT: Open Compute-Optimal Language Models Trained on the Cerebras Wafer-Scale Cluster. CoRR abs/2304.03208.

[88] Richard Diehl Martinez, Pietro Lesci, and Paula Buttery. 2024. Tending Towards Stability: Convergence Challenges in Small Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Yaser* Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, 3275–3286. https://doi.org/10.18653/v1/2024.findings-emnlp.187

[89] Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. arXiv:2305.14233. Retrieved from https://arxiv.org/abs/2305.14233

[90] Tinghe Ding. 2024. MobileAgent: Enhancing mobile control via human-machine interaction and SOP integration. arXiv:240104124. Retrieved from https://arxiv.org/abs/2401.04124

[91] Ricardo Dominguez-Olmedo, Moritz Hardt, and Celestine Mendler-Dünner. 2023. Questioning the survey responses of large language models. arXiv:2306.07951. Retrieved from https://arxiv.org/abs/2306.07951

[92] Qian Dong, Yiding Liu, Qingyao Ai, Haitao Li, Shuaiqiang Wang, Yiqun Liu, Dawei Yin, and Shaoping Ma. 2023. I3 retriever: Incorporating implicit interaction in pre-trained language models for passage retrieval. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. ACM, 441–451.

[93] Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, et al. 2024. Hymba: A hybrid-head architecture for small language models. arXiv:2411.13676. Retrieved from https://arxiv.org/abs/2411.13676

[94] Xinrun Du, Zhouliang Yu, Songyang Gao, Ding Pan, Yuyang Cheng, Ziyang Ma, Ruibin Yuan, Xingwei Qu, Jiaheng Liu, Tianyu Zheng, Xinchen Luo, Guorui Zhou, Wenhu Chen, and Ge Zhang. 2024. Chinese tiny LLM: Pretraining a Chinese-centric large language model. arXiv:240404167. Retrieved from https://arxiv.org/abs/2404.04167

[95] Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. GLM: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics* Vol. 1 Long Papers, Association for Computational Linguistics, 320–335.

[96] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The Llama 3 herd of models. arXiv:2407.21783. Retrieved from https://arxiv.org/abs/2407.21783

[97] Kazuki Egashira, Mark Vero, Robin Staab, Jingxuan He, and Martin Vechev. 2024. Exploiting LLM quantization. arXiv:2405.18137. Retrieved from https://arxiv.org/abs/2405.18137

[98] Ronen Eldan, and Yuanzhi Li. 2023. Tinystories: How small can language models be and still speak coherent English?. arXiv:2305.07759. Retrieved from https://arxiv.org/abs/2305.07759

[99] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. 2018. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Netw.: Off. J. Int. Neural Netw. Soc*. 107, (2018), 3–11. DOI: https://doi.org/10.1016/j.neunet.2017.12.012

[100] David Esiobu, Xiaoqing Tan, Saghar Hosseini, Megan Ung, Yuchen Zhang, Jude Fernandes, Jane Dwivedi-Yu, Eleonora Presani, Adina Williams, and Eric Smith. 2023. ROBBIE: Robust bias evaluation of large generative language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 3764–3814. Retrieved from https://aclanthology.org/2023.emnlp-main.230.

[101] Hugging Face. 2024. *SmolVLM - small yet mighty Vision Language Model*. Retrieved from https://huggingface.co/blog/smolvlm. Accessed: November 26, 2024.

[102] William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res*. 23, 120 (2022), 1–39. DOI: https://doi.org/10.48550/arXiv.2101.03961

[103] Shangbin Feng, Weijia Shi, Yuyang Bai, Vidhisha Balachandran, Tianxing He, and Yulia Tsvetkov. 2023. Knowledge card: Filling LLMs' knowledge gaps with plug-in specialized language models. arXiv:2305.09955. Retrieved from https://arxiv.org/abs/2305.09955

[104] Elias Frantar, and Dan Alistarh. 2023. SparseGPT: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*. PMLR, 10323–10337.

[105] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*.

[106] Hao Fu, Yao Peng, and Tushar Khot. 2022. How does GPT Obtain its Ability? Tracing Emergent Abilities of Language Models to their Sources. *Yao Fu's Notion* (Dec. 2022) Retrieved from https://yaofu.notion.site/How-does-GPT-Obtain-its-Ability-Tracing-Emergent-Abilities-of-Language-Models-to-their-Sources-b9a57ac0fcf74f30a1ab9e3e36fa1dc1

[107] Yao Fu, Hao Peng, and Litu Ou. Ashish sabharwal, and tushar khot. 2023. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning*. PMLR, 10421–10430.

[108] Chongming Gao, Shiqi Wang, Shijun Li, Jiawei Chen, Xiangnan He, Wenqiang Lei, Biao Li, Yuan Zhang, and Peng Jiang. 2023. CIRS: Bursting filter bubbles by counterfactual interactive recommender system. *ACM Trans. Inf. Syst*. 42, 1 (2023), 1–27. DOI: https://doi.org/10.1145/3594871

[109] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The Pile: An 800GB dataset of diverse text for language modeling. arXiv:2101.00027. Retrieved from https://arxiv.org/abs/2101.00027

[110] Luyu Gao, and Jamie Callan. 2022. Unsupervised corpus aware language model pre-training for dense passage retrieval. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, Vol. 1 Long Papers, Association for Computational Linguistics, 2843–2853.

[111] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, et al. 2024. The Language Model Evaluation Harness. DOI: https://doi.org/10.5281/zenodo.12608602

[112] Shangqian Gao, Chi-Heng Lin, Ting Hua, Zheng Tang, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. 2024. DISP-LLM: Dimension-Independent structural pruning for large language models. In *The Thirty-Eighth Annual Conference on Neural Information Processing Systems*. Retrieved from https://openreview.net/forum?id=YxaY6tHgg0

[113] Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. 2024. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *The Twelfth International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=uNrFpDPMyo

[114] Alex Gichamba, Tewodros Kederalah Idris, Brian Ebiyau, Eric Nyberg, and Teruko Mitamura. 2024. ColBERT retrieval and ensemble response scoring for language model question answering. arXiv:240810808. Retrieved from https://arxiv.org/abs/2408.10808

[115] Karan Goel. 2024. The OnDevice Intelligence Update. Retrieved from https://www.cartesia.ai/blog/on-device

[116] Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. 2019. Openwebtext corpus.

[117] Shreya Goyal, Sumanth Doddapaneni, Mitesh M. Khapra, and Balaraman Ravindran. 2023. A survey of adversarial defenses and robustness in nlp. *Comput. Surveys* 55, 14s (2023), 1–39.

[118] Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. 2024. OLMo: Accelerating the science of language models. arXiv:2402.00838. Retrieved from https://arxiv.org/abs/2402.00838

[119] Albert Gu, and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. arXiv:2312.00752. Retrieved from https://arxiv.org/abs/2312.00752

[120] Naibin Gu, Peng Fu, Xiyu Liu, Bowen Shen, Zheng Lin, and Weiping Wang. 2024. Light-PEFT: Lightening Parameter-Efficient Fine-Tuning via Early Pruning. In *Findings of the Association for Computational Linguistics: ACL 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, 7528–7541. DOI: https://doi.org/10.18653/v1/2024.findings-acl.447

[121] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*.

[122] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. 2023. Textbooks are all you need. arXiv:230611644. Retrieved from https://arxiv.org/abs/2306.11644

[123] Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, Y. K. Li, et al. 2024. DeepSeek-Coder: When the Large Language Model Meets Programming–The Rise of Code Intelligence. *arXiv preprint arXiv:2401.14196*.

[124] Jinyang Guo, Jianyu Wu, Zining Wang, Jiaheng Liu, Ge Yang, Yifu Ding, Ruihao Gong, Haotong Qin, and Xianglong Liu. 2024. Compressing large language models by joint sparsification and quantization. In *Forty-First International Conference on Machine Learning*.

[125] Shangwei Guo, Chunlong Xie, Jiwei Li, Lingjuan Lyu, and Tianwei Zhang. 2022. Threats to pre-trained language models: Survey and taxonomy. *arXiv preprint arXiv:2202.06862*.

[126] Song Guo, Jiahang Xu, Li Lyna Zhang, and Mao Yang. 2023. Compresso: Structured pruning with collaborative prompting learns compact large language models. *arXiv preprint arXiv:2310.05015*.

[127] Zhen Guo, Peiqi Wang, Yanwei Wang, and Shangdi Yu. 2023. Improving small language models on PubMedQA via generative data augmentation. arXiv:230507804. Retrieved from https://arxiv.org/abs/2305.07804

[128] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Adv. Neural Inf. Process. Syst.*, 28 (2015), 1135–1143.

[129] Zixu Hao, Huiqiang Jiang, Shiqi Jiang, Ju Ren, and Ting Cao. 2024. Hybrid SLM and LLM for edge-cloud collaborative inference. In *Proceedings of the Workshop on Edge and Mobile Foundation Models*. ACM, 36–41.

[130] Tim Hartill, Diana Benavides-Prado, Michael Witbrock, and Patricia J. Riddle. 2023. Answering unseen questions with smaller language models using rationale generation and dense retrieval. arXiv:230804711. Retrieved from https://arxiv.org/abs/2308.04711

[131] Tim Hartill, Neset Tan, Michael Witbrock, and Patricia J. Riddle. 2023. Teaching smaller language models to generalise to unseen compositional questions. arXiv:230800946. Retrieved from https://arxiv.org/abs/2308.00946

[132] Kai He, Rui Mao, Qika Lin, Yucheng Ruan, Xiang Lan, Mengling Feng, and Erik Cambria. 2023. A survey of large language models for healthcare: From data, technology, and applications to accountability and ethics. arXiv:2310.05694. Retrieved from https://arxiv.org/abs/2310.05694

[133] Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=sE7-XhLxHA

[134] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. arXiv:2006.03654. Retrieved from https://arxiv.org/abs/2006.03654

[135] Narges Heidari, Parham Moradi, and Abbas Koochari. 2022. An attention-based deep learning method for solving the cold-start and sparsity issues of recommender systems. *Knowl.-Based Syst*. 256 (2022), 109835. DOI: https://doi.org/10.1016/j.knosys.2022.109835

[136] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. arXiv:2009.03300. Retrieved from https://arxiv.org/abs/2009.03300

[137] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=d7KBjmI3GmQ

[138] Dan Hendrycks, and Kevin Gimpel. 2016. Gaussian error linear units (gelus). arXiv:1606.08415. Retrieved from https://arxiv.org/abs/1606.08415

[139] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. arXiv:1503.02531. Retrieved from https://arxiv.org/abs/1503.02531

[140] Sepp Hochreiter, and Jürgen Schmidhuber. 1996. LSTM can solve hard long time lag problems. *Adv. Neural Inf. Process. Syst*. 9, (1996), 473–479. DOI: https://doi.org/10.5555/2998981.2999048

[141] Junyuan Hong, Jinhao Duan, Chenhui Zhang, Zhangheng Li, Chulin Xie, Kelsey Lieberman, James Diffenderfer, Brian R. Bartoldson, Ajay Kumar Jaiswal, Kaidi Xu, et al. 2024. Decoding compressed trust: Scrutinizing the trustworthiness of efficient LLMs under compression. In *Proceedings of the Forty-First International Conference on Machine Learning, ICML*. Retrieved from https://openreview.net/forum?id=e3Dpq3WdMv

[142] Yutong Meng Yuhao Wang Hongcheng Liu, and Yusheng Liao. 2023. XieZhi: Chinese law large language model. Retrieved from https://github.com/LiuHC0428/LAW_GPT

[143] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*. PMLR, 2790–2799.

[144] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL '23*. Association for Computational Linguistics, 8003–8017.

[145] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. arXiv:2106.09685. Retrieved from https://arxiv.org/abs/2106.09685

[146] Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. 2024. Minicpm: Unveiling the potential of small language models with scalable training strategies. arXiv:2404.06395. Retrieved from https://arxiv.org/abs/2404.06395

[147] Xing Hu, Yuan Chen, Dawei Yang, Sifan Zhou, Zhihang Yuan, Jiangyong Yu, and Chen Xu. 2024. I-LLM: Efficient integer-only inference for fully-quantized low-bit large language models. arXiv:2405.17849. Retrieved from https://arxiv.org/abs/2405.17849

[148] Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023. Large language models can Self-Improve. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

[149] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. arXiv:2311.05232. Retrieved from https://arxiv.org/abs/2311.05232

[150] Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and Xiaojuan Qi. 2024. Billm: Pushing the limit of post-training quantization for LLMS. arXiv:2402.04291. Retrieved from https://arxiv.org/abs/2402.04291

[151] Wenyu Huang, Guancheng Zhou, Hongru Wang, Pavlos Vougiouklis, Mirella Lapata, and Jeff Z. Pan. 2024. Less is More: Making Smaller Language Models Competent Subgraph Retrievers for Multi-hop KGQA. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, 15787–15803. https://doi.org/10.18653/v1/2024.findings-emnlp.927

[152] Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, et al. 2024. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Adv. Neural Inf. Process. Syst*. 36 (2024), 62991–63010.

[153] Yangsibo Huang, Samyak Gupta, Mengzhou Xia, Kai Li, and Danqi Chen. 2023. Catastrophic jailbreak of open-source LLMs via exploiting generation. arXiv:2310.06987. Retrieved from https://arxiv.org/abs/2310.06987

[154] Yuheng Huang, Jiayang Song, Zhijie Wang, Shengming Zhao, Huaming Chen, Felix Juefei-Xu, and Lei Ma. 2023. Look before you leap: An exploratory study of uncertainty measurement for large language models. arXiv:2307.10236. Retrieved from https://arxiv.org/abs/2307.10236

[155] Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2020. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *International Conference on Learning Representations*.

[156] Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. 2023. Llama guard: LLM-based input-output safeguard for human-AI conversations. arXiv:2312.06674. Retrieved from https://arxiv.org/abs/2312.06674

[157] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive mixtures of local experts. *Neural Comput*. 3, 1 (1991), 79–87.

[158] Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, et al. 2023. Phi-2: The surprising power of small language models. In *Microsoft Research Blog*.

[159] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Hwang, and Jong C. Park. 2023. Test-Time Self-Adaptive small language models for question answering. In *Findings of the Association for Computational Linguistics: EMNLP '23*. Association for Computational Linguistics, 15459–15469.

[160] Ananya Harsh Jha, Tom Sherborne, Evan Pete Walsh, Dirk Groeneveld, Emma Strubell, and Iz Beltagy. 2024. Just CHOP: Embarrassingly simple LLM compression. arXiv:230514864. Retrieved from https://arxiv.org/abs/2305.14864

[161] Yixin Ji, Yang Xiang, Juntao Li, Wei Chen, Zhongyi Liu, Kehai Chen, and Min Zhang. 2024. Feature-based low-rank compression of large language models via Bayesian optimization. arXiv:2405.10616. Retrieved from https://arxiv.org/abs/2405.10616

[162] Ziwei Ji, Tiezheng Yu, Yan Xu, Nayeon Lee, Etsuko Ishii, and Pascale Fung. 2023. Towards mitigating LLM hallucination via self reflection. In *Findings of the Association for Computational Linguistics: EMNLP '23*. Association for Computational Linguistics, 1827–1843.

[163] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7B. arXiv:2310.06825. Retrieved from https://arxiv.org/abs/2310.06825

[164] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. arXiv:2401.04088. Retrieved from https://arxiv.org/abs/2401.04088

[165] Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *ICLR 2024 Workshop*

on Mathematical and Empirical Understanding of Foundation Models. Retrieved from https://openreview.net/forum?id=9YvfRrpmyw
[166] Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2567–2577.
[167] Rudolph Emil Kalman. 1960. A new approach to linear filtering and prediction problems. *Trans. ASME, D* 82 (1960), 35–44.
[168] Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. 2024. Gear: An efficient KV cache compression recipefor near-lossless generative inference of LLM. arXiv:2403.05527. Retrieved from https://arxiv.org/abs/2403.05527
[169] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. arXiv:2001.08361. Retrieved from https://arxiv.org/abs/2001.08361
[170] Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, et al. 2023. DSPy: Compiling declarative language model calls into self-improving pipelines. arXiv:2310.03714. Retrieved from https://arxiv.org/abs/2310.03714
[171] Jeonghoon Kim, Jung Hyun Lee, Sungdong Kim, Joonsuk Park, Kang Min Yoo, Se Jung Kwon, and Dongsoo Lee. 2023. Memory-efficient fine-tuning of compressed large language models via Sub-4-bit integer quantization. *Adv. Neural Inf. Process. Syst*. 36 (2023), 36187–36207.
[172] Minsoo Kim, Sihwa Lee, Janghwan Lee, Sukjin Hong, Du-Seong Chang, Wonyong Sung, and Jungwook Choi. 2023. Token-scaled logit distillation for ternary weight generative language models. *Adv. Neural Inf. Process. Syst*. 36 (2023), 42097–42118.
[173] Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. 2023. Squeezellm: Dense-and-sparse quantization. arXiv:2306.07629. Retrieved from https://arxiv.org/abs/2306.07629
[174] Yoon Kim, and Alexander M. Rush. 2016. Sequence-Level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 1317–1327.
[175] Young Jin Kim, Raffy Fahim, and Hany Hassan Awadalla. 2023. Mixture of quantized experts (MoQE): Complementary effect of low-bit quantization and robustness. arXiv:2310.02410. Retrieved from https://arxiv.org/abs/2310.02410
[176] Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. 2024. DistiLLM: Towards streamlined distillation for large language models. arXiv:2402.03898. Retrieved from https://arxiv.org/abs/2402.03898
[177] Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, et al. 2022. The stack: 3 TB of permissively licensed source code. arXiv:2211.15533. Retrieved from https://arxiv.org/abs/2211.15533
[178] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *Proceedings of the Eleventh International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=VD-AYtP0dve
[179] Divyanshu Kumar, Anurakt Kumar, Sahil Agarwal, and Prashanth Harshangi. 2024. Fine-tuning, quantization, and LLMs: Navigating unintended outcomes. arXiv:2404.04392. Retrieved from https://arxiv.org/abs/2404.04392
[180] Ohjoon Kwon, Donghyeon Jeon, Nayoung Choi, Gyu-Hwung Cho, Hwiyeol Jo, Changbong Kim, Hyunwoo Lee, Inho Kang, Sun Kim, and Taiwoo Park. 2024. SLM as guardian: Pioneering AI safety with small language model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, Franck Dernoncourt, Daniel Preoţiuc-Pietro, and Anastasia Shimorina (Eds.). Association for Computational Linguistics, 1333–1350. DOI: https://doi.org/10.18653/v1/2024.emnlp-industry.99
[181] Yanis Labrak, Adrien Bazoge, Emmanuel Morin, Pierre-Antoine Gourraud, Mickael Rouvier, and Richard Dufour. 2024. Biomistral: A collection of open-source pretrained large language models for medical domains. arXiv:2402.10373. Retrieved from https://arxiv.org/abs/2402.10373
[182] Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An Evaluation Dataset for Intent Classification and Out-of-Scope Prediction. In *Emnlp-Ijcnlp 2019*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, 1311–1316. DOI: https://doi.org/10.18653/v1/D19-1131
[183] Hugo Laurençon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, Teven Le Scao, Leandro Von Werra, Chenghao Mou, Eduardo González Ponferrada, Huu Nguyen, et al. 2022. The bigscience roots corpus: A 1.6 tb composite multilingual dataset. *Adv. Neural Inf. Process. Syst*. 35 (2022), 31809–31826.
[184] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2023. Bloom: A 176b-parameter open-access multilingual language model. arxiv:2211.05100. Retrieved from https://arxiv.org/abs/2211.05100

[185] Hojae Lee, Junho Kim, and SangKeun Lee. 2024. Mentor-KD: Making small language models better multi-step reasoners. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, 17643–17658. DOI: https://doi.org/10.18653/v1/2024.emnlp-main.977

[186] Jooyoung Lee, Fan Yang, Thanh Tran, Qian Hu, Emre Barut, and Kai-Wei Chang. 2024. Can small language models help large language models reason better?: LM-Guided chain-of-thought. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING '24)*, 2835–2843.

[187] Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, et al. 2022. xFormers: A modular and hackable transformer modelling library. Retrieved from https://github.com/facebookresearch/xformers

[188] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. arXiv:1607.06450. Retrieved from https://arxiv.org/abs/1607.06450v1

[189] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *Adv. Neural Inf. Process. Syst*. 35, (2022), 3843–3857.

[190] Chenglin Li, Qianglong Chen, Liangyue Li, Caiyu Wang, Yicheng Li, Zulong Chen, and Yin Zhang. 2023. Mixed distillation helps smaller language model better reasoning. arXiv:2312.10730. Retrieved from https://arxiv.org/abs/2312.10730

[191] Guangyan Li, Yongqiang Tang, and Wensheng Zhang. 2024. LoRAP: Transformer sub-layers deserve differentiated structured compression for large language models. arXiv:2404.09695. Retrieved from https://arxiv.org/abs/2404.09695

[192] Haitao Li, Qingyao Ai, Jia Chen, Qian Dong, Zhijing Wu, Yiqun Liu, Chong Chen, and Qi Tian. 2024. BLADE: Enhancing black-box large language models with small domain-specific models. arXiv:2403.18365. Retrieved from https://arxiv.org/abs/2403.18365

[193] Haoran Li, Dadi Guo, Donghao Li, Wei Fan, Qi Hu, Xin Liu, Chunkit Chan, Duanyi Yao, Yuan Yao, and Yangqiu Song. 2024. PrivLM-Bench: A multi-level privacy evaluation benchmark for language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 54–73.

[194] Junyi Li, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023. HaluEval: A Large-Scale hallucination evaluation benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 6449–6464. DOI: https://doi.org/10.18653/v1/2023.emnlp-main.397

[195] Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, et al. 2024. DataComp-LM: In search of the next generation of training sets for language models. arXiv:2406.11794. Retrieved from https://arxiv.org/abs/2406.11794

[196] Luchang Li, Sheng Qian, Jie Lu, Lunxi Yuan, Rui Wang, and Qin Xie. 2024. Transformer-lite: high-efficiency deployment of large language models on mobile phone GPUs. arXiv:2403.20041. Retrieved from https://arxiv.org/abs/2403.20041

[197] Pingzhi Li, Xiaolong Jin, Yu Cheng, and Tianlong Chen. 2024. Examining post-training quantization for mixture-of-experts: A benchmark. arXiv:2406.08155. Retrieved from https://arxiv.org/abs/2406.08155

[198] Quan Li, Tianxiang Zhao, Lingwei Chen, Junjie Xu, and Suhang Wang. 2024. Enhancing graph neural networks with limited labeled data by actively distilling knowledge from large language models. arXiv:2407.13989. Retrieved from https://arxiv.org/abs/2407.13989

[199] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. 2023. StarCoder: May the source be with you! arXiv:2305.06161. Retrieved from https://arxiv.org/abs/2305.06161

[200] Shengrui Li, Xueting Han, and Jing Bai. 2024. Nuteprune: Efficient progressive pruning with numerous teachers for large language models. arXiv:2402.09773. Retrieved from https://arxiv.org/abs/2402.09773

[201] Tianlin Li, Qian Liu, Tianyu Pang, Chao Du, Qing Guo, Yang Liu, and Min Lin. 2024. Purifying large language models by ensembling a small language model. arXiv:2402.14845. Retrieved from https://arxiv.org/abs/2402.14845

[202] Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori B. Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, Vol. 1 Long Papers, Association for Computational Linguistics, 12286–12312.

[203] Xiang Lisa Li, and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. arXiv:2101.00190. Retrieved from https://arxiv.org/abs/2101.00190

[204] Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need II: Phi-1.5 Technical Report. arXiv:2309.05463. Retrieved from https://arxiv.org/abs/2309.05463

[205] Yun Li, Lin Niu, Xipeng Zhang, Kai Liu, Jianchen Zhu, and Zhanhui Kang. 2023. E-Sparse: Boosting the large language model inference through entropy-based N:M sparsity. arXiv:2310.15929. Retrieved from https://arxiv.org/abs/2310.15929

[206] Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. 2023. Large language models in finance: A survey. *In Proceedings of the Fourth ACM International Conference on AI in Finance*, 374–382.

[207] Wing Lian, Guan Wang, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet, and Vong, Teknium. 2023. SlimOrca: An Open Dataset of GPT-4 Augmented FLAN Reasoning Traces, with Verification. Retrieved from https://https://huggingface.co/Open-Orca/SlimOrca

[208] Jinggui Liang, Lizi Liao, Hao Fei, and Jing Jiang. 2024. Synergizing large language models and Pre-Trained smaller models for conversational intent discovery. In *Findings of the Association for Computational Linguistics ACL '24*. Association for Computational Linguistics, 14133–14147.

[209] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2023. Holistic evaluation of language models. In TMLR *'23*. Retrieved from https://openreview.net/forum?id=iO4LZibEqW

[210] Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. 2024. Jamba: A hybrid transformer-Mamba language model. arXiv:2403.19887. Retrieved from https://arxiv.org/abs/2403.19887

[211] Jianghao Lin, Rong Shan, Chenxu Zhu, Kounianhua Du, Bo Chen, Shigang Quan, Ruiming Tang, Yong Yu, and Weinan Zhang. 2024. Rella: Retrieval-enhanced large language models for lifelong sequential behavior comprehension in recommendation. In *Proceedings of the ACM on Web Conference 2024*, 3497–3508.

[212] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. *Proc. Mach. Learn. Syst*. 6 (2024), 87–100.

[213] Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, Vol. 1 Long Papers, Association for Computational Linguistics, 3214–3252.

[214] Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, et al. 2022. Few-shot learning with multilingual generative language models. In *EMNLP-Main 2022*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, 9019–9052. DOI: https://doi.org/10.18653/v1/2022.emnlp-main.616

[215] Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, et al. 2024. Rho-1: Not all tokens are what you need. arXiv:2404.07965. Retrieved from https://arxiv.org/abs/2404.07965

[216] Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. 2024. DeepSeek-v2: A strong, economical, and efficient mixture-of-experts language model. arXiv:2405.04434. Retrieved from https://arxiv.org/abs/2405.04434

[217] Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. 2024. Tuning language models by proxy. arXiv:2401.08565. Retrieved from https://arxiv.org/abs/2401.08565

[218] Jiashuo Liu, Zheyan Shen, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. 2021. Towards out-of-distribution generalization: A survey. arXiv:2108.13624. Retrieved from https://arxiv.org/abs/2108.13624

[219] Qijiong Liu, Nuo Chen, Tetsuya Sakai, and Xiao-Ming Wu. 2024. Once: Boosting content-based recommendation with both open-and closed-source large language models. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. ACM, 452–461.

[220] Suqing Liu, Zezhu Yu, Feiran Huang, Yousef Bulbulia, Andreas Bergen, and Michael Liut. 2024. Can small language models with retrieval-augmented generation replace large language models when learning computer science?. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education*, Vol. 1, ACM, 388–393.

[221] Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. 2023. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. arXiv:2312.15685. Retrieved from https://arxiv.org/abs/2312.15685

[222] Yinpeng Liu, Jiawei Liu, Xiang Shi, Qikai Cheng, and Wei Lu. 2024. Let's learn step by step: Enhancing in-context learning ability with curriculum learning. arXiv:2402.10738. Retrieved from https://arxiv.org/abs/2402.10738

[223] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. arXiv:1907.11692. Retrieved from https://arxiv.org/abs/1907.11692

[224] Yanming Liu, Xinyue Peng, Xuhong Zhang, Weihao Liu, Jianwei Yin, Jiannan Cao, and Tianyu Du. 2024. RA-ISF: Learning to answer and understand from retrieval augmentation via iterative self-feedback. arXiv:2403.06840. Retrieved from https://arxiv.org/abs/2403.06840

[225] Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2023. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. *Adv. Neural Inf. Process. Syst*. 36 (2023), 52342–52364.

[226] Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2023. LLM-QAT: Data-free quantization aware training for large language models. arXiv:2305.17888. Retrieved from https://arxiv.org/abs/2305.17888

[227] Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, et al. 2024. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. arXiv:2402.14905. Retrieved from https://arxiv.org/abs/2402.14905

[228] Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. 2024. On LLMs-Driven synthetic data generation, curation, and evaluation: A survey. In *Findings of the Association for Computational Linguistics ACL '24*. Association for Computational Linguistics, 11065–11082.

[229] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*. PMLR, 22631–22648.

[230] Shayne Longpre, Robert Mahari, Anthony Chen, Naana Obeng-Marnu, Damien Sileo, William Brannon, Niklas Muennighoff, Nathan Khazam, Jad Kabbara, Kartik Perisetla, et al. 2023. The data provenance initiative: A large scale audit of dataset licensing & attribution in AI. arXiv:2310.16787. Retrieved from https://arxiv.org/abs/2310.16787

[231] Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, et al. 2024. StarCoder 2 and the stack v2: The next generation. arXiv:2402.19173. Retrieved from https://arxiv.org/abs/2402.19173

[232] Wenhao Lu, Jian Jiao, and Ruofei Zhang. 2020. Twinbert: Distilling knowledge to twin-structured compressed bert models for large-scale retrieval. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. ACM, 2645–2652.

[233] Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Xiwen Zhang, Nicholas D. Lane, and Mengwei Xu. 2024. Small language models: Survey, measurements, and insights. arXiv:2409.15790. Retrieved from https://arxiv.org/abs/2409.15790

[234] Haitong Luo, Xuying Meng, Suhang Wang, Tianxiang Zhao, Fali Wang, Hanyun Cao, and Yujun Zhang. 2024. Enhance graph alignment for large language models. arXiv:2410.11370. Retrieved from https://arxiv.org/abs/2410.11370

[235] Renqian Luo, Liai Sun, Yingce Xia, Tao Qin, Sheng Zhang, Hoifung Poon, and Tie-Yan Liu. 2022. BioGPT: Generative pre-trained transformer for biomedical text generation and mining. *Briefings in Bioinformatics* 23, 6 (2022). bbac409.

[236] Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. 2024. The era of 1-bit LLMS: All large language models are in 1.58 bits. arXiv:2402.17764. Retrieved from https://arxiv.org/abs/2402.17764

[237] Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *Adv. Neural Inf. Process. Syst*. 36 (2023), 21702–21720.

[238] Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting in Retrieval-Augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 5303–5315.

[239] Yubo Ma, Yixin Cao, YongChing Hong, and Aixin Sun. 2023. Large language model is not a good few-shot information extractor, but a good reranker for hard samples! arXiv:2303.08559. Retrieved from https://arxiv.org/abs/2303.08559

[240] Yuhan Ma, Chenyou Fan, and Haiqi Jiang. 2023. Sci-cot: Leveraging large language models for enhanced knowledge distillation in small models for scientific QA. In *2023 9th International Conference on Computer and Communications (ICCC)*. IEEE, 2394–2398.

[241] Yingwei Ma, Yue Liu, Yue Yu, Yuanliang Zhang, Yu Jiang, Changjian Wang, and Shanshan Li. 2024. At which training stage does code data help LLMs reasoning? In *The Twelfth International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=KIPJKST4gw

[242] Ian Magnusson, Akshita Bhagia, Valentin Hofmann, Luca Soldaini, Ananya Harsh Jha, Oyvind Tafjord, Dustin Schwenk, Evan Pete Walsh, Yanai Elazar, Kyle Lo, et al. 2023. Paloma: A benchmark for evaluating language model fit. arXiv:2312.10523. Retrieved from https://arxiv.org/abs/2312.10523

[243] Dakota Mahan, Ryan Carlow, Louis Castricato, Nathan Cooper, and Christian Laforte. [n. d.] Stable beluga models. Retrieved from https://huggingface.co/stabilityai/StableBeluga2

[244] Vladimir Malinovskii, Denis Mazur, Ivan Ilin, Denis Kuznedelev, Konstantin Burlachenko, Kai Yi, Dan Alistarh, and Peter Richtarik. 2024. PV-tuning: Beyond straight-through estimation for extreme LLM compression. arXiv:2405.14852. Retrieved from https://arxiv.org/abs/2405.14852

[245] Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. SelfCheckGPT: Zero-Resource Black-Box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural*

Language Processing, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 9004–9017. DOI: https://doi.org/10.18653/v1/2023.emnlp-main.557

[246] Sachin Mehta, Mohammad Hossein Sekhavat, Qingqing Cao, Maxwell Horton, Yanzi Jin, Chenfan Sun, Seyed Iman Mirzadeh, Mahyar Najibi, Dmitry Belenko, Peter Zatloukal, et al. 2024. OpenELM: An efficient language model family with open training and inference framework. In *Workshop on Efficient Systems for Foundation Models II (ICML '24)*.

[247] Dheeraj Mekala, Alex Nguyen, and Jingbo Shang. 2024. Smaller language models are capable of selecting instruction-tuning training data for larger language models. arXiv:2402.10430. Retrieved from https://arxiv.org/abs/2402.10430

[248] Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024. ShortGPT: Layers in large language models are more redundant than you expect. arXiv:2403.03853. Retrieved from https://arxiv.org/abs/2403.03853

[249] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. arXiv:1609.07843. Retrieved from https://arxiv.org/abs/1609.07843

[250] Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-Tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 12076–12100. https://doi.org/10.18653/v1/2023.emnlp-main.741

[251] Go Min-Su. 2024. Deep Learning Bible - 8. Large Language Models. WikiDocs. Retrieved from https://wikidocs.net/237419

[252] Eric Mitchell, Rafael Rafailov, Archit Sharma, Chelsea Finn, and Christopher D. Manning. 2024. An emulator for fine-tuning large language models using small language models. In *The Twelfth International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=Eo7kv0sllr

[253] Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agarwal, Xuxi Chen, Anastasia Razdaibiedina, Erik Jones, Kriti Aggarwal, et al. 2023. Orca 2: Teaching small language models how to reason. arXiv:2311.11045. Retrieved from https://arxiv.org/abs/2311.11045

[254] Lingbo Mo, Boshi Wang, Muhao Chen, and Huan Sun. 2024. How trustworthy are Open-Source LLMs? An assessment under malicious demonstrations shows their vulnerabilities. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Vol. 1 Long Papers, Association for Computational Linguistics, 2775–2792.

[255] John Xavier Morris, Wenting Zhao, Justin T. Chiu, Vitaly Shmatikov, and Alexander M. Rush. 2024. Language model inversion. In *The Twelfth International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=t9dWHpGkPj

[256] Maximilian Mozes, Xuanli He, Bennett Kleinberg, and Lewis D. Griffin. 2023. Use of LLMS for illicit purposes: Threats, prevention measures, and vulnerabilities. arXiv:2308.12833. Retrieved from https://arxiv.org/abs/2308.12833

[257] Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Evan Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. 2025. OLMoE: Open mixture-of-experts language models. In *The Thirteenth International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=xXTkbTBmqq

[258] Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of GPT-4. arXiv:2306.02707. Retrieved from https://arxiv.org/abs/2306.02707

[259] Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. 2024. Compact language models via pruning and knowledge distillation. arXiv:2407.14679. Retrieved from https://arxiv.org/abs/2407.14679

[260] Rithesh Murthy, Liangwei Yang, Juntao Tan, Tulika Manoj Awalgaonkar, Yilun Zhou, Shelby Heinecke, Sachin Desai, Jason Wu, Ran Xu, Sarah Tan, Jianguo Zhang, Zhiwei Liu, Shirley Kokane, Zuxin Liu, Ming Zhu, Huan Wang, Caiming Xiong, and Silvio Savarese. 2024. MobileAIBench: Benchmarking LLMs and LMMs for on-device use cases. arXiv:240610290. Retrieved from https://arxiv.org/abs/2406.10290

[261] Kalyan Nakka, Jimmy Dani, and Nitesh Saxena. 2024. Is on-device AI broken and exploitable? Assessing the trust and ethics in small language models. arXiv:2406.05364. Retrieved from https://arxiv.org/abs/2406.05364

[262] Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. 2024. Using an LLM to help with code understanding. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. ACM, 1–13.

[263] Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo Ponti. 2024. Dynamic memory compression: Retrofitting LLMs for accelerated inference. In *Forty-First International Conference on Machine Learning*. Retrieved from https://openreview.net/forum?id=tDRYrAkOB7

[264] Thuat Nguyen, Chien Van Nguyen, Viet Dac Lai, Hieu Man, Nghia Trung Ngo, Franck Dernoncourt, Ryan A. Rossi, and Thien Huu Nguyen. 2024. CulturaX: A cleaned, enormous, and multilingual dataset for large language models in 167 languages. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING '24)*. Association for Computational Linguistics, 4226–4237.

[265] Tuan Dung Nguyen, Yuan-Sen Ting, Ioana Ciuca, Charles O'Neill, Ze-Chang Sun, Maja Jabłońska, Sandor Kruk, Ernest Perkowski, Jack Miller, Jason Jason Jingsh Li, et al. 2023. AstroLLaMA: Towards specialized foundation models in astronomy. In *Proceedings of the Second Workshop on Information Extraction from Scientific Publications*. Association for Computational Linguistics, 49–55.

[266] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, and Yinfei Yang. 2022. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 9844–9855. DOI: https://doi.org/10.18653/v1/2022.emnlp-main.669

[267] Rodrigo Nogueira, and Kyunghyun Cho. 2019. Passage re-ranking with BERT. arXiv:1901.04085. Retrieved from https://arxiv.org/abs/1901.04085

[268] A. Noorian. 2024. A BERT-based sequential POI recommender system in social media. *Comput. Stand. Interf*. 87 (2024), 103766. DOI: https://doi.org/10.1016/j.csi.2023.103766

[269] OpenAI. 2024. *GPT-4o mini: Advancing cost-efficient intelligence*. Retrieved from https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/. Accessed: July 18, 2024.

[270] OpenAI. 2024. *Hello GPT-4o*. Retrieved from https://openai.com/index/hello-gpt-4o/. Accessed: May 13, 2024.

[271] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Adv. Neural Inf. Process. Syst*. 35, (2022), 27730–27744.

[272] Shankar Padmanabhan, Yasumasa Onoe, Michael Zhang, Greg Durrett, and Eunsol Choi. 2023. Propagating knowledge updates to lms through distillation. *Adv. Neural Inf. Process. Syst*. 36 (2023), 47124–47142.

[273] Jupinder Parmar, Shrimai Prabhumoye, Joseph Jennings, Mostofa Patwary, Sandeep Subramanian, Dan Su, Chen Zhu, Deepak Narayanan, Aastha Jhunjhunwala, Ayush Dattagupta, et al. 2024. Nemotron-4 15B Technical Report. arXiv:2402.16819. Retrieved from https://arxiv.org/abs/2402.16819

[274] Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. 2024. OpenWebMath: An open dataset of High-Quality mathematical web text. In *The Twelfth International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=jKHmjlpViu

[275] Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. arXiv:2406.17557. Retrieved from https://arxiv.org/abs/2406.17557

[276] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The RefinedWeb dataset for Falcon LLM: Outperforming curated corpora with web data, and web data only. arXiv:2306.01116. Retrieved from https://arxiv.org/abs/2306.01116

[277] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, et al. 2023. RWKV: Reinventing RNNs for the Transformer Era. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 14048–14077. DOI: https://doi.org/10.18653/v1/2023.findings-emnlp.936

[278] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with GPT-4. arXiv:2304.03277. Retrieved from https://arxiv.org/abs/2304.03277

[279] Zhiyuan Peng, Xuyang Wu, Qifan Wang, and Yi Fang. 2023. Soft prompt tuning for augmenting dense retrieval with large language models. arXiv:2307.08303. Retrieved from https://arxiv.org/abs/2307.08303

[280] Ethan Perez, Sam Ringer, Kamile Lukosiute, Karina Nguyen, Edwin Chenandet al. 2023. Discovering language model behaviors with Model-Written evaluations. In *Findings of ACL '23*. Association for Computational Linguistics, 13387–13434.

[281] Pascal Pfeiffer, Philipp Singer, and Yauhen Babakhin, Gabor Fodor, Nischay Dhankhar, and Sri Satish Ambati. 2024. H_2O-Danube3 Technical Report. arXiv:2407.09276. Retrieved from https://arxiv.org/abs/2407.09276

[282] Karmvir Singh Phogat, Sai Akhil Puranam, Sridhar Dasaratha, Chetan Harsha, and Shashishekar Ramakrishna. 2024. Fine-tuning Smaller Language Models for Question Answering over Financial Documents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, 10528–10548. https://doi.org/10.18653/v1/2024.findings-emnlp.617

[283] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. *Proc. Mach. Learn. Syst*. 5, 606–624.

[284] Ofir Press, Noah Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=R8sQPpGCv0
[285] Ruiyang Qin, Jun Xia, Zhenge Jia, Meng Jiang, Ahmed Abbasi, Peipei Zhou, Jingtong Hu, and Yiyu Shi. 2023. Enabling on-device large language model personalization with self-supervised data selection and synthesis. arXiv:2311.12275. Retrieved from https://arxiv.org/abs/2311.12275
[286] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*.
[287] Haohao Qu, Liangbo Ning, Rui An, Wenqi Fan, Tyler Derr, Hui Liu, Xin Xu, and Qing Li. 2024. A survey of Mamba. arXiv:2408.01129. Retrieved from https://arxiv.org/abs/2408.01129
[288] Haohao Qu, Yifeng Zhang, Liangbo Ning, Wenqi Fan, and Qing Li. 2024. SSD4Rec: A structured state space duality model for efficient sequential recommendation. arXiv:2409.01192. Retrieved from https://arxiv.org/abs/2409.01192
[289] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI Blog* 1, 8 (2019), 9.
[290] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Adv. Neural Inf. Process. Syst*. 36, (2023), 53728–53741.
[291] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res*. 21, 140 (2020), 1–67. DOI: https://doi.org/10.5555/3455716.3455856
[292] Mohammad Wali Ur Rahman, Murad Mehrab Abrar, Hunter Gibbons Copening, Salim Hariri, Sicong Shao, Pratik Satam, and Soheil Salehi. 2023. Quantized transformer language model implementations on edge devices. arXiv:2310.03971. Retrieved from https://arxiv.org/abs/2310.03971
[293] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–16.
[294] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-Context Retrieval-Augmented language models. *Trans. Assoc. Comput. Linguist*. 11, (2023), 1316–1331. DOI: https://doi.org/10.1162/tacl_a_00605
[295] Krithika Ramesh, Arnav Chavan, Shrey Pandit, and Sunayana Sitaram. 2023. A comparative study on the impact of model compression techniques on fairness in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, Vol. 1 Long Papers, Association for Computational Linguistics, 15762–15782. Retrieved from https://aclanthology.org/2023.acl-long.878
[296] Al Mamunur Rashid, George Karypis, and John Riedl. 2008. Learning preferences of new users in recommender systems: An information theoretic approach. *ACM SIGKDD Explor. Newsl*. 10, 2 (2008), 90–100. DOI: https://doi.org/10.1145/1540276.1540302
[297] Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2024. Androidinthewild: A large-scale dataset for android device control. *Adv. Neural Inf. Process. Syst*. 36,. 2024.
[298] Stephen Robertson, and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Ret*. 3, 4 (2009), 333–389. DOI: https://doi.org/10.1561/1500000019
[299] Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code Llama: Open foundation models for code. arXiv:2308.12950. Retrieved from https://arxiv.org/abs/2308.12950
[300] Caitlin Sadowski, and Greg Levin. 2007. *Simhash: Hash-Based Similarity Detection.* Technical report, Google.
[301] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Commun. ACM* 64, 9 (2021), 99–106. DOI: https://doi.org/10.1145/3474381
[302] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2024. Toolformer: Language models can teach themselves to use tools. *Adv. Neural Inf. Process. Syst*. 36, (2024). DOI: https://doi.org/10.18653/v1/2025.realm-1.14
[303] Rico Sennrich, Jannis Vamvas, and Alireza Mohammadshahi. 2023. Mitigating hallucinations and off-target machine translation with source-contrastive and language-contrastive decoding. arXiv:2309.07098. Retrieved from https://arxiv.org/abs/2309.07098
[304] Zeyang Sha, and Yang Zhang. 2024. Prompt stealing attacks against large language models. arXiv:2402.12959. Retrieved from https://arxiv.org/abs/2402.12959
[305] Yu Shang, Yu Li, Fengli Xu, and Yong Li. 2024. Synergy-of-thoughts: Eliciting efficient reasoning in hybrid language models. arXiv:2402.02563. Retrieved from https://arxiv.org/abs/2402.02563

[306] Yuzhang Shang, Zhihang Yuan, Qiang Wu, and Zhen Dong. 2023. PB-LLM: Partially binarized large language models. arXiv:231000034. Retrieved from https://arxiv.org/abs/2310.00034

[307] Hang Shao, Bei Liu, and Yanmin Qian. 2024. One-shot sensitivity-aware mixed sparsity pruning for large language models. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 11296–11300.

[308] Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R. Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R. Johnston, et al. 2023. Towards understanding sycophancy in language models. arXiv:2310.13548. Retrieved from https://arxiv.org/abs/2310.13548

[309] Erfan Shayegani, Md Abdullah Al Mamun, Yu Fu, Pedram Zaree, Yue Dong, and Nael Abu-Ghazaleh. 2023. Survey of vulnerabilities in large language models revealed by adversarial attacks. arXiv:2310.10844. Retrieved from https://arxiv.org/abs/2310.10844

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

[311] Noam Shazeer. 2020. Glu variants improve transformer. arXiv:2002.05202. Retrieved from https://arxiv.org/abs/2002.05202

[312] Bowen Shen, Zheng Lin, Yuanxin Liu, Zhengxiao Liu, Lei Wang, and Weiping Wang. 2022. COST-EFF: Collaborative optimization of spatial and temporal efficiency with slenderized multi-exit language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, 1719–1730. https://doi.org/10.18653/v1/2022.emnlp-main.112

[313] Bowen Shen, Zheng Lin, Daren Zha, Wei Liu, Jian Luan, Bin Wang, and Weiping Wang. 2024. Pruning Large Language Models to Intra-module Low-rank Architecture with Transitional Activations. In *Findings of the Association for Computational Linguistics: ACL 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, 9781–9793. https://doi.org/10.18653/v1/2024.findings-acl.582

[314] Weizhou Shen, Chenliang Li, Hongzhan Chen, Ming Yan, Xiaojun Quan, Hehong Chen, Ji Zhang, and Fei Huang. 2024. Small LLMs are weak tool learners: A Multi-LLM agent. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 16658–16680. https://doi.org/10.18653/v1/2024.emnlp-main.929

[315] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*. PMLR, 31094–31116.

[316] Wentao Shi, Xiangnan He, Yang Zhang, Chongming Gao, Xinyue Li, Jizhi Zhang, Qifan Wang, and Fuli Feng. 2024. Large language models are learnable planners for long-term recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1893–1903.

[317] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-LM: Training multi-billion parameter language models using model parallelism. arXiv:1909.08053. Retrieved from https://arxiv.org/abs/1909.08053

[318] Parshin Shojaee, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, and Chandan K. Reddy. 2024. LLM-SR: Scientific equation discovery via programming with large language models. arXiv:2404.18400. Retrieved from https://arxiv.org/abs/2404.18400

[319] Manli Shu, Weili Nie, De-An Huang, Zhiding Yu, Tom Goldstein, Anima Anandkumar, and Chaowei Xiao. 2022. Test-time prompt tuning for zero-shot generalization in vision-language models. *Adv. Neural Inf. Process. Syst*. 35, (2022), 14274–14289. DOI: https://doi.org/10.5555/3600270.3601308

[320] Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2023. Large language models encode clinical knowledge. *Nature* 620, 7972 (2023), 172–180. DOI: https://doi.org/10.1038/s41586-023-06291-2

[321] Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R. Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. Retrieved from https://cerebras.ai/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama; https://huggingface.co/datasets/cerebras/SlimPajama-627B

[322] Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, et al. 2024. Dolma: An Open Corpus of Three Trillion Tokens for Language Model Pretraining Research. arXiv:2402.00159. Retrieved from https://arxiv.org/abs/2402.00159

[323] Sofia Eleni Spatharioti, David M. Rothschild, Daniel G. Goldstein, and Jake M. Hofman. 2023. Comparing traditional and LLM-based search for consumer choice: A randomized experiment. arXiv:2307.03744. Retrieved from https://arxiv.org/abs/2307.03744

[324] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing* 568, (2024), 127063. DOI: https://doi.org/10.1016/j.neucom.2023.127063

[325] Liangtai Sun, Yang Han, Zihan Zhao, Da Ma, Zhennan Shen, Baocai Chen, Lu Chen, and Kai Yu. 2024. Scieval: A multi-level large language model evaluation benchmark for scientific research. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38, 19053–19061.

[326] Lichao Sun, Yue Huang, Haoran Wang, Siyuan Wu, Qihui Zhang, Chujie Gao, Yixin Huang, Wenhan Lyu, Yixuan Zhang, Xiner Li, et al. 2024. TrustLLM: Trustworthiness in large language models. arXiv:2401.05561. Retrieved from https://arxiv.org/abs/2401.05561

[327] Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2024. A simple and effective pruning approach for large language models. In *Proceedings of the Twelfth International Conference on Learning Representations*. ICLR.

[328] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: A compact task-agnostic BERT for resource-limited devices. arXiv:2004.02984. Retrieved from https://arxiv.org/abs/2004.02984

[329] Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 9275–9293.

[330] Alon Talmor, and Jonathan Berant. 2018. The web as a Knowledge-Base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Marilyn Walker, Heng Ji, and Amanda Stent (Eds.). Association for Computational Linguistics, 641–651. DOI: https://doi.org/10.18653/v1/N18-1059

[331] Jiejun Tan, Zhicheng Dou, Yutao Zhu, Peidong Guo, Kun Fang, and Ji-Rong Wen. 2024. Small models, big insights: leveraging slim proxy models to decide when and what to retrieve for LLMs. arXiv:2402.12052. Retrieved from https://arxiv.org/abs/2402.12052

[332] Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. 2023. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. arXiv:2306.05301. Retrieved from https://arxiv.org/abs/2306.05301

[333] Xuemei Tang, Jun Wang, and Qi Su. 2024. Small language model is a good guide for large language model in Chinese entity relation extraction. arXiv:2402.14373. Retrieved from https://arxiv.org/abs/2402.14373

[334] Yehui Tang, Fangcheng Liu, Yunsheng Ni, Yuchuan Tian, Zheyuan Bai, Yi-Qi Hu, Sichao Liu, Shangling Jui, Kai Han, and Yunhe Wang. 2024. Rethinking optimization and architecture for tiny language models. arXiv:2402.02791. Retrieved from https://arxiv.org/abs/2402.02791

[335] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford Alpaca: An instruction-following LLaMA model. Retrieved from https://github.com/tatsu-lab/stanford_alpaca

[336] Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science. arXiv:2211.09085. Retrieved from https://arxiv.org/abs/2211.09085

[337] CodeGemma Team. 2024. CodeGemma: Open code models based on Gemma. arXiv:2406.11409. Retrieved from https://arxiv.org/abs/2406.11409

[338] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. arXiv:2403.08295. Retrieved from https://arxiv.org/abs/2403.08295

[339] Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. arXiv:2408.00118. Retrieved from https://arxiv.org/abs/2408.00118

[340] TensorOpera Team. 2024. *TensorOpera Unveils Fox Foundation Model: A Pioneering Small Language Model (SLM) for Cloud and Edge*. Retrieved from https://blog.tensoropera.ai/tensoropera-unveils-fox-foundation-model-a-pioneering-open-source-slm-leading-the-way-against-tech-giants/. Accessed: June 13, 2024.

[341] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2464–2469.

[342] Omkar Thawakar, Ashmal Vayani, Salman Khan, Hisham Cholakal, Rao M. Anwer, Michael Felsberg, Tim Baldwin, Eric P. Xing, and Fahad Shahbaz Khan. 2024. Mobillama: Towards accurate and lightweight fully transparent GPT. arXiv:2402.16840. Retrieved from https://arxiv.org/abs/2402.16840

[343] Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian Yu, Haitao Mi, and Dong Yu. 2024. Toward self-improvement of LLMs via imagination, searching, and criticizing. arXiv:2404.12253. Retrieved from https://arxiv.org/abs/arXiv:2404.12253.

[344] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. arXiv:2302.13971. Retrieved from https://arxiv.org/abs/2302.13971

[345] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. arXiv:2307.09288. Retrieved from https://arxiv.org/abs/2307.09288

[346] Jonathan Tow, Marco Bellagente, Dakota Mahan, and Carlos Riquelme. 2024. StableLM 3B 4E1T. Retrieved from https://huggingface.co/stabilityai/stablelm-3b-4e1t

[347] Trieu H. Trinh, and Quoc V. Le. 2018. A simple method for commonsense reasoning. arXiv:1806.02847. Retrieved from https://arxiv.org/abs/1806.02847

[348] Adina Trufinescu. 2024. Discover the New Multi-Lingual High-Quality Phi-3.5 SLMs. Retrieved from https://techcommunity.microsoft.com/t5/ai-azure-ai-services-blog/discover-the-new-multi-lingual-high-quality-phi-3-5-slms/ba-p/4225280

[349] Dennis Ulmer, Martin Gubri, Hwaran Lee, Sangdoo Yun, and Seong Joon Oh. 2024. Calibrating large language models using their generations only. arXiv:2403.05973. Retrieved from https://arxiv.org/abs/2403.05973

[350] Sander Van Der Linden. 2022. Misinformation: Susceptibility, spread, and interventions to immunize the public. *Nat. Med*. 28, 3 (2022), 460–467. DOI: https://doi.org/10.1038/s41591-022-01713-6

[351] Chien Van Nguyen, Xuan Shen, Ryan Aponte, Yu Xia, Samyadeep Basu, Zhengmian Hu, Jian Chen, Mihir Parmar, Sasidhar Kunapuli, Joe Barrow, et al. 2024. A survey of small language models. arXiv:2410.20011. Retrieved from https://arxiv.org/abs/2410.20011

[352] A. Vaswani. 2017. Attention is all you need. *Adv. Neural Inf. Process. Syst*. 30 (2017), 5998–6008. DOI: https://doi.org/10.5555/3295222.3295349

[353] Olga Veksler. 2023. Test time adaptation with regularized loss for weakly supervised salient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7360–7369.

[354] Paul Voigt, and Axel Von Dem Bussche. 2017. The EU general data protection regulation (GDPR). *A Practical Guide* (1st ed.). Springer International Publishing, Cham.

[355] Yuxian Wan, Wenlin Zhang, and Zhen Li. 2023. Multi-Task feature Self-Distillation for Semi-Supervised machine translation. In *International Conference on Neural Information Processing*. Springer, 238–254.

[356] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 353–355.

[357] Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, et al. 2023. DecodingTrust: A comprehensive assessment of trustworthiness in GPT models. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

[358] Boxin Wang, Chejian Xu, Shuohang Wang, Zhe Gan, Yu Cheng, Jianfeng Gao, Ahmed Hassan Awadallah, and Bo Li. 2021. Adversarial glue: A multi-task benchmark for robustness evaluation of language models. arXiv:2111.02840. Retrieved from https://arxiv.org/abs/2111.02840

[359] Fali Wang, Minhua Lin, Yao Ma, Hui Liu, Qi He, Xianfeng Tang, Jiliang Tang, Jian Pei, and Suhang Wang. 2025. A survey on small language models in the era of large language models: Architecture, capabilities, and trustworthiness. In. *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25)*. ACM, 6173–6183. DOI: https://doi.org/10.1145/3711896.3736563

[360] Fali Wang, Hui Liu, Zhenwei Dai, Jingying Zeng, Zhiwei Zhang, Zongyu Wu, Chen Luo, Zhen Li, Xianfeng Tang, Qi He, et al. 2025. AgentTTS: Large language model agent for test-time compute-optimal scaling strategy in complex tasks. arXiv:2508.00890. Retrieved from https://arxiv.org/abs/2508.00890

[361] Guan Wang, Sijie Cheng, Qiying Yu, and Changling Liu. 2023. OpenLLMs: Less is More for Open-Source Models. DOI: https://doi.org/10.5281/zenodo.8105775

[362] Guan Wang, Sijie Cheng, Xianyuan Zhan, Xiangang Li, Sen Song, and Yang Liu. 2024. OpenChat: Advancing open-source language models with Mixed-Quality data. In *The Twelfth International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=AOJyfhWYHf

[363] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. 2023. Bitnet: Scaling 1-bit transformers for large language models. arXiv:2310.11453. Retrieved from https://arxiv.org/abs/2310.11453

[364] Jindong Wang, H. U. Xixu, Wenxin Hou, Hao Chen, Runkai Zheng, Yidong Wang, Linyi Yang, Wei Ye, Haojun Huang, Xiubo Geng, et al. n.d. On the robustness of ChatGPT: An adversarial and out-of-distribution perspective. In *ICLR 2023 Workshop on Trustworthy and Reliable Large-Scale Machine Learning Models*.

[365] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, Vol. 1 Long Papers. Association for Computational Linguistics, 11897–11916. https://doi.org/10.18653/v1/2024.acl-long.642

[366] Peifeng Wang, Zhengyang Wang, Zheng Li, Yifan Gao, Bing Yin, and Xiang Ren. 2023. SCOTT: Self-Consistent chain-of-thought distillation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, Vol. 1 Long Papers, Association for Computational Linguistics, 5546–5558.

[367] Wenxiao Wang, Wei Chen, Yicong Luo, Yongliu Long, Zhengkai Lin, Liye Zhang, Binbin Lin, Deng Cai, and Xiaofei He. 2024. Model compression and efficient inference for large language models: A survey. arXiv:2402.09748. Retrieved from https://arxiv.org/abs/2402.09748

[368] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. arXiv:2002.10957. Retrieved from https://arxiv.org/abs/2002.10957

[369] Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R. Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. 2024. SciBench: Evaluating College-Level scientific Problem-Solving abilities of large language models. In *Forty-First International Conference on Machine Learning*. Retrieved from https://openreview.net/forum?id=bq1JEgioLr

[370] Yuxia Wang, Haonan Li, Xudong Han, Preslav Nakov, and Timothy Baldwin. 2024. Do-Not-Answer: Evaluating Safeguards in LLMs. In *Findings of the Association for Computational Linguistics: EACL '24*. Association for Computational Linguistics, 896–911. Retrieved from https://aclanthology.org/2024.findings-eacl.61

[371] Yile Wang, Peng Li, Maosong Sun, and Yang Liu. 2023. Self-Knowledge guided retrieval augmentation for large language models. In *Findings of the Association for Computational Linguistics: EMNLP '23*. Association for Computational Linguistics, 10303–10315.

[372] Yubo Wang, Xueguang Ma, and Wenhu Chen. 2023. Augmenting black-box LLMS with medical textbooks for clinical question answering. arXiv:2309.02233. Retrieved from https://arxiv.org/abs/2309.02233

[373] Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaeiandet al. 2022. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *EMNLP '22*. Association for Computational Linguistics, 5085–5109. DOI: https://doi.org/10.18653/v1/2022.emnlp-main.340

[374] Yuyan Wang, Mohit Sharma, Can Xu, Sriraj Badam, Qian Sun, Lee Richardson, Lisa Chung, Ed. H. Chi, and Minmin Chen. 2022. Surrogate for Long-Term user experience in recommender systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 4100–4109. DOI: https://doi.org/10.1145/3534678.3539073

[375] Yuling Wang, Changxin Tian, Binbin Hu, Yanhua Yu, Ziqi Liu, Zhiqiang Zhang, Jun Zhou, Liang Pang, and Xiao Wang. 2024. Can small language models be good reasoners for sequential recommendation? In *Proceedings of the ACM on Web Conference 2024*. ACM, 3876–3887.

[376] Yuqing Wang, and Yun Zhao. 2024. RUPBench: Benchmarking reasoning under perturbations for robustness evaluation in large language models. arXiv:2406.11020. Retrieved from https://arxiv.org/abs/2406.11020

[377] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are Zero-Shot learners. In *International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=gEZrGCozdqR

[378] Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2023. Magicoder: Source code is all you need. arXiv:2312.02120. Retrieved from https://arxiv.org/abs/2312.02120

[379] Johannes Welbl, Amelia Glaese, Jonathan Uesato, Sumanth Dathathri, John Mellor, Lisa Anne Hendricks, Kirsty Anderson, Pushmeet Kohli, Ben Coppin, and Po-Sen Huang. 2021. Challenges in detoxifying language models. In *Findings of the Association for Computational Linguistics: EMNLP '21*. Association for Computational Linguistics, 2447–2469. DOI: https://doi.org/10.18653/v1/2021.findings-emnlp.210

[380] Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-Generated Text*, 94–106.

[381] Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. 2024. AutoDroid: LLM-powered task automation in android. arXiv:2308.15272. Retrieved from https://arxiv.org/abs/2308.15272

[382] Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. 2024. QuRating: Selecting High-Quality data for training language models. In *Forty-First International Conference on Machine Learning*. Retrieved from https://openreview.net/forum?id=GLGYYqPwjy

[383] Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Vol. 1 Long Papers, Association for Computational Linguistics, 1112–1122. DOI: https://doi.org/10.18653/v1/N18-1101

[384] Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2021. Empowering news recommendation with pre-trained language models. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1652–1656.

[385] Wu, Abdul Waheed, Chiyu Zhang, Muhammad Abdul-Mageed, and Alham Fikri Aji. 2024. Minghao LaMini-LM: A diverse herd of distilled models from Large-Scale instructions. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, Yvette Graham and Matthew Purver (Eds.). Association for Computational Linguistics, 944–964. Retrieved from https://aclanthology.org/2024.eacl-long.57

[386] Taiqiang Wu, Cheng Hou, Shanshan Lao, Jiayi Li, Ngai Wong, Zhe Zhao, and Yujiu Yang. 2024. Weight-Inherited Distillation for Task-Agnostic BERT Compression. In *Findings of the Association for Computational Linguistics: NAACL 2024*, Kevin Duh, Helena Gomez, and Steven Bethard (Eds.). Association for Computational Linguistics, 13–28. https://doi.org/10.18653/v1/2024.findings-naacl.2

[387] Xuansheng Wu, Huachi Zhou, Yucheng Shi, Wenlin Yao, Xiao Huang, and Ninghao Liu. 2024. Could small language models serve as recommenders? Towards data-centric cold-start recommendation. In *Proceedings of the ACM on Web Conference 2024*. ACM, 3566–3575.

[388] Zhuofeng Wu, He Bai, Aonan Zhang, Jiatao Gu, V. G. Vydiswaran, Navdeep Jaitly, and Yizhe Zhang. 2024. Divide-or-conquer? Which part should you distill your LLM? arXiv:2402.15000. Retrieved from https://arxiv.org/abs/2402.15000

[389] Nuwa Xi, Yuhan Chen, Sendong Zhao, Haochun Wang, Bing Qin, and Ting Liu. 2024. AS-ES learning: Towards efficient CoT learning in small models. arXiv:2403.01969. Retrieved from https://arxiv.org/abs/2403.01969

[390] Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2024. Sheared LLaMA: Accelerating language model pre-training via structured pruning. In *The Twelfth International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=09iOdaeOzp

[391] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*. PMLR, 38087–38099.

[392] Tinghao Xie, Xiangyu Qi, Yi Zeng, Yangsibo Huang, Udari Madhushani Sehwag, Kaixuan Huang, Luxi He, Boyi Wei, Dacheng Li, Ying Sheng, et al. 2024. Sorry-bench: Systematically evaluating large language model safety refusal behaviors. arXiv:2406.14598. Retrieved from https://arxiv.org/abs/2406.14598

[393] Tong Xie, Yuwei Wan, Wei Huang, Zhenyu Yin, Yixuan Liu, Shaozhou Wang, Qingyuan Linghu, Chunyu Kit, Clara Grazian, Wenjie Zhang, et al. 2023. Darwin series: Domain specific large language models for natural science. arXiv:2308.13565. Retrieved from https://arxiv.org/abs/2308.13565

[394] Weikai Xie, Li Zhang, Shihe Wang, Rongjie Yi, and Mengwei Xu. 2024. DroidCall: A dataset for LLM-powered android intent invocation. arXiv:2412.00402. Retrieved from https://arxiv.org/abs/2412.00402

[395] Xuan Xie, Jiayang Song, Zhehua Zhou, Yuheng Huang, Da Song, and Lei Ma. 2024. Online safety analysis for LLMs: A benchmark, an assessment, and a path forward. arXiv:2404.08517. Retrieved from https://arxiv.org/abs/2404.08517

[396] Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021. BERxiT: Early exiting for BERT with better Fine-Tuning and extension to regression. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty (Eds.). Association for Computational Linguistics, Online, 91–104. https://doi.org/10.18653/v1/2021.eacl-main.8

[397] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. arXiv:2304.12244. Retrieved from https://arxiv.org/abs/2304.12244

[398] Canwen Xu, Yichong Xu, Shuohang Wang, Yang Liu, Chenguang Zhu, and Julian McAuley. 2023. Small models are valuable plug-ins for large language models. arXiv:2305.08848. Retrieved from https://arxiv.org/abs/2305.08848

[399] Daliang Xu, Wangsong Yin, Xin Jin, Ying Zhang, Shiyun Wei, Mengwei Xu, and Xuanzhe Liu. 2023. LLMCad: Fast and scalable on-device large language model inference. arXiv:2309.04255. Retrieved from https://arxiv.org/abs/2309.04255

[400] Daliang Xu, Hao Zhang, Liming Yang, Ruiqi Liu, Gang Huang, Mengwei Xu, and Xuanzhe Liu. 2024. Empowering 1000 tokens/second on-device LLM prefilling with MLLM-NPU. arXiv:2407.05858. Retrieved from https://arxiv.org/abs/2407.05858

[401] Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao, Fangyuan Wang, and Hongwei Hao. 2015. Short text clustering via convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, Phil Blunsom, Shay Cohen, Paramveer Dhillon, and Percy Liang (Eds.). Association for Computational Linguistics, 62–69. DOI: https://doi.org/10.3115/v1/W15-1509

[402] Minrui Xu, Niyato Dusit, Jiawen Kang, Zehui Xiong, Shiwen Mao, Zhu Han, Dong In Kim, and Khaled B. Letaief. 2024. When large language model agents meet 6G networks: Perception, grounding, and alignment. arXiv:2401.07764. Retrieved from https://arxiv.org/abs/2401.07764

[403] Mengwei Xu, Wangsong Yin, Dongqi Cai, Rongjie Yi, Daliang Xu, Qipeng Wang, Bingyang Wu, Yihao Zhao, Chen Yang, Shihe Wang, et al. 2024. A survey of resource-efficient llm and multimodal foundation models. arXiv:2401.08092. Retrieved from https://arxiv.org/abs/2401.08092

[404] Weijia Xu, Sweta Agrawal, Eleftheria Briakou, Marianna Martindale, and Marine Carpuat. 2023. Understanding and detecting hallucinations in neural machine translation via model introspection. *Trans. Assoc. Comput. Linguist*. 11, (2023), 546–564. DOI: https://doi.org/10.1162/tacl_a_00563

[405] Yuzhuang Xu, Xu Han, Zonghan Yang, Shuo Wang, Qingfu Zhu, Zhiyuan Liu, Weidong Liu, and Wanxiang Che. 2024. OneBit: Towards extremely low-bit large language models. arXiv:2402.11295. Retrieved from https://arxiv.org/abs/2402.11295

[406] Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. arXiv:2401.15884. Retrieved from https://arxiv.org/abs/2401.15884

[407] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 Technical Report. arXiv:2407.10671. Retrieved from https://arxiv.org/abs/2407.10671

[408] Chuanpeng Yang, Wang Lu, Yao Zhu, Yidong Wang, Qian Chen, Chenlong Gao, Bingjie Yan, and Yiqiang Chen. 2024. Survey on knowledge distillation for large language models: Methods, evaluation, and application. arXiv:2407.01885. Retrieved from https://arxiv.org/abs/2407.01885

[409] Cheng Yang, Yang Sui, Jinqi Xiao, Lingyi Huang, Yu Gong, Yuanlin Duan, Wenqi Jia, Miao Yin, Yu Cheng, and Bo Yuan. 2024. MoE-I^2: Compressing Mixture of Experts Models through Inter-Expert Pruning and Intra-Expert Low-Rank Decomposition. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, 10456–10466. DOI: https://doi.org/10.18653/v1/2024.findings-emnlp.612

[410] Kevin Yang, Dan Klein, Asli Celikyilmaz, Nanyun Peng, and Yuandong Tian. 2024. RLCD: Reinforcement learning from contrastive distillation for LM alignment. In *The Twelfth International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=v3XXtxWKi6

[411] Kailai Yang, Tianlin Zhang, Ziyan Kuang, Qianqian Xie, Jimin Huang, and Sophia Ananiadou. 2024. MentaLLaMA: Interpretable mental health analysis on social media with large language models. In *Proceedings of the ACM on Web Conference 2024*, 4489–4500.

[412] Linyi Yang, Shuibai Zhang, Libo Qin, Yafu Li, Yidong Wang, Hanmeng Liu, Jindong Wang, Xing Xie, and Yue Zhang. 2023. GLUE-X: Evaluating Natural Language Understanding Models from an Out-of-Distribution Generalization Perspective. In *Findings of the Association for Computational Linguistics: ACL '23*. Association for Computational Linguistics, 12731–12750. DOI: https://doi.org/10.18653/v1/2023.findings-acl.806

[413] Linyi Yang, Shuibai Zhang, Zhuohao Yu, Guangsheng Bao, Yidong Wang, Jindong Wang, Ruochen Xu, Wei Ye, Xing Xie, Weizhu Chen, and Yue Zhang. 2024. Supervised knowledge makes large language models better In-context learners. In *The Twelfth International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=bAMPOUF227

[414] Runming Yang, Taiqiang Wu, Jiahao Wang, Pengfei Hu, Ngai Wong, and Yujiu Yang. 2024. LLM-Neo: Parameter efficient knowledge distillation for large language models. arXiv:2411.06839. Retrieved from https://arxiv.org/abs/2411.06839

[415] Yifei Yang, Zouying Cao, and Hai Zhao. 2024. Laco: Large language model pruning via layer collapse. arXiv:2402.11187. Retrieved from https://arxiv.org/abs/2402.11187

[416] Yu Yang, Siddhartha Mishra, Jeffrey N. Chiang, and Baharan Mirzasoleiman. 2024. SmallToLarge (S2L): Scalable data selection for fine-tuning large language models by summarizing training trajectories of small models. In *The Thirty-Eighth Annual Conference on Neural Information Processing Systems*. Retrieved from https://openreview.net/forum?id=K9IGlMQpif

[417] Yizhe Yang, Huashan Sun, Jiawei Li, Runheng Liu, Yinghao Li, Yuhang Liu, Heyan Huang, and Yang Gao. 2023. Mindllm: Pre-training lightweight large language model from scratch, evaluations and domain applications. arXiv:2310.15777. Retrieved from https://arxiv.org/abs/2310.15777

[418] Zhou Yang, Zhaochun Ren, Wang Yufeng, Shizhong Peng, Haizhou Sun, Xiaofei Zhu, and Xiangwen Liao. 2024. Enhancing empathetic response generation by augmenting LLMs with small-scale empathetic models. arXiv:2402.11801. Retrieved from https://arxiv.org/abs/2402.11801

[419] Yunzhi Yao, Shaohan Huang, Wenhui Wang, Li Dong, and Furu Wei. 2021. Adapt-and-distill: Developing small, fast and effective pretrained language models for domains. arXiv:2106.13474. Retrieved from https://arxiv.org/abs/2106.13474.

[420] Mert Yazan, Suzan Verberne, and Frederik Situmeang. 2024. The impact of quantization on retrieval-augmented generation: An analysis of small LLMs. arXiv:2406.10251. Retrieved from https://arxiv.org/abs/2406.10251

[421] Rongjie Yi, Liwei Guo, Shiyun Wei, Ao Zhou, Shangguang Wang, and Mengwei Xu. 2023. EdgeMoE: Fast on-device inference of MoE-based large language models. arXiv:2308.14352. Retrieved from https://arxiv.org/abs/2308.14352

[422] Rongjie Yi, Xiang Li, Weikai Xie, Zhenyan Lu, Chenghua Wang, Ao Zhou, Shangguang Wang, Xiwen Zhang, and Mengwei Xu. 2024. PhoneLM: An efficient and capable small language model family through principled pre-training. arXiv:2411.05046. Retrieved from https://arxiv.org/abs/2411.05046

[423] Wen-Tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Katrin Erk and Noah A. Smith (Eds.). Association for Computational Linguistics, 201–206. DOI: https://doi.org/10.18653/v1/P16-2033

[424] Wangsong Yin, Mengwei Xu, Yuanchun Li, and Xuanzhe Liu. 2024. LLM as a system service on mobile devices. arXiv:240311805. Retrieved from https://arxiv.org/abs/2403.11805

[425] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. MetaMath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=N8N0hgNDRt

[426] Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Rankrag: Unifying context ranking with retrieval-augmented generation in LLMS. arXiv:2407.02485. Retrieved from https://arxiv.org/abs/2407.02485

[427] Zhongzhi Yu, Zheng Wang, Yuhan Li, Haoran You, Ruijie Gao, Xiaoya Zhou, Sreenidhi Reedy Bommu, Yang Katie Zhao, and Yingyan Celine Lin. 2024. EDGE-LLM: Enabling efficient large language model adaptation on edge devices via layerwise unified compression and adaptive layer tuning and voting. arXiv:2406.15758. Retrieved from https://arxiv.org/abs/2406.15758

[428] Jinliang Yuan, Chen Yang, Dongqi Cai, Shihe Wang, Xin Yuan, Zeling Zhang, Xiang Li, Dingge Zhang, Hanzi Mei, Xianqing Jia, et al. 2024. Mobile foundation model as firmware. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, 279–295.

[429] Sha Yuan, Hanyu Zhao, Zhengxiao Du, Ming Ding, Xiao Liu, Yukuo Cen, Xu Zou, Zhilin Yang, and Jie Tang. 2021. Wudaocorpora: A super large-scale Chinese Corpora for pre-training language models. *AI Open* 2, (2021), 65–68. DOI: https://doi.org/10.1016/j.aiopen.2021.06.001

[430] Xiaohan Yuan, Jinfeng Li, Dongxia Wang, Yuefeng Chen, Xiaofeng Mao, Longtao Huang, Hui Xue, Wenhai Wang, Kui Ren, and Jingyi Wang. 2024. S-Eval: Automatic and adaptive test generation for benchmarking safety evaluation of large language models. arXiv:2405.14191. Retrieved from https://arxiv.org/abs/2405.14191

[431] Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen Tse Huang, Pinjia He, Shuming Shi, and Zhaopeng Tu. 2024. GPT-4 is too smart to Be safe: Stealthy chat with LLMs via cipher. In *The Twelfth International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=MbfAK4s61A.

[432] Shengbin Yue, Wei Chen, Siyuan Wang, Bingxuan Li, Chenchen Shen, Shujun Liu, Yuxuan Zhou, Yao Xiao, Song Yun, Wei Lin, et al. 2023. Disc-lawllm: Fine-tuning large language models for intelligent legal services. arXiv:2309.11325. Retrieved from https://arxiv.org/abs/2309.11325

[433] Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2024. Mammoth: Building math generalist models through hybrid instruction tuning. In *The Twelfth International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=yLClGs770I

[434] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence?. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 4791–4800.

[435] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. *Adv. Neural Inf. Process. Syst*. 32, (2019). DOI: https://doi.org/10.5555/3454287.3455099

[436] Biao Zhang, and Rico Sennrich. 2019. Root mean square layer normalization. *Adv. Neural Inf. Process. Syst*. 32, (2019), 12381–12392. DOI: https://doi.org/10.5555/3454287.3455397

[437] Cheng Zhang, Jianyi Cheng, George A. Constantinides, and Yiren Zhao. 2024. LQER: low-rank quantization error reconstruction for LLMs. arXiv:2402.02446. Retrieved from https://arxiv.org/abs/2402.02446

[438] Collin Zhang, John X. Morris, and Vitaly Shmatikov. 2024. Extracting prompts by inverting LLM outputs. arXiv:2405.15012. Retrieved from https://arxiv.org/abs/2405.15012

[439] Chen Zhang, Dawei Song, Zheyu Ye, and Yan Gao. 2023. Towards the law of capacity gap in distilling language models. arXiv:2311.07052. Retrieved from https://arxiv.org/abs/2311.07052

[440] Dan Zhang, Ziniu Hu, Sining Zhoubian, Zhengxiao Du, Kaiyu Yang, Zihan Wang, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024. Sciglm: Training scientific language models with self-reflective instruction annotation and tuning. arXiv:2401.07950. Retrieved from https://arxiv.org/abs/2401.07950

[441] Di Zhang, Wei Liu, Qian Tan, Jingdan Chen, Hang Yan, Yuliang Yan, Jiatong Li, Weiran Huang, Xiangyu Yue, Dongzhan Zhou, et al. 2024. ChemLLM: A chemical large language model. arXiv:2402.06852. Retrieved from https://arxiv.org/abs/2402.06852

[442] Kaiyan Zhang, Jianyu Wang, Ermo Hua, Biqing Qi, Ning Ding, and Bowen Zhou. 2024. Cogenesis: A framework collaborating large and small language models for secure context-aware instruction following. arXiv:2403.03129. Retrieved from https://arxiv.org/abs/2403.03129

[443] Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. 2023. Loraprune: Pruning meets low-rank parameter-efficient fine-tuning. arXiv:2305.18403. Retrieved from https://arxiv.org/abs/2305.18403

[444] Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. TinyLlama: An open-source small language model. arXiv:240102385. Retrieved from https://arxiv.org/abs/2401.02385

[445] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. arXiv:2205.01068. Retrieved from https://arxiv.org/abs/2205.01068

[446] Xinran Zhang, Xin Yuan, Yunwei Li, and Yanru Zhang. 2019. Cold-Start representation learning: A recommendation approach with bert4Movie and movie2Vec. In *Proceedings of the 27th ACM International Conference on Multimedia*. ACM, 2612–2616.

[447] Yingtao Zhang, Haoli Bai, Haokun Lin, Jialin Zhao, Lu Hou, and Carlo Vittorio Cannistraci. 2024. Plug-and-play: An efficient post-training pruning method for large language models. In *The Twelfth International Conference on Learning Representations*.

[448] Yiming Zhang, Nicholas Carlini, and Daphne Ippolito. 2024. Effective prompt extraction from language models. arXiv:230706865. Retrieved from https://arxiv.org/abs/2307.06865

[449] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. 2024. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. *Adv. Neural Inf. Process. Syst*. 36, (2024), 34661–34710. DOI: https://doi.org/10.5555/3666122.3667628

[450] Bowen Zhao, Hannaneh Hajishirzi, and Qingqing Cao. 2024. APT: Adaptive pruning and tuning pretrained language models for efficient training and inference. In *Forty-First International Conference on Machine Learning*.

[451] Junchen Zhao, Yurun Song, Simeng Liu, Ian G. Harris, and Sangeetha Abdu Jyothi. 2023. LinguaLinked: A distributed large language model inference system for mobile devices. arXiv:2312.00388. Retrieved from https://arxiv.org/abs/2312.00388

[452] Juntao Zhao, Borui Wan, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. LLM-PQ: Serving LLM on heterogeneous clusters with phase-aware partition and adaptive quantization. arXiv:2403.01136. Retrieved from https://arxiv.org/abs/2403.01136

[453] Kun Zhao, Bohao Yang, Chen Tang, Chenghua Lin, and Liang Zhan. 2024. SLIDE: A framework integrating small and large language models for open-domain dialogues evaluation. arXiv:2405.15924. Retrieved from https://arxiv.org/abs/2405.15924

[454] Theodore Zhao, Mu Wei, J. Samuel Preston, and Hoifung Poon. 2023. Automatic calibration and error correction for large language models via Pareto optimal self-supervision. arXiv:2306.16564. Retrieved from https://arxiv.org/abs/2306.16564

[455] Youpeng Zhao, Ming Lin, Huadong Tang, Qiang Wu, and Jun Wang. 2024. Merino: entropy-driven design for generative language models on IoT devices. arXiv:240307921. Retrieved from https://arxiv.org/abs/2403.07921

[456] Zhengyun Zhao, Qiao Jin, Fangyuan Chen, Tuorui Peng, and Sheng Yu. 2022. PMC-patients: A large-scale dataset of patient summaries and relations for benchmarking retrieval-based clinical decision support systems. arXiv:2202.13876. Retrieved from https://arxiv.org/abs/2202.13876

[457] Zhanhui Zhou, Zhixuan Liu, Jie Liu, Zhichen Dong, Chao Yang, and Yu Qiao. 2024. Weak-to-strong search: Align large language models via searching over small language models. In *The Thirty-Eighth Annual Conference on Neural Information Processing Systems*. Retrieved from https://openreview.net/forum?id=dOJ6CqWDf1

[458] Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, Vol. 1 Long Papers, Association for Computational Linguistics, 3277–3287.

[459] Jiachen Zhu, Jianghao Lin, Xinyi Dai, Bo Chen, Rong Shan, Jieming Zhu, Ruiming Tang, Yong Yu, and Weinan Zhang. 2024. Lifelong personalized low-rank adaptation of large language models for recommendation. arXiv:2408.03533. Retrieved from https://arxiv.org/abs/2408.03533

[460] Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Zhenqiang Gong, et al. 2023. Promptbench: Towards evaluating the robustness of large language models on adversarial prompts. arXiv:2306.04528. Retrieved from https://arxiv.org/abs/2306.04528

[461] Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2023. A survey on model compression for large language models. arXiv:2308.07633. Retrieved from https://arxiv.org/abs/2308.07633

[462] Yun Zhu, Yinxiao Liu, Felix Stahlberg, Shankar Kumar, Yu Hui Chen, Liangchen Luo, Lei Shu, Renjie Liu, Jindong Chen, and Lei Meng. 2023. Towards an on-device agent for text rewriting. arXiv:2308.11807. Retrieved from https://arxiv.org/abs/2308.11807

[463] Yuanyuan Zhuang, and Jaekyeong Kim. 2021. A BERT-based multi-criteria recommender system for hotel promotion management. *Sustainability* 13, 14 (2021), 8039. DOI: https://doi.org/10.3390/su13148039

[464] Terry Yue Zhuo, Yujin Huang, Chunyang Chen, and Zhenchang Xing. 2023. Red teaming chatGPT via jailbreaking: Bias, robustness, reliability and toxicity. arXiv:2301.12867. Retrieved from https://arxiv.org/abs/2301.12867

[465] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, Zico Kolter, and Matt Fredrikson. 2023. J Universal and transferable adversarial attacks on aligned language models. arXiv:2307.15043. Retrieved from https://arxiv.org/abs/2307.15043

[466] Lixin Zou, Weixue Lu, Yiding Liu, Hengyi Cai, Xiaokai Chu, Dehong Ma, Daiting Shi, Yu Sun, Zhicong Cheng, Simiu Gu, et al. 2022. Pre-trained language model-based retrieval and ranking for web search. *ACM Trans. Web* 17, 1 (2022), 1–36. DOI: https://doi.org/10.1145/3568681

[467] Jingwei Zuo, Maksim Velikanov, Dhia Eddine Rhaiem, Ilyas Chahed, Younes Belkada, Guillaume Kunsch, and Hakim Hacid. 2024. Falcon Mamba: The first competitive attention-free 7B language model. arXiv:2410.05355. Retrieved from https://arxiv.org/abs/2410.05355

Received 2 January 2025; revised 18 August 2025; accepted 23 August 2025