

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, USA

TZUHAO MO, University of Pennsylvania, USA

QIUHAO LU, WANJING WANG, and RUI LI, UTHealth Houston, USA

JUNJIE XU, The Pennsylvania State University, USA

XIANFENG TANG and QI HE, Amazon, USA

YAO MA, Rensselaer Polytechnic Institute, USA

MING HUANG, UTHealth Houston, USA

SUHANG WANG*, The Pennsylvania State University, USA

Large language models (LLM) have demonstrated emergent abilities in text generation, question answering, and reasoning, facilitating various tasks and domains. Despite their proficiency in various tasks, LLMs like LaPM 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 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>.

*Corresponding author.

Authors' addresses: Fali Wang, fqw5095@psu.edu; Zhiwei Zhang; Xianren Zhang; Zongyu Wu, The Pennsylvania State University, University Park, USA; TzuHao Mo, University of Pennsylvania, Philadelphia, USA; Qiuha Lu; Wanjing Wang; Rui Li, UTHealth Houston, Houston, USA; Junjie Xu, The Pennsylvania State University, University Park, USA; Xianfeng Tang; Qi He, Amazon, Palo Alto, USA; Yao Ma, Rensselaer Polytechnic Institute, Troy, USA; Ming Huang, UTHealth Houston, Houston, USA; Suhang Wang, The Pennsylvania State University, University Park, USA, szw494@psu.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.

Manuscript submitted to ACM

CCS Concepts: • Computing methodologies → Natural language generation.

ACM Reference Format:

Fali Wang, Zhiwei Zhang, Xianren Zhang, Zongyu Wu, TzuHao Mo, Qiuhan Lu, Wanjing Wang, Rui Li, Junjie Xu, Xianfeng Tang, Qi He, Yao Ma, Ming Huang, and Suhang Wang. 2018. A Comprehensive Survey of Small Language Models in the Era of Large Language Models: Techniques, Enhancements, Applications, Collaboration with LLMs, and Trustworthiness. *J. ACM* 37, 4, Article 111 (August 2018), 76 pages. <https://doi.org/XXXXXX.XXXXXXX>

1 INTRODUCTION

The evolution of neural language models (LMs) from BERT's [77] pre-training and fine-tuning paradigm to T5's [250] pre-training plus prompting approach, and finally to GPT-3's [33] 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 [311], programming [227, 294], recommendation systems [327], information retrieval [38, 136, 204, 281], mobile-device control [80], scientific discovery [275, 379], medical question answering [30, 325], and legal question answering [10]. In particular, the recent emergence of proprietary commercial models including ChatGPT, Bard, and Claude, and open-sourced models such as Llama [84, 301, 302] 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 [149], 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 [84].

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 [84], trained on 16K 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 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 [299]; (4) To boost performance in specialized domains like 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 [44, 112, 139, 244, 327].

Recently, small language models (SLMs) have shown great potential in alleviating these issues while achieving performance comparable to LLMs for domain-specific problems [1, 24, 104, 128, 199, 243, 296, 299, 352, 382]. We Owing to fewer parameters, SLMs excel in efficiency, cost, flexibility, and customization. They provide significant computational

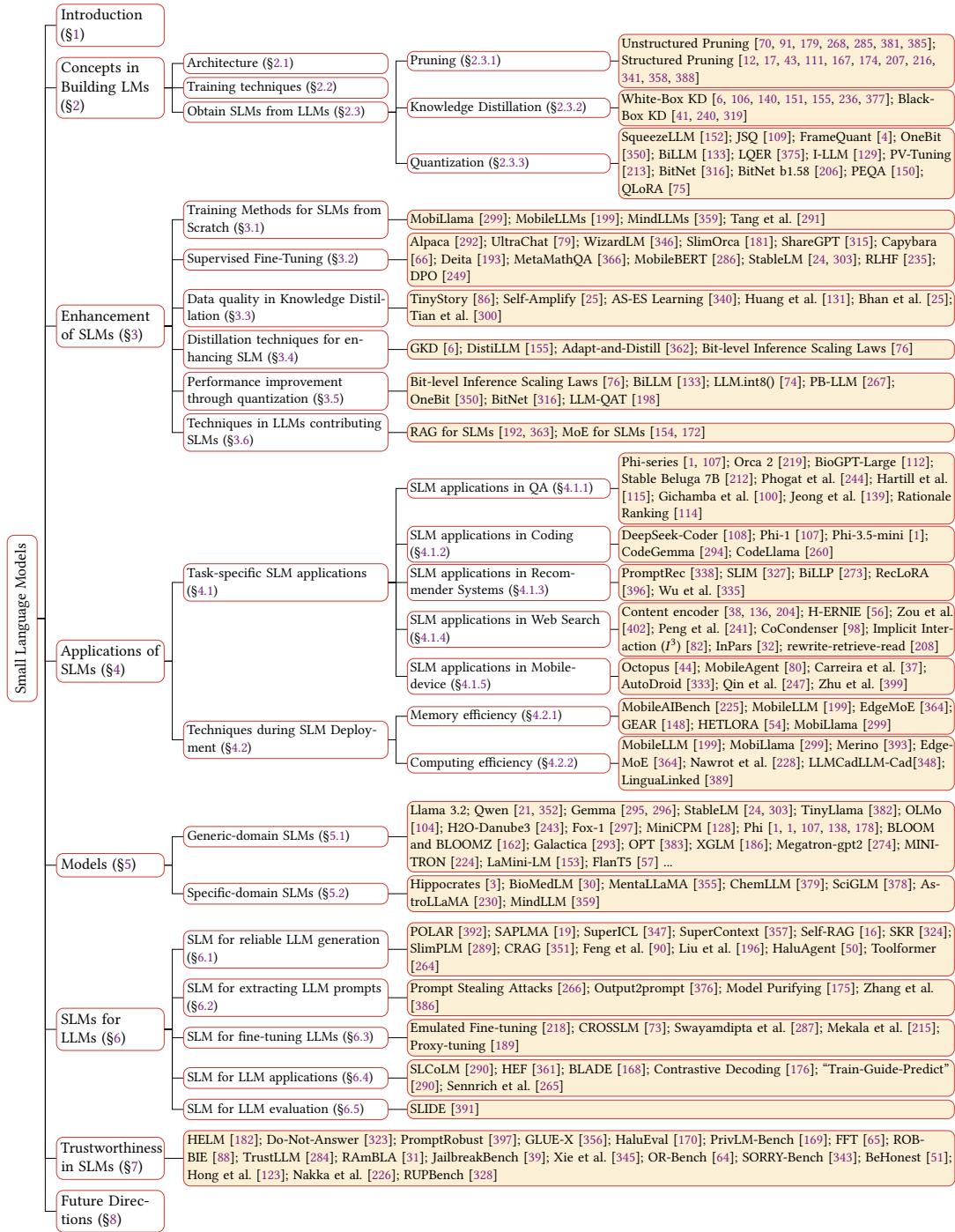


Fig. 1. Overview of Small Language Models.

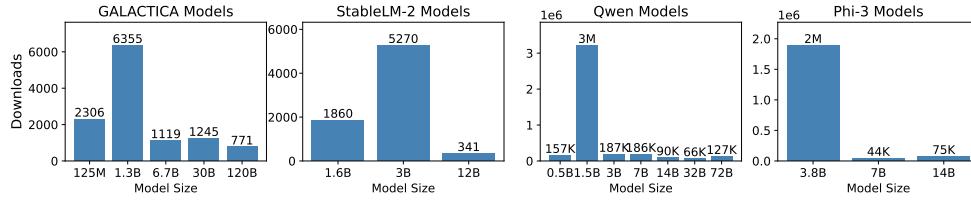


Fig. 2. Download Statistics Last Month in Huggingface for LLMs with Various Model Sizes, obtained on October 7, 2024.

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 & 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, small language models 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 [199], while others consider the term “small language model” relative to the larger counterparts [163, 290, 327], with no consensus on a unified definition in the current landscape of LLMs. Research suggests SLMs for mobile devices, typically possessing around 6GB of memory, consist of sub-billion parameter models [199], whereas others classify models with up to 10 billion parameters as small, noting their lack of emergent abilities [94]. 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 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 training techniques optimized for SLMs, such as quantization-aware training [198, 316, 350] and selective architectural component choices [199, 299], aim to enhance performance in specific applications [32, 44, 244, 260, 338]. These methods have led to the development of numerous open-source, general-purpose, and domain-specific SLMs [3, 24, 30, 296, 352, 378]. Beyond their inherent capabilities, SLMs can also serve as a module or effective proxies for enhancing LLMs [218, 266, 339, 349, 361, 392]. Despite the commendable performance of SLMs, it is crucial not to overlook their credibility issues, such as the risks of producing hallucinations and privacy breaches [81, 85, 123, 158, 158, 220, 226, 242, 312, 328, 370]. However, currently, there is no comprehensive survey thoroughly exploring these works on SLMs in the era of LLMs. Therefore, this paper offers the first comprehensive survey that analyzes various aspects of SLMs in the LLM era and their future directions. The overview structure of our paper 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, quantization, and leveraging LLM-enhancing technologies to optimize SLMs.

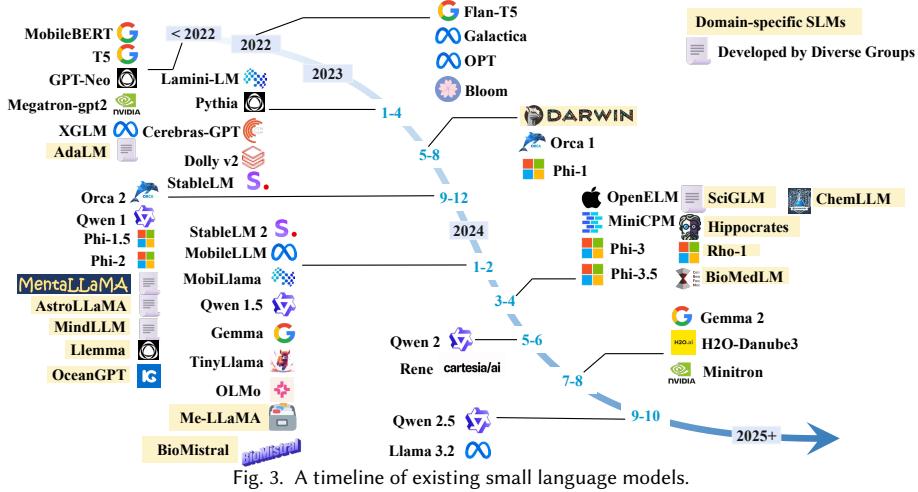


Fig. 3. A timeline of existing small language models.

- 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 investigate the trustworthiness issues of SLMs, including hallucination and privacy concerns, by providing a taxonomic summary of current evaluation methods.

2 FOUNDATIONAL CONCEPTS IN BUILDING LANGUAGE MODELS

This section will introduce foundational concepts and background knowledge that are important for language models. We will introduce the basic concepts both in architecture and the training process respectively. The advanced training strategy to improve SLM performance will be introduced in Section 3.

2.1 Architecture of SLMs

Generally, the architecture of small language models (SLMs) is based on LLMs but optimized for computational efficiency and scalability. SLMs commonly employ the Transformer architecture [308] (see Figure 4), which utilizes self-attention mechanisms to manage long-range text dependencies, essential for maintaining performance with constrained resources.

2.1.1 Transformer [308] for SLMs. The Transformer's self-attention mechanism [308] allows SLMs to efficiently capture contextual information across longer sequences, even with limited resources. SLM Transformers generally adopt an

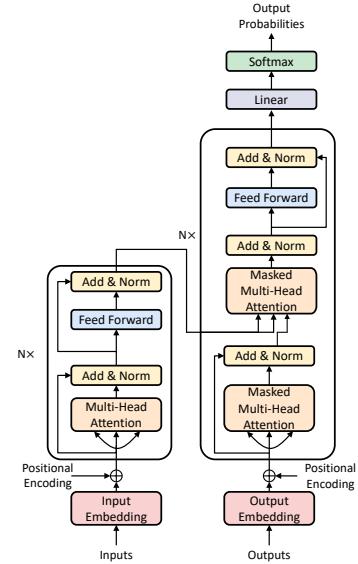


Fig. 4. Transformer architecture.

Manuscript submitted to ACM

encoder-decoder structure featuring self-attention mechanisms, feedforward networks, positional embeddings, and layer normalization.

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

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

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) [308] 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

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \text{head}_2, \dots, \text{head}_h)\mathbf{W}^O, \text{ with } \text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V), \quad (1)$$

Each head in the Multi-Head Attention 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)** [270] 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)** [188] 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** [67, 68] 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 $\text{FFN}(x) = \sigma(x\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2$. 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) **GELU (Gaussian Error Linear Unit)** [121] 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 [77] and GPT [248] 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) **SiLU (Sigmoid Linear Unit)** [87] 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) **SwiGLU (Swish-Gated Linear Units)** [271] 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, a smooth non-linear activation, is expressed as $\text{Swish}(x) = x \cdot \text{sigmoid}(x)$. This combination enhances expressiveness and computational efficiency, making it a preferred choice in advanced models such as the Qwen series [352].

Positional Embeddings in Transformer models [308] 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:

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \quad PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \quad (2)$$

where pos represents the position within the sequence, i is the dimension index, and d_{model} is the dimensionality of the model. This method alternates between sine and cosine functions across dimensions. BERT-family models [77, 195], in contrast, employ learned positional embeddings, enhancing flexibility and adaptation to various text structures. To improve the model's capacity for understanding the relative positions of tokens within a sequence, **Rotary Positional Embedding (RoPE)** [282] 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 [165] stabilizes the training process by normalizing layer outputs, accelerating convergence. Two types of layer normalization are commonly used [165]: (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, **RMS Norm (Root Mean Square Layer Normalization)** [374] 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 i -th input, and ϵ is a small constant to prevent division by zero.

2.1.2 Mamba Model [69, 105]. Mamba leverages a refined version of the Transformer architecture, which includes improvements in multi-head attention and feedforward networks. Specifically, Mamba uses **dynamic attention heads**, which adjusts the number of active attention heads based on the complexity of the input sequence. This approach mirrors the Multi-Head Attention mechanism within the Transformer but with the added benefit of dynamically scaling the attention resources to match the task's demands. This dynamic adaptation allows Mamba to maintain high performance while minimizing unnecessary computations for simpler sequences, making it highly efficient. Additionally, Mamba incorporates **adaptive feedforward networks**, which align closely with the Feedforward Network architecture in Transformers. These adaptive networks dynamically adjust the depth of the feedforward layers based on the input's complexity, ensuring that computational resources are allocated efficiently. This adaptive mechanism preserves power

for more demanding tasks while keeping the model lightweight for simpler tasks, directly improving the efficiency of the Transformer architecture in SLMs.

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 Pretrain. Typically, pre-training paradigms for language models are divided into encoder-based and decoder-based approaches. Encoder-based models, such as BERT [77], 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 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}_{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 [248], 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}_{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. **Tokenization Strategies** play a vital role in handling diverse vocabularies without increasing model size. Advanced methods such as Byte-Pair Encoding (BPE) [95] and WordPiece [280] break text into subwords [77], allowing the model to manage rare and compound words efficiently. These strategies ensure that SLMs maintain a balance between vocabulary coverage and model compactness, crucial for improving generalization while minimizing computational demands.

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, such as Low-Rank Adaptation (LoRA), prefix-tuning, and adapter modules, are particularly effective for SLMs. **Low-Rank Adaptation (LoRA)** [127] modifies Transformer weights by introducing trainable low-rank matrices \mathbf{A} and \mathbf{B} for efficient fine-tuning, avoiding significant alterations to pre-trained weights. The update is represented as:

$$\Delta \mathbf{W} = \mathbf{AB}^T \quad (3)$$

The fine-tuned weight matrix used in Transformer operations then becomes:

$$\mathbf{W}_{\text{ft}} = \mathbf{W} + \alpha \Delta \mathbf{W} \quad (4)$$

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** [177] 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** [125] 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:

$$\text{Adapter}(\mathbf{h}) = \mathbf{h} + \mathbf{W}_{\text{up}} \cdot \sigma(\mathbf{W}_{\text{down}} \cdot \mathbf{h} + \mathbf{b}_{\text{down}}) + \mathbf{b}_{\text{up}} \quad (5)$$

where \mathbf{h} is the input hidden state, \mathbf{W}_{down} and \mathbf{W}_{up} are the projection matrices, \mathbf{b}_{down} and \mathbf{b}_{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: $x_i = \arg \max_x P(x | x_{<i})$, where x_i is the token with the highest probability at the i -th 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 SLM from LLM

Obtaining a small language model (SLM) from a large language model (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, knowledge distillation, and quantization. Pruning removes less critical parameters, reducing model size while aiming to maintain performance. Knowledge distillation 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 [113]. 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* [320, 398]. An illustration of unstructured pruning and structured pruning is shown in Figure 5.

Unstructured Pruning [70, 91, 179, 268, 285, 381, 385] prunes an LLM by removing weights individually without considering its internal structure.

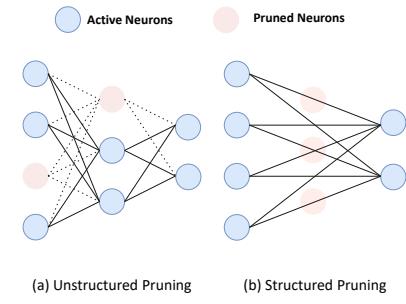


Fig. 5. Visualizations of unstructured and structured pruning.

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 [91] is a representative unstructured pruning method that can reduce large-scale GPT models like OPT-175B [383] and BLOOM-176B [162] to up to 60% sparsity using a novel sparse regression solver. Wanda [285] 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 Low-rank Adaptation (LoRA) [127], such as LoRAPrune [381].

Structured Pruning [12, 17, 43, 111, 167, 174, 207, 216, 341, 358, 388], 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 [216] 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 [207] 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 [174] 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.

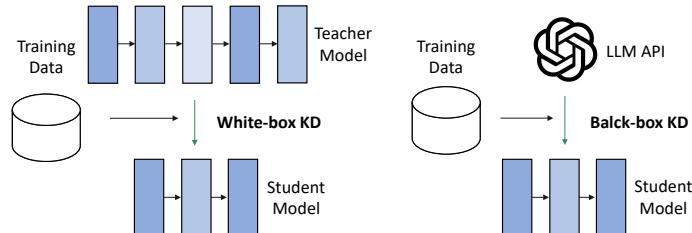


Fig. 6. Illustration of white-box and black-box knowledge distillation [217].

2.3.2 Knowledge Distillation. Knowledge distillation (KD) compresses a larger teacher model into a smaller student model by training the student to mimic the teacher’s outputs [122]. 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 [320, 353, 398] as shown in Figure 6. In **White-Box KD**, the student has access to the teacher’s internal states or output distributions [6, 106, 140, 151, 155, 236, 377]. Generalized Knowledge Distillation (GKD) [155] 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 [41, 240, 319]. Methods like Distilling Step-by-Step [126] use teacher-generated rationales to train smaller models, improving

Table 1. Representative quantization methods.

Mehtods	Bit	Type	Technical Contribution	Problems
SqueezeLLM [152]	3-bit	PTQ	Sensitivity-based non-uniform quantization, dense and sparse decomposition	ultra-low bit quantization
JSQ [109]	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 [350]	1-bit	PTQ	Quantization-aware knowledge distillation	1-bit quantization
BiLLM [133]	1-bit	PTQ	Crucial Weights Selection, Block-based error compensation	1-bit quantization
LQER [375]	Flexible	PTQ	Quantization Error Minimization	better compression-accuracy trade-offs
I-LLM [129]	Flexible	PTQ	Fully-Smooth Block-Reconstruction, Dynamic Integer-only MatMul and Integer-only Non-linear Operators	Integer-only Quantization
PV-Tuning [213]	1-bit/2-bit	PTQ	PV algorithm	better compression-accuracy trade-offs.
BitNet [316]	1-bit	QAT	1-bit Transformer Architecture	1-bit quantization
BitNet b1.58 [206]	{1, 0, 1}	QAT	Ternary Parameters	1-bit quantization
PEQA [150]	Flexible	QAT	Quantization Scales Optimization	Parameter-Efficient Finetuning
QLoRA [75]	NF4	QAT	4-bit NormalFloat and Double Quantization	Parameter-Efficient Finetuning

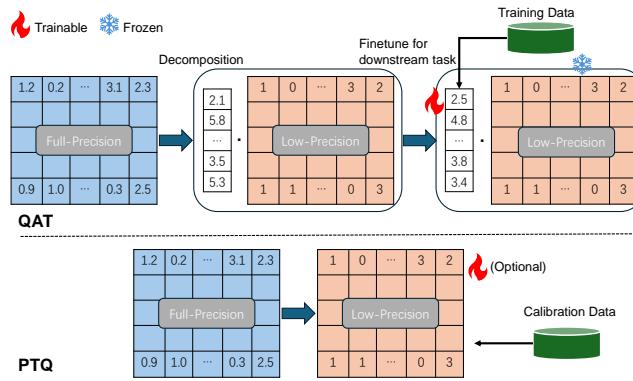


Fig. 7. Illustration of quantization-aware training (QAT) and post-training quantization (PTQ).

performance with fewer examples. LaMini-LM [336] 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 *Quantization-Aware Training* (QAT). Figure 7 illustrates the two quantization methods. **Post-Training Quantization**, 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 w ; the linear operation can be expressed as $y = wx$, while the quantized version is given by $y = Q(w)x$. Generally, the quantization function Q is defined as [184]:

$$Q(w) = \Delta \cdot \text{Round}\left(\frac{w}{\Delta}\right), \quad \Delta = \frac{\max(|w|)}{2^{N-1}},$$

where N is the number of quantization bits, and Δ is the quantization scale factor determined by the absolute maximum value of w . **Quantization-Aware Training (QAT)** enhances LLM efficiency by incorporating quantization directly into the training process, often resulting in higher accuracy compared to PTQ. During QAT, the forward pass utilizes

Table 2. Comparison of Model Compression Techniques

Criteria	Pruning	Knowledge Distillation	Quantization	Low-Rank Techniques
Definition	Removes unneeded parameters	Transfers knowledge from a larger to a smaller model	Lowers the precision of parameters	Uses low-rank decomposition on weights
Goal	Reduces size and computation	Shrinks model while retaining performance	Decreases size and speeds up processing	Reduces parameters and computation
Method	Cuts weights or layers based on importance	Smaller model mimics larger model's output	Converts parameters to lower precision	Decomposes matrices into smaller components
Advantages	Reduces size and computation significantly	Preserves performance in smaller models	Speeds up inference, less storage	Efficient, mostly preserves performance
Disadvantages	May reduce accuracy, irregular memory access	Resource-intensive, requires large teacher model	Potential accuracy loss, may need specific hardware	Effectiveness varies, requires rank selection
Model Size Impact	High reduction	Significant reduction through knowledge transfer	High, tied to precision reduction	Moderate, reduces redundant parameters
Performance Impact	Possible degradation if over-pruned	Maintains if well distilled	Minor to moderate loss, depends on method	Mostly retains performance, may need tuning
Complexity	Moderate, needs parameter evaluation	High, involves dual-model training	Moderate, varies with precision level	Moderate, requires matrix factorization
Use Cases	Resource-limited settings	Efficient model creation for limited resources	Fast inference needs, edge devices	When weight matrices are redundant

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 post-training quantization methods are summarized in Table 1, detailing precision, addressed problems, and technical contributions of each method.

2.3.4 Low-Rank Techniques. Low-rank techniques compress LLMs by approximating high-dimensional weight matrices with two lower-dimensional matrices, reducing computational and memory requirements. A matrix \mathbf{W} of size $m \times n$ is approximated as $\mathbf{W} \approx \mathbf{A} \times \mathbf{B}$, where \mathbf{A} is $m \times r$ and \mathbf{B} is $r \times n$, with r much smaller than m or n , significantly reducing the number of parameters. Building on this concept, Ji et al. [141] propose a low-rank compression 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. [54] tackles 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.

2.4 Comparison between Different Compression Methods

To summarize the key differences between pruning, knowledge distillation, quantization, and low-rank techniques for model compression, we present a comparative table highlighting their definitions, goals, advantages, disadvantages, and typical use cases, as shown in Table 2.

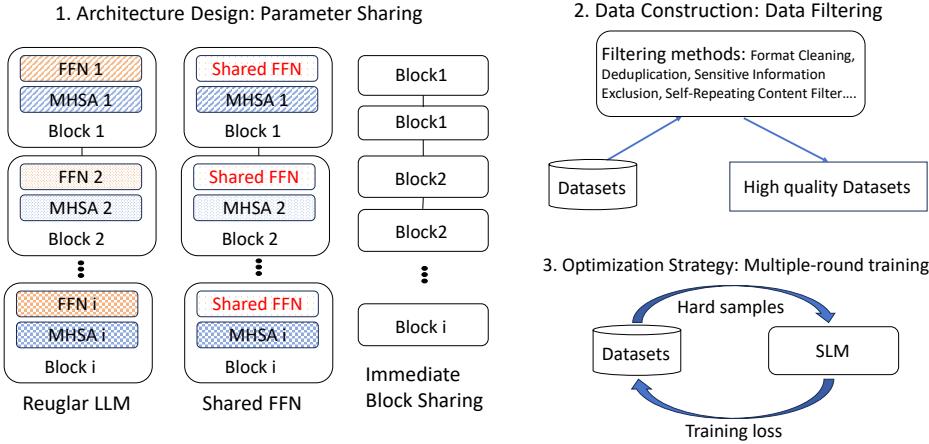


Fig. 8. Innovative Training Methods for Small Language Models from Scratch

3 ADVANCED ENHANCEMENT STRATEGIES FOR SMALL LANGUAGE MODELS

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 knowledge distillation and quantization techniques, and techniques frequently used in LLMs such as mixture-of-experts to enhance SLM for specific applications. A summary of enhancement techniques is also summarized in Table 3.

3.1 Innovative Training Methods for Small Language Models from Scratch

In scenarios with limited resources, we aim to train small language models to provide efficient, cost-effective solutions tailored for specific domains, while still maintaining competitive performance with larger models. Training small language models (SLMs) from scratch involves unique strategies that diverge significantly from those used for large language models (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 8, 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 8, parameter sharing is achieved by two approaches: (i) a single Feed-Forward Network (FFN) module is shared by every transformer layer, and (ii) entire transformer blocks are shared. For example, to design language models for scenarios that require on-device processing ability and energy, **FFN layer sharing/reusing** (see Figure 8 (1)) can be a potential option. By sharing FFN layers, the model 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 [299] which surpasses the performance of existing SLMs of comparable size. Generally, deeper and thinner models consistently perform better than shallower and wider ones [199]. Based on this observation, **Transformer Block-wise Sharing** is another 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. Experiments show

Table 3. Advanced enhancement methods for SLM.

Topic	Method	Main Contribution
Training from Scratch	MindLLM [359]	Bilingual pre-trained lightweight models with advanced architectural features
	MobiLlama [299]	An on-device SLM with dual objectives of model capability and efficiency through parameter sharing
	MobileLLM [199]	A deep and thin architecture with advanced techniques to optimize LLM deployment on mobile devices
Supervised Fine-tuning	MobileBERT [286]	Compact BERT that can fine-tune on downstream tasks.
	Alpaca 7B [292]	52k ChatGPT-generated instruction-following examples from 175 self-instructed seed tasks to tune Llama 7B [301].
	RLHF [235]	Gather human-preferred data, train a reward model, and fine-tune the LM using reinforcement learning.
	DPO [249]	Adjust the log probabilities of preferred versus non-preferred responses using a dynamic weighting mechanism, preventing model degradation issues.
Data Quality in KD	TinyStory [86]	Small language models can generate coherent stories using a child-friendly dataset and foundational vocabulary
	AS-ES [340]	Categorize reasoning steps into extractive and abstractive segments for enhanced small model CoT capabilities
	Self-Amplify [25]	A proxy-free, self-generating rationale method for small language models to automate CoT data annotation
Distillation for SLM	GKD [6]	Align training and inference distributions using on-policy sequences and flexible divergence measures
	DistiLLM [155]	A skew KL divergence loss for stability and an adaptive off-policy approach for efficient student-generated output utilization.
	Adapt-and-Distill [362]	Enhances small models by first domain adapting both teacher and student models before distillation
Quantization	SmoothQuant [342]	8-bit quantization by balancing quantization difficulty between activations and weights using a per-channel scaling transformation
	BiLLM [133]	Post-training quantization by using a Hessian-based metric to identify salient weights and applying binary residual approximation and optimal splitting search for precise quantization.
	LLM-QAT [198]	Quantization by data-free knowledge distillation and fine-tuning with logit distillation from the full-precision model
	PB-LLM [267]	Selectively binarizes non-salient weights while preserving salient ones in higher precision, balancing model compression and accuracy
	OneBit [350]	Sign-Value-Independent Decomposition to achieve near 1-bit quantization, balancing extreme compression with minimal performance loss
	BitNet [316]	A 1-bit Transformer architecture with BitLinear layers and quantization-aware training, optimizing low-precision representation for better accuracy and training stability.
	BitNet b1.58 [206]	Enhanced BitNet by introducing a ternary weight system, achieving full-precision performance from 3 billion parameters with reduced memory and latency costs
LLMs for SLM	Ma et al. [209]	An adaptive filtering and re-ranking paradigm combining LLMs and SLMs improves Information Extraction tasks
	MoQE [154]	Apply quantization only to expert weights and achieve better performance than the dense model trained on the same dataset.
	SLM-RAG [192]	Suggests that SLMs equipped with Retrieval-Augmented Generation (RAG) can perform comparably to Large Language Models (LLMs).

that simply reusing the same transformer blocks can still improve performance [199]. This improvement likely comes from the fact that repeating the blocks effectively maintains the model's depth and complexity, allowing it to capture more information without adding new parameters. While repeat-all-over sharing generally yields better performance,

block-wise immediate sharing optimizes cache utilization. This is because the shared weights remain in the cache and can be processed again immediately. This technique is applied in MobileLLMs [199] which has 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 small language models, the emphasis on data quality surpasses that of quantity and diversity [359]. Experiments demonstrate that using a quality filtering approach to remove low-quality data can lead to improved performance in SLMs [359]. 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) Format cleaning, which involves removing HTML, CSS, and JS identifiers, as well as non-textual elements, to ensure clean text input; (ii) Low-quality content filtering that excludes webpages with low text-to-content ratios; (iii) Deduplication using Locality-Sensitive Hashing (LSH), particularly the SimHash algorithm [71, 261], to prevent content redundancy; (iv) Sensitive information exclusion through heuristics and a sensitive word vocabulary to filter out offensive and illegal content, with special tokens replacing private information; and (v) A self-repeating content filter that eliminates repetitive phrases typical of advertisements, enhancing the training dataset's information value [40, 359]. These steps collectively ensure that training data has high-quality, informative texts. SLMs also significantly benefit from these techniques. For example, MindLLMs [359], which are bilingual lightweight language models (available in 1.3B and 3B versions), adopt these data processing techniques and achieve improved capability acquisition, while mitigating issues like catastrophic forgetting.

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 [291]. Considering some examples are hard to fit, hard examples can be trained with a high probability [291]. For each round of training, the data sampling probability is updated according to the overall loss of that sample. Experiments results show that two rounds of training and a 50% sampling rate are a good trade-off between performance and training efficiency. Tang et al. [291] 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 large language models with similar model sizes on multiple benchmark datasets, achieving an average performance increase of 8.87%.

Insights: We draw several key insights from the training techniques of SLMs:

- For parameter sharing techniques, maintaining complexity and depth of model structure is essential for maintaining model performance (e.g., shared Feed-Forward Networks [299] and transformer blocks) [199].
- Data quality is more important than data quantity in the effectiveness of SLMs [359].
- Different from LLMs, we can take advantage of the compact nature of SLMs and employ more flexible training strategies, such as multiple-round training [291].

3.2 Supervised Fine-Tuning (SFT) for Enhancing SLM performance

Supervised Fine-Tuning (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 dictated by the training data. For example, various models, such as Alpaca [292], UltraChat [79], WizardLM [346], SlimOrca [181], ShareGPT [315], Capybara

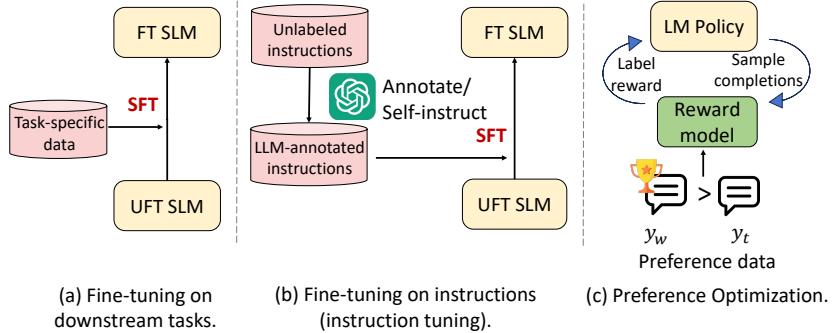


Fig. 9. Fine-tuning for Enhancing SLMs

[66], Deita [193], and MetaMathQA [366], incorporates a suite of conversational datasets to enhance their capabilities in context-aware dialogue and instruction adherence. Usually, as shown in Figure 9, existing SFL methods can be categorized into 3 categories: (i) *Classical fine-tuning with downstream data* [77, 248] 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 [392], or optimizing for mobile devices with MobileBERT [286]. (ii) *Instruction tuning* with LLM-generated data [79, 181, 292] or human-generated questions with LLM annotations [315] aims to align generative models with specific instructions, enhancing their instruction-following and reasoning capabilities. For example, Alpaca 7B [292] uses 52k ChatGPT-generated instruction-following examples from 175 self-instructed seed tasks to tune Llama 7B [301]. Meanwhile, StableLM [24, 303] is trained on the Restruct-v1 dataset, which includes summarization, question-answering, and sentiment analysis tasks, using instruction data from [202]. (iii) *Preference optimization with human feedback* [235, 249, 315] aims to better align language models with human preferences. Reinforcement Learning from Human Feedback (RLHF) [235] gathers human-preferred data, trains a reward model, and fine-tunes the LM using reinforcement learning. Direct Preference Optimization (DPO) [249] 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 & 3B apply SFT and DPO in post-training to enhance alignment with instructions and human preferences.

3.3 Data Quality in Knowledge Distillation (KD)

Transitioning from the discussion on training SLMs from scratch, this section delves into the critical role of data quality in Knowledge Distillation (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 10, 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 Other Models. Due to the limitations of model size, studies have shown that training SLMs requires simple and comprehensible data [86, 163, 340]. **TinyStory** [86] demonstrates that language models with a relatively small number of parameters (tens of millions) can still generate coherent stories tailored for children aged 3-4

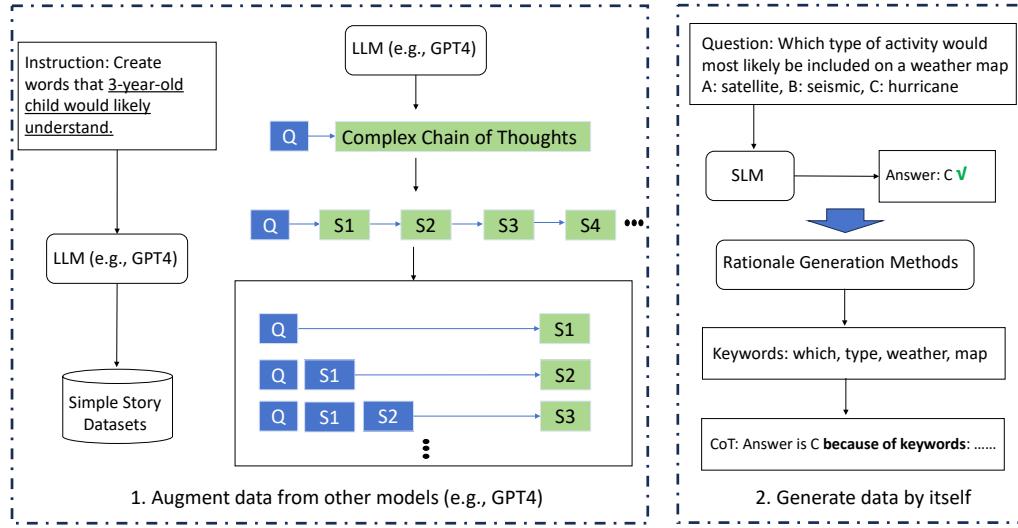


Fig. 10. Data Quality in Knowledge Distillation (KD)

years. This is achieved by prompting GPT-3.5 or GPT-4 [2] to create simple and easily understandable stories from three keywords selected from a foundational vocabulary of 1,500 words. The generated stories are then used to train SLMs, enabling them to produce similar narratives. This approach shows that varied 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. Many efforts to enhance the Chain-of-Thought (CoT) capabilities of small models involve using LLMs to generate high-quality CoT data. These data are then employed to train small models in an end-to-end fashion to mimic the CoT reasoning process [210, 340]. **AS-ES Learning** [340] argues that previous methods often overlook the limited capacity of small models to learn complex reasoning, despite being provided with very detailed reasoning processes. Even these detailed processes still require more nuanced capabilities, such as extraction and abstraction. Therefore, this study introduces a novel training paradigm that categorizes reasoning steps into extractive segments, which remind the model of the context and set the stage for subsequent conclusions, and abstractive segments that infer additional insights not explicitly stated in the context.

Augment Data from Itself. Besides distilling data from other LLMs, language models can also train on their own outputs [25, 131, 300]. 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 [131]. Similarly, SLMs can generate their training data with the aid of existing rationale generation methods. **Self-Amplify** [25] notes that human annotation of Chain-of-Thought (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

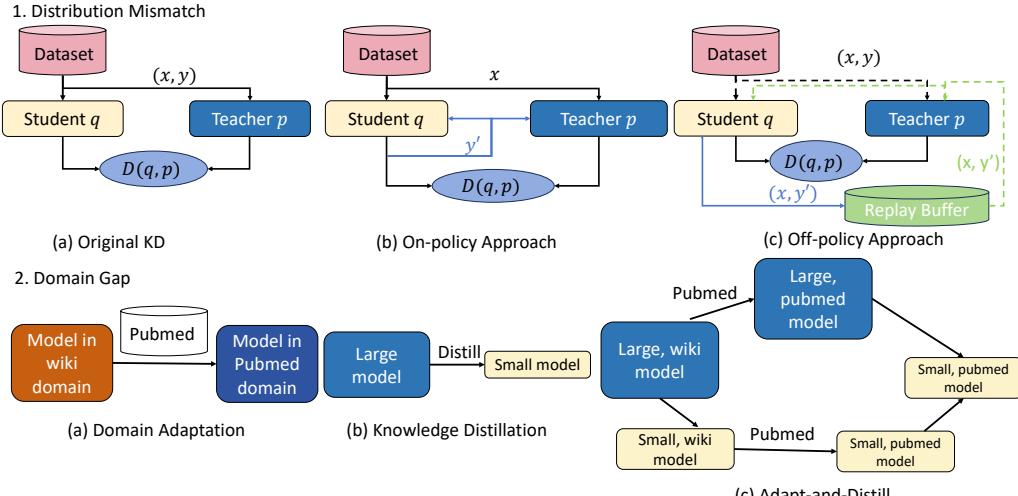


Fig. 11. Distillation Techniques for Enhancing SLM Performance

distilling knowledge from LLMs to SLMs, ensuring that the smaller models can maximize their performance gains. As shown in Figure 11, two main gaps between LLMs and SLMs lead to challenges in distillation: *distribution mismatch* and *domain gap*. *Distribution mismatch* [6, 155] 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* [362] 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 knowledge distillation, illustrated in Figure 11 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 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 11 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. Generalized Knowledge Distillation (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 11 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** [155] employs this off-policy approach and improves the efficiency of KD.

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 11 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 11 Domain Gap (b), Knowledge distillation 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 prior to knowledge distillation. As illustrated in Figure 11 Domain Gap (c), **Adapt-and-Distill** [362] tackles the domain gap by distilling general large models into smaller ones. This paper 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.

Insights: Here are some insights from distillation techniques:

- Sampling SLM outputs during the training process is the main approach to resolving distribution mismatch.
- Techniques like Adapt-and-Distill address the domain gap by first adapting both the teacher (LLMs) and the student (SLMs) models to the target domain before proceeding with distillation.

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 7, we categorize these quantization methods into two main approaches: Post-Training Quantization (PTQ), where quantization is conducted on a well-trained fixed model, and Quantization-Aware Training (QAT), where quantization is integrated into the training process. This section introduces advanced techniques in PTQ and QAT respectively.

Post-Training Quantization (PTQ) primarily includes weight quantization and activation quantization. Weight quantization aims to quantize model parameters while preserving performance. **GPTQ** [92] compresses LLMs to 4-bit or 2-bit by quantizing weights layer-by-layer to minimize layer-wise quantization errors. **PB-LLM** [267], applicable to both PTQ and QAT, retains the most salient weights while binarizing the rest based on magnitudes. **BiLLM** [133], 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 few bits and introducing significant errors. To address this, **LLM.int8()** [74] isolates outlier features for 16-bit processing and handles the rest in 8-bit. **SmoothQuant** [342] 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.

Quantization-Aware Training (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** [267] 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 paper explores both post-training quantization (PTQ) and quantization-aware training (QAT) to fine-tune and recover the performance of partially binarized models, achieving a balance between compression and accuracy. **OneBit** [350] and **BitNet** [316] 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** [206] improves on the original BitNet by introducing a ternary matrix weight system -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** [198] employs data-free knowledge distillation, 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 post-training quantization methods.

Insights: Insights drawn from quantization strategies include:

- Post-Training Quantization techniques primarily focus on quantizing model weights, where selecting salient weights is crucial. Beyond weight quantization, handling outliers in activation signals is a significant challenge in quantizing activations.
- Quantization-Aware Training methods show that low-bit quantization (e.g., 1-bit models) requires additional tuning to maintain performance. Knowledge can be distilled from the model before quantization to the quantized model.

3.6 Techniques in LLMs Contributing to SLMs

To enhance the performance of LLMs, various techniques such as Retrieval-Augmented Generation (RAG) and Mixture of Experts (MoE) are employed. This section discusses their potential to maintain or improve the performance of SLMs within constrained computational budgets. However, effectively integrating these advanced techniques into SLMs, which have inherently limited capabilities, remains an open 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 gain a better understanding of 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 [363] compares the performance of the original FP16 and the quantized INT4 on multiple 7B and 8B LLMs, indicating that for a 7B LLM already performing effectively, quantization does not compromise its performance or its ability to reason over long contexts. Another study [192] compares SLMs at the 7B level with RAG to larger models such as GPT-3.5 and GPT-4 [2], suggesting that SLMs equipped with Retrieval-Augmented Generation 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.

Mixture-of-Experts (MoE) offers an efficient scaling strategy for LLMs through expert parallelism. However, this approach requires significant memory overhead, necessitating model compression techniques for its adoption in SLMs. Several studies [154, 172] have found that traditional quantization techniques are less effective for MoE models. The Mixture of Quantized Experts (**MoQE**) [154] demonstrates that expert layers in MoE models are more resilient to quantization compared to traditional feedforward network (FFN) layers. They apply quantization only to expert weights and achieve superior performance compared to dense models trained on the same dataset. Similarly, another study [172] suggests that since FFN weights only engage with a subset of input tokens, more bits should be allocated to attention weights. These studies collectively indicate that in MoE architectures, FFN layers should be either quantized or allocated fewer bits.

Insights: Here are some insights from LLM techniques that contribute to SLMs:

- Implementing RAG in SLMs shows significant promise. Developing techniques to better tailor retrieved information for SLMs could be particularly beneficial.
- Quantization methods can effectively compress large MoE models by reducing bit allocation on FFN layers. Future research might explore the potential of training small MoE models from scratch.

4 APPLICATIONS OF SMALL LANGUAGE MODELS

In this section, we delve into the applications of small language models (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, employing specialized techniques to enhance SLM performance. Additionally, deploying SLMs often involves considerations of memory and runtime efficiency, which are crucial for 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. Question-answering 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 Question-Answering. Question-answering (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

Table 4. Task-specific SLM Applications

Aspect	Representative work	Key point
SLM in QA	Alpaca [292]	Tune Llama 7B [301] using 52k ChatGPT-generated examples from 175 seed tasks.
	Stable Beluga 7B [212]	Employ explanation tuning to Llama-2 7B [302] on an Orca-style dataset with explanatory LLM answers. capabilities
	Fine-tuned BioGPT Guo et al. [112]	Fine-tuning BioGPT (1.6B) [205] on PubMedQA.
	Financial SLMs [244]	Transfer financial knowledge from GPT-4 [2] to multiple SLMs, such as Phi-3-Mini [1], via program of thought (PoT).
	ColBERT [100]	Fetch retrieval documents for SLMs to answer complex domain-specific questions.
	T-SAS [139]	Enhance SLMs adaptability with self-generated pseudo labels.
SLM in Coding	Rationale Ranking [114]	For OOD questions, generate intermediate reasoning steps or rationales.
	Phi-3.5-mini [1]	New addition to the Phi-3 series and focus on high-quality, reasoning-dense data.
	TinyLlama [382]	A transformer model with 1.1 billion parameters, has been trained from scratch on a massive corpus of 3 trillion tokens.
	CodeLlama [260]	A derivative of Llama 2, undergo a rigorous fine-tuning process on domain-specific datasets
SLM in Recom-mendation	CodeGemma [294]	Stemming from Google DeepMind's Gemma framework, also exhibit a focused approach to enhancing coding capabilities through fine-tuning.
	PromptRec [338]	Training on prompt templates
	SLIM [327]	Step-by-step Knowledge Distillation
	BiLLP [273]	LLaMa-2-7B as planner and reflector
	ONCE [191]	LLaMa-2-7B as Content Encoder
SLM in Web Search	RecLoRA [396]	personalized low-rank adaptation
	Content encoder [38, 136, 204]	Encode concatenated queries and documents and integrates correlations at character, word, and phrase levels post-output layer.
	Ranker [56, 231]	Introduce a retrieval framework that identifies candidates, which are subsequently re-ranked using a specially fine-tuned T5
SLM in Mobile-device	Rewriter [208]	Utilize the "rewrite-retrieve-read" framework to bridge the gap between queries and needed knowledge by rewriting inputs.
	Octopus [44]	Calling software APIs via learning in documents
	MobileAgent [80]	Standard Operating Procedure (SOP)
	Revolutionizing Mobile Interaction [37]	Text-to-action control and tests on 6GB Android devices and 4GB devices
	AutoDroid [333]	Interaction based on GUI and APP knowledge injection
	On-device Agent for Text Rewriting [399]	Data Knowledge Distillation from LLMs

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 small language models (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' impressive question-answering capabilities, the cost of training with over 3.4T tokens on 512 H100 GPUs for 10 days [1] poses a challenge for many researchers and developers. Instruction tuning [329] offers a cost-effective alternative fine-tuning method, enhancing small models via fine-tuning on large foundation models' outputs. Alpaca 7B [292] tunes Llama 7B [301] using 52k ChatGPT-generated

examples from 175 seed tasks. In blind tests, the performance of Alpaca 7B matches the LLM text-davinci-003 [33], winning 90 of 179 comparisons. This behavior cloning is effective in mimicking the style of teacher models. However, this approach may not improve small model performance for reasoning-intensive QA tasks where accuracy, not style, is crucial [52]. To counter this, Stable Beluga 7B [212] employs explanation tuning, refining Llama-2 7B [302] on an Orca-style dataset with explanatory LLM answers to enhance reasoning capabilities. Explanation Tuning extracts detailed answers from LLMs using system instructions, but effectiveness varies with system instructions, and instructions that work for larger models like GPT-4 may not suit smaller models. Similarly, SLMs face challenges in identifying optimal system instructions in various tasks for training that differ from LLMs'. Therefore, Orca 2 [219] proposes a cautious reasoning SLM training process: (1) starting with diverse tasks, (2) guided by the performance of Orca 1 [223] to select system instructions, (3) writing task-specific instructions, and (4) employing Prompt Erasing to replace detailed instructions with generic ones, encouraging models to help SLMs learn not just task solutions but also deeper reasoning abilities. Orca 2 introduces a new dataset with approximately 817K instances, training on Llama-2-7B or Llama-2-13B [302] with 32 NVIDIA A100 GPUs over about 70 hours, demonstrating efficient instruction tuning compared to training from scratch. Evaluation on the ARC-Challenge shows Orac-2-7B achieves 78.41 accuracy, outperforming Llama-2-Chat-70B (67.66) and close to ChatGPT (84.73), showing that instruction tuning in Orca 2 can rival models 5-10 times its size.

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 the financial domain, Phogat et al. [244] transfer financial QA abilities from generic teacher LLMs such as GPT-4 [2] to specialized student SLMs, such as Phi-3-Mini [1], using datasets including FinQA [48], ConvFinQA [49], and TATQA [395]. They apply step-wise arithmetic annotation via a program of thought (PoT) [45] prompting, preparing few-shot examples that cover concept understanding, formula writing, extracting relevant entities, and performing calculations. Incorrect codes and formats in teacher annotations are removed before fine-tuning SLMs with LoRA [127]. Tests on FinQA show Phi-Mini closely matches GPT-4's accuracy (77.59 vs. 77.51 zero-shot and 78.46 few-shot), improving fine-grained financial QA abilities, especially in concept understanding and entity extraction. (2) In the medical field, Guo et al. [112] enhance student SLMs, including domain-specific BioGPT (1.6B) [205] and general Llama 7B [301], by fine-tuning on enriched PubMedQA [146] 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. Additionally, there are three interesting insights: (i) Direct tests on PubMedQA reveal that BioGPT (1.6B) performs comparably or worse than the larger Llama 7B (0.594 vs 0.63 accuracy, 0.495 vs 0.387 macro-F1). (ii) When fine-tuned on the same PubMedQA training data, BioGPT (1.6B) notably outperforms Llama 7B, scoring 0.498 to Llama 7B's 0.463 macro-F1, highlighting that domain knowledge helps domain SLMs learn faster than general models. (iii) Stronger teacher LLMs improve fine-tuning outcomes. For example, BioGPT (1.6B) fine-tuned with new QA data from ChatGPT and GPT-4 shows that the latter setup, with a macro-F1 of 0.520, outperforms the former's 0.498, indicating that more knowledgeable teachers significantly enhance SLM specialization. We report the detailed results of comparisons of instruction-tuned domain-specific language models (SLMs) for QA and larger language models on FinQA [48] and PubMedQA [146], as shown in Table 5.

Table 5. Comparison of instruction-tuned domain SLMs for QA and LLMs on FinQA [48] and PubMedQA [146].

Model	Size	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 [47]	70B	PubMedQA	Zero-shot	81.6
RankRAG-llama3-70B [367]	70B	PubMedQA	Zero-shot	79.8
Flan-PaLM [277]	540B	PubMedQA	Few-shot	79.0
GAL 120B [293]	120B	PubMedQA	Zero-shot	77.6
Flan-PaLM [277]	62B	PubMedQA	Few-shot	77.2
BioGPT [205]	345M	PubMedQA	Zero-shot	78.2
BioGPT-Large [205]	1.5B	PubMedQA	Zero-shot	81.0

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 Chain-of-Thought (CoT) prompting, Retrieval-Augmented Generation (RAG), and self-adaptive techniques. We categorize these techniques into the following groups:

- (1) **Retrieval-Augmented Generation (RAG):** *Incorporating External Knowledge for Domain-Specific QA* Retrieval-augmented generation (RAG) addresses out-of-domain (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 small language models to provide accurate answers on specialized topics. In the telecommunications domain, Gichamba et al. [100] 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. Hartill et al. [115] propose an approach combining multi-task pre-training with dense retrieval for compositional questions. The model decomposes complex, multi-hop queries and retrieves evidence from external sources, showing improved performance on benchmarks like CommonsenseQA [288] and ARC-DA [58]. EffiChainQA [259] introduces a chain-of-reasoning framework that enhances reasoning in open-domain QA through retrieval and question decomposition. By breaking down queries and retrieving documents using a retriever fine-tuned on HotpotQA [360], EffiChainQA achieves performance comparable to state-of-the-art models on datasets like HotpotQA, demonstrating the efficacy of SLMs in managing complex tasks with lower computational costs.
- (2) **Self-Adaptive Techniques:** *Enhancing Model Adaptability with Self-Generated Pseudo Labels* Despite LMs' capacity to store extensive general knowledge applicable across various tasks, they often exhibit suboptimal performance when transferring and adapting this knowledge to specific downstream tasks. Fine-tuning, while effective, 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 [276, 309]. Jeong et al. [139] introduce Test-time Self-Adaptive Small LMs (**T-SAS**), a framework for SLMs that first stochastically generates multiple answers for an unlabeled question. The most plausible answer is then selected via majority voting to enhance pseudo-label accuracy, and samples with low agreement are filtered out. Evaluated on QA datasets like Natural Questions [159], TriviaQA [147], and SQuAD [253], T-SAS shows improved model adaptability. For example, on the SQuAD dataset, Flan-T5 base (250M) [57] attains a 63.02 EM score, nearly matching the 69.94 score achieved by the fine-tuned Flan-T5 base with a training set.

(3) **Chain-of-Thought (CoT) Prompting:** *Generating Intermediate Reasoning Steps* Chain-of-thought (CoT) prompting enhances small language models' ability to handle out-of-domain (OOD) questions by encouraging them to generate intermediate reasoning steps. By articulating step-by-step thinking, CoT enables models to navigate unfamiliar topics more effectively and arrive at accurate answers. Rationale Ranking [114] propose to use an SLM that generates intermediate reasoning steps, or rationales, to provide answers grounded in evidence. By generating and ranking these rationales, the SLM supports its answers with a clear chain of reasoning, reducing the risk of hallucination when faced with unfamiliar or out-of-domain questions. Rationale ranking further enhances this process by scoring the generated rationales based on relevance and truthfulness. This ranking allows the model to prioritize the best rationale and improve accuracy. Together, these techniques ensure the model delivers reliable answers by systematically selecting the most appropriate reasoning pathway, even when encountering unfamiliar queries.

Comparison between LLMs and SLMs for QA. When comparing LLMs like GPT-4 [2] or BLOOM-175B [162] 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 trade-off is between scale and specialization: LLMs handle diverse domains but may need additional techniques like Retrieval-Augmented Generation (RAG) 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, 138, 178] showcase SLMs' evolution in coding tasks. For instance, Phi-1 [107], a Transformer with 1.3 billion parameters, specializes in basic Python coding and achieves notable scores in benchmarks such as HumanEval [107], 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 3.8 billion 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 [84] and surpassing smaller ones like Gemma-2 [296].

Another avenue of development is the fine-tuning of general-purpose SLMs for coding tasks [21, 108, 203, 260, 294]. For instance, CodeLlama models [260], derivatives of Llama 2 [302], 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 [260]. CodeGemma models [294], 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 [294].

Comparison between SLMs and LLMs on Coding. Table 6 provides a comparative analysis of SLMs and LLMs on coding benchmarks HumanEval [42] and MBPP [18]. Insights include: (i) Small SLMs (1.3B - 3.8B Parameters) like Phi-3.5-mini [305] achieve high scores, demonstrating the efficacy of small models. Mid-sized SLMs (6.7B - 9B Parameters), such as DeepSeek-Coder 6.7B [108] and Llama 3.1 8B [84], 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 [84], GPT-4o [234], and Claude 3.5 Sonnet [14] excel, supporting the idea that bigger models generalize better across diverse coding tasks; (ii) There's a notable trade-off between computational efficiency and performance, with larger models requiring more resources, impacting their practical deployment in constrained environments; (iii) Specialized training and fine-tuning, as used in models like DeepSeek-Coder [108], 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. All models listed are chat or instruct versions, and performance are sourced from respective research papers or technical reports [84, 108, 260, 294, 305].

Model	Size	HumanEval	MBPP
DeepSeek-Coder [108]	1.3B	65.2	49.4
CodeGemma [294]	2B	37.8	49.2
Gemma 2 [296]	2B	17.7	40.2
Phi-3.5-mini [305]	3.8B	62.8	69.6
DeepSeek-Coder [108]	6.7B	78.6	65.4
CodeGemma [294]	7B	60.4	55.2
Llama 3.1 [84]	8B	66.5	69.4
Gemma 2 [296]	9B	61.0	69.3
GPT-3.5 Turbo	-	68.0	71.2
DeepSeek-Coder [108]	33B	79.3	70.0
Llama 3.1 [84]	70B	80.5	75.4
Llama 3.1 [84]	405B	89.0	78.8
GPT-4o OpenAI [234]	-	90.2	81.4
Claude 3.5 Sonnet [14]	-	92.0	76.6

4.1.3 SLM Applications in Recommender Systems. Recommender systems are essential in various online services, helping to manage information overload and meet users' 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 recommendation. Next, we introduce the details.

SLM for System Cold Start Problem. Traditional recommendation systems, which utilize historical user-item interactions like 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 [256]. Although LLMs address this with in-context learning, their slow and costly inference restricts real-time use. Thus, **PromptRec** [338] 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 measured by mutual information between documents and user-item interactions, using a C4 corpus subset [250], and by developing training prompts for different domains, enhancing cold-start performance. Results show that enhanced SLMs like BERT-mini [77], 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 [119, 232, 384, 400]. For example, ADLRS [119] employs BERT to convert web-crawled item profiles into vectors that highlight key aspects, aiding recommender systems in acquiring essential initial information.

Recent studies have employed LLMs to address the cold-start problem. For instance, Sanner et al. [263] uses PaLM 62B [55] with prompts to infer preferences for cold users based on their textual profiles, **LLM Interaction Simulator (LLM-InS)** [130] uses Llama 2 (size unspecified) to simulate interactions and generate user interaction sets for cold items, enabling simultaneous training of cold and warm items to bridge their embedding gap, and Wang et al. [317] applies PaLM to determine user preferences for cold-start items from textual descriptions of users' historical behaviors and new item descriptions. Despite the limited exploration of SLMs in the LLM era for recommendation cold-start problems, researching them is worthwhile due to their enhanced in-context learning capabilities and small size.

SLM for Mitigating Popularity Bias. Popularity bias in recommender systems, characterized by a discrepancy between item popularity in training datasets and the open world, often arises from training on closed-loop datasets that contain narrow and fixed information. Recent LLMs utilize their expansive open-world knowledge to enhance reasoning about user-item interactions [183, 191], thereby mitigating this bias by enriching the inputs to recommenders with more comprehensive item information. However, the substantial resource demands of LLMs limit their practical application. To address this challenge, the Step-by-step Knowledge Distillation Framework for Recommendation (**SLIM**) [327] distills the reasoning capabilities of LLMs into smaller language models (SLMs), retaining only 4% of the original parameters—specifically transitioning from ChatGPT to Llama 7B [301]. SLIM employs streamlined templates to facilitate this transfer, enabling SLMs to bolster recommender systems by incorporating additional item information. Evaluations across three Amazon review categories—games, food, and home—demonstrate that SLIM not only maintains high recommendation quality but also effectively reduces popularity bias, all while significantly lowering operational costs.

SLM for Long-term Planning. Traditional recommendation systems primarily focus on optimizing users' immediate responses, often maximizing short-term benefits but neglecting long-term engagement, leading to issues like confining users within an echo chamber of preferred information and filter bubbles [96, 326]. To address this, it is crucial to integrate planning abilities into recommendations to consider both immediate and long-term outcomes. Language models, with their world knowledge and reasoning abilities, are anticipated to offer powerful planning capabilities. **BiLLP** [273] introduces a hierarchical learning approach with macro-learning and micro-learning phases. Macro-learning involves a Planner and a Reflector, both implemented as SLM instances such as Llama-2-7B [302]. The Planner uses high-level experiences to create long-term plans, while the Reflector updates the Planner by reflecting on past actions. Micro-learning employs an SLM-based Actor-Critic mechanism for personalized planning, where the Actor converts plans into actions and the Critic evaluates the actions for long-term benefits. Similar to SLMs for cold-start problems, the use of SLMs for long-term planning is also underexplored and merits further investigation.

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 [127] 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** [396] utilizes Vicuna-7B [53] to integrate personalized knowledge into SLMs/LLMs tailored for recommendation tasks, as illustrated in Figure 12. Specifically, RecLoRA maintains a set of parallel, independent LoRA weights (A_i, B_i), allowing for the customization of language model parameters to match individual user preferences more effectively.

SLM as 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 systems now employ language models fine-tuned on specific datasets as text encoders. For example, Wu et al. [335] conducts pioneering work using a pre-trained language model to enhance large-scale news recommender systems by substituting traditional news encoders with a BERT model [77]. While models such as BERT [77] may struggle to capture essential content when pre-trained on limited data, more versatile LMs such as Llama-2-7B [302] enhance text descriptions and demonstrate improvements in recommendation

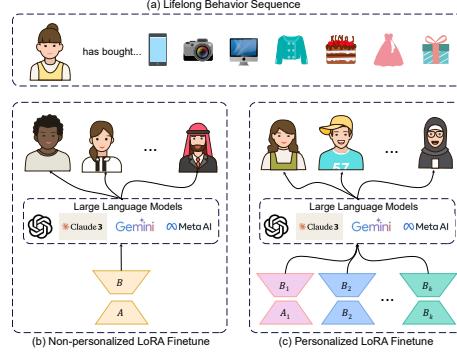


Fig. 12. The illustration of lifelong behavior sequence and personalized low-rank adaption (LoRA) for recommendation [396].

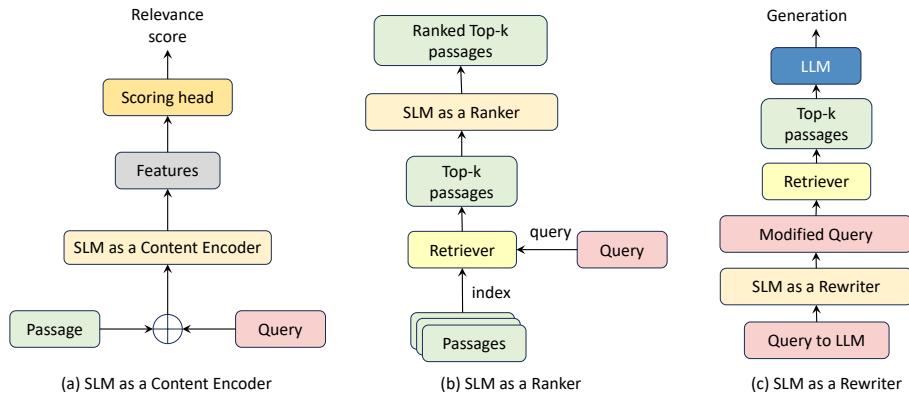


Fig. 13. Roles of SLM in Web Search.

performance through advanced embedding techniques. **ONCE** [191] propose using Llama-2-7B [302] as an encoder to overcome the limitations of BERT in content-based recommendations. Additionally, the study explores the synergistic use of LLMs such as ChatGPT in recommendation systems, finding that SLMs optimized with LoRA [127] 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 deep language understanding from pre-training [56, 82, 318]. 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: *content encoder* [38, 136, 204], *ranker* [56, 231], and *rewriter* [208], as depicted in Figure 13. Next, we give details.

SLM as a Content Encoder. Text embeddings are vector representations of natural language that encode semantic information, widely used in retrieval. SLM-based embedding retrieval, also known as dense retrieval, leverages deep language understanding from pre-training to effectively address semantic challenges. **H-ERNIE** [56] features a hierarchical model structure that encodes both the query and the document at various levels of granularity, such as character, word, and phrase. This design addresses the ambiguous model understanding in web searches, exemplified by a search query for "red panda" returning results related to "panda." The model includes an aggregation module that collects information from finer-grained layers to construct the coarser-grained layers, enhancing the specificity and relevance of search results. **Implicit Interaction (I^3)** [82] uses BERT [77] as a content encoder, generating implicit pseudo-queries from passages to enable high online efficiency with offline caching of passage vectors. Similarly, Zou et al. [402] employs BERT and ERNIE as encoders, adapting them for integration into retrieval systems through techniques like fine-tuning for doc-query similarity tasks. To improve online service responsiveness, they suggest using knowledge distillation to create lightweight service ranking models that preserve effectiveness while meeting latency demands.

However, ERNIE and BERT-style models overlook advancements in SLMs like context length extension. Thus, Wang et al. [318] employs the improved Mistral-7B [143] and fine-tunes it on synthetic data across 93 languages generated by GPT-4 [2] to enhance embeddings. These enhanced embeddings show competitive performance on the BEIR [298]

and MTEB [222] benchmarks, improving by a notable margin (+2%). They also effectively handle personalized passkey retrieval for inputs up to 32k tokens by modifying the rotation base of position embeddings, extending the context well beyond the usual 512-token limit. Peng et al. [241] employs LLaMa-7B [301] and Vicuna-7B [53] as semantic encoders for embedding retrieval, demonstrating improved performance through soft prompt tuning. **CoCondenser** [98] 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. CoCondenser applies an unsupervised contrastive loss, minimizing data manipulation and batch size needs. Tests on MS-MARCO [22], Natural Questions [159], and Trivia QA [147] demonstrate its robustness and efficacy.

SLM as a Ranker. The reranking task, the following retrieval, improves the order of multiple candidates to enhance retrieval quality because rerankers are more accurate than embedding retrievers. **InPars (Inquisitive Parrots for Search)** [32] introduces a retrieval framework that employs the T5 base 220M [250] as a re-ranker to enhance the BM25 retriever [258]. Initially, BM25 selects 1,000 candidates, which are then re-ranked using a fine-tuned T5 model, called monoT5. This T5 model is adapted as a binary classifier to predict the relevance of a document to a query. The training data, generated by GPT-3 [33], creates queries for specific documents and selects negative examples randomly. Experimental results demonstrate that this specifically fine-tuned SLM-enhanced retriever significantly outperforms the version utilizing GPT-3 [33]. For instance, performance on the TREC-DL 2020 dataset [63] shows that retrieval with monoT5-220M achieves a 0.3599 MAP score, surpassing that of GPT-3 (175B), which scores 0.3163.

SLM as a Rewriter. The query to the retriever, often in the form of several keywords, may expose a knowledge gap between the actual query and the knowledge required for effective querying, limiting retrieval performance. For instance, in the retrieval-augmented generation, this gap places a burden on prompt engineering to the downstream LLMs. To address this, the "**rewrite-retrieve-read**" framework [208] employs T5-large [250] as a rewriter to bridge the gap between queries and necessary knowledge by rewriting inputs. A specialized, trainable language model, known as the "rewriter," is used for this task. It is trained via reinforcement learning, with downstream LLM performance as a reward, to better adapt queries for downstream tasks. Experimental results show that this trainable SLM rewriter exceeds the performance of general LLM rewrites. For example, on HotpotQA, the rewrite-retrieve-read framework achieves a 45.97 F1 score, 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 has raised privacy concerns, with their large size limiting practicality on mobile devices and hindering real-time responses in urgent scenarios such as medical emergencies. To address these limitations, researchers are developing smaller, domain-specific models that deliver accurate results, making SLMs a viable alternative. This subsection introduces SLM applications on mobile devices, categorizing works into three aspects: software API calls, mobile control, and basic NLP applications on devices.

SLM for Software API Call. Integrating LLMs with external APIs enhances their capabilities by accessing up-to-date information and specialized functionalities, but fine-tuning them for API calls incurs significant training costs. Consequently, there is a trend toward developing smaller, task-specific language models that preserve essential functionality while reducing training costs. Nonetheless, these smaller models pose increased risks of errors and precision issues, which are critical for solving various tasks. In response, **Octopus** [44] crafts a diverse dataset from over 30,000 widely-used APIs and implements curriculum learning strategies [194] to enhance accuracy in selecting the appropriate API functions. This approach has been applied in models such as Codellama-7b [260] and Google's Gemma series [295], significantly improving API call performance. For instance, Octopus-gemma2B achieves a function accuracy of 93% compared to ChatGPT's 50% and closely approaches GPT-4's 96% on their crafted dataset.

SLM for Mobile Control. LLM 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, LLMs offer scalable automation via GUI-based text contents. **MobileAgent** [80] integrates instructions and Standard Operating Procedures (SOP) to enhance SLM performance in mobile control applications. An example, as shown in Figure 14, begins with the AI agent receiving a goal, such as booking a dental appointment. It analyzes the mobile screen and query, processes prior actions and environmental factors like UI elements, and forms a prompt. The agent then generates an output—choosing an appointment—and executes actions by identifying relevant UI elements on the mobile interface, such as text and XPath. They fine-tune the Qwen-7B model [21] on a dataset from the AIA medical application, evaluating it on the AitW benchmark [257], where it demonstrates superior performance to GPT-4 [2] without additional inference costs. AitW is recognized for its relevance in mobile operations. Carreira et al. [37] addresses privacy and latency concerns of LLMs by running a small model offline on mobile devices. This model, fine-tuned with data generated by ChatGPT-3.5, achieves text-to-action capabilities for tasks such as making calls and conducting web searches. The RedPajama-INCITE-Chat-3B-v1¹ model is chosen for its size and chatting capabilities, employing native code and model quantization techniques. Despite hardware constraints, it delivers adequate performance on 6GB Android devices and operates on 4GB devices.

AutoDroid [333] is developed to enhance Android app interactions through a GUI. Figure 15 shows an example of LLM-powered mobile task automation [37] that a user asking to be reminded about doing laundry on Aug 17. The process involves four steps: (1) clicking 'New Event', (2) entering 'laundry' in the 'Title' field, (3) clicking 'Save', and (4) completing the task. The agent uses the phone GUI, guided by Vicuna-7B and the app's domain knowledge, to complete the task. AutoDroid dynamically integrates SLM-derived commonsense and app-specific knowledge, using it to generate privacy-filtered prompts that drive app interactions based on user commands. Its effectiveness is validated on the proposed DroidTask benchmark, showcasing superior performance over GPT-3.5 and GPT-4 [2]. For example, their Vicuna-7B AutoDroid achieves 57.7%, outperforming ChatGPT's 34.7% and GPT-4's 54.5% [2]. These studies illustrate the potential of customizing smaller, domain-specific SLMs in mobile environments, addressing traditional memory limitations while maintaining essential functionalities.

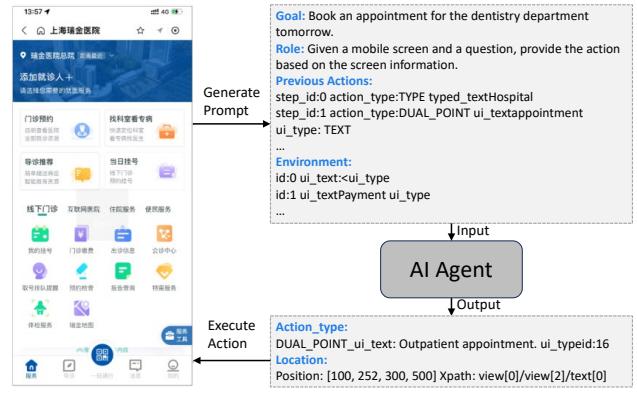


Fig. 14. An example workflow for an automated execution tool [80]. The screenshot in the left is taken from [80].



Fig. 15. An illustration of Vicuna-7B-powered mobile task automation [37] shows a user asking to be reminded about doing laundry on Aug 17. The figure is taken from [37].

¹<https://huggingface.co/togethercomputer/RedPajama-INCITE-Chat-3B-v1>

SLM for Basic NLP Applications on Devices To enable personalization on mobile devices while ensuring privacy, performing basic NLP tasks such as text rewriting directly on the device is essential, avoiding data transfer to the cloud. Qin et al. [247] introduces a framework that utilizes self-supervised data selection and synthesis for on-device fine-tuning, leveraging sparse annotations and limited storage effectively. This approach, demonstrated in Figure 16, employs the Llama-3B model [301] and the LoRA fine-tuning method [127], enhancing personalization by efficiently managing data through metrics like embedding entropy and domain-specific scores. In addressing mobile text rewriting, Zhu et al. [399] train the compact Palm 2-XXS model [13] using data generated by the larger Palm 2-L to ensure user privacy and accommodate device constraints. Their new evaluation benchmark, MESSAGEREWRITEVAL, demonstrates that this mobile model surpasses the performance of traditional LLMs such as LLaMA [301] and Alpaca-7B [292]. For example, the BLEU score of the proposed SFT Palm 2-XXS is 34.59, outperforming the 16.65 achieved by LLaMa 7B [301]. Performance benchmarks on devices like the Samsung S23 and Pixel 7 Pro show significantly lower latency (36.2ms and 59.8ms, respectively) compared to tests conducted on a MacBook M1 Pro with a 4-bit quantized Llama 7B model (18-22 tokens/second), validating the effectiveness and practicality of this mobile-centric approach in text rewriting tasks.

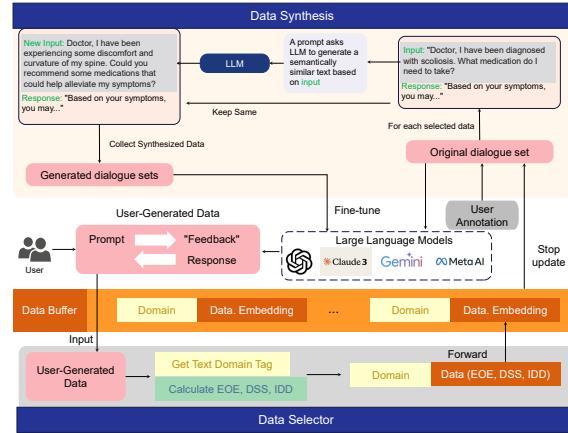


Fig. 16. Overview of the framework for fine-tuning LLMs using synthesized and user-generated data [247]. Fine-tune LLMs using data from data selection and following data generation. The process involves Data Synthesis, where prompts are used to generate semantically similar text, creating new dialogue sets. Data Selection follows, where user-generated data is processed, and tagged with domain labels, and metrics such as EOE, DSS, and IDD are calculated. The selected data is used to fine-tune the LLM, with user annotations guiding the process. The framework continuously refines LLMs based on this iterative data generation and selection.

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 search such as document encoding, text reordering, and query rewriting, often outperforming LLMs through techniques such as supervised fine-tuning, 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.

Table 7. On-device Deployment Optimization Techniques

Aspect	Representative Work	Key Point
Memory Efficiency Optimization	EDGE-LLM [368]	Edge LLMs use LUC and adaptive tuning for efficiency
	LLM-PQ [390]	Optimize quantization and layer partitioning for complex setups.
	AWQ [184]	Preserve key weights based on activation distribution, not weight distribution
	MobileAI Bench [225]	Evaluation
	MobileLLM [199]	Evaluation
	EdgeMoE [364]	Load experts on activation, tripling memory savings.
	GEAR [148]	Enhance KV cache quantization by integrating error-reduction techniques.
	DMC [228]	Adaptively compress KV cache, optimizing storage efficiency.
Runtime Efficiency Optimization	Transformer-Lite [171]	Optimize KV cache to reduce redundancy and memory use.
	LLMaaS [365]	LLMaaS manages apps via chunk-wise KV cache optimization on mobiles.
	EdgeMoE [364]	Predict expert needs, boosting inference speed and reducing latency.
Efficiency Optimization	LLMCad [348]	Use SLM for fast token generation and cloud verification.
	LinguaLinked [389]	Optimize data flow and load, enhancing multi-threading efficiency.

4.2 SLM Deployment on Mobile and Edge Devices

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 range 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 [184, 251, 368, 390], the cache of MoE experts [364], and KV cache compression [148].

Quantization. Quantization, a prevalent and efficient approach in designing and deploying SLMs, reduces the model's numerical precision to significantly save memory while maintaining accuracy. We outline quantization strategies in Sections 2.3.3 and 3.5. Here we focus on representative quantization works related to edge devices and their evaluations of memory usage. Regarding quantization works related to edge devices, the **EDGE-LLM** framework [368] 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 Adaptive Layer Tuning and Voting scheme to optimize memory

use while ensuring performance. Meanwhile, **LLM-PQ** [390] 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)** [184] is a hardware-friendly, low-bit, weight-only quantization method for on-device LLMs that preserves key weights based on activation distribution, not weight distribution, to minimize quantization loss. AWQ uses per-channel scaling to optimize quantization error reduction without relying on backpropagation, thus maintaining LLMs' generalization across different domains and modalities without overfitting.

Regarding the quantization evaluations of memory usage, MobileAIBench [225] evaluates the impact of different quantization levels, including 3-bit, 4-bit, 8-bit, and 16-bit, on large language and multimodal models on iPhone 14. Firstly, it shows that quantization can significantly reduce disk usage while maintaining performance, e.g., Llama 2 7B's disk usage decreases from 13 GB at 16-bit to 3.6 GB at 4-bit, while the F1 score in HotpotQA [360] only drops from 0.21 to 0.198. Secondly, while 4-bit quantization generally preserves effective performance, reducing to 3-bit significantly impairs model accuracy. For instance, Mobile-VLM-1.7B [337] shows a marked decrease in accuracy on the VQA-v2 [103] benchmark, dropping from 0.622 at 16-bit to 0.607 at 4-bit, and plummeting to 0.076 at 3-bit. This highlights that while more aggressive quantization levels can yield greater memory savings, they may drastically reduce model accuracy. Rahman et al. [251] evaluates MobileBERT [286] on Raspberry Pi 3B edge devices using TensorFlow-Lite at 8-bit, 16-bit, and 32-bit quantizations. The evaluation uses crafted tweet data and a reputation polarity classification task. Results show 32-bit consumes 74.9 W/sample with 1.66s latency, while 8-bit reduces consumption to 47.49 W/sample and latency to 1.06s. Accuracy is nearly identical, at 0.685 for 32-bit and 0.684 for 8-bit. Memory usage is 64.1% for 32-bit and 66.1% for 8-bit. Quantization significantly reduces latency and energy consumption by approximately 1.5 times while maintaining performance levels. However, memory usage is not optimized due to the need for dequantization during inference. MobileLLM [199] applies quantization to its MobileLLM and MobileLLM-LS models, which have 125M and 350M parameters respectively, resulting in a modest accuracy reduction of less than 0.5 points.

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 [137] 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 [89], with 32 experts per layer, require 54GBs of memory for inference, exceeding the capacity of most edge devices. Yi et al. [364] notes 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** [364] introduces a method where experts are loaded into an expert memory buffer only when activated, achieving approximately 3 \times 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 [245]. 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 or storage [11, 272], although this can introduce significant overhead due to the switching costs between GPUs and CPUs. An alternative approach, token dropping, compresses the cache size by retaining only the most important tokens, based on the commonly low attention scores [99, 197, 387]. Yet, such methods struggle with complex tasks requiring long responses or chain-of-thought (CoT) reasoning, as performance declines at high compression levels due to increased estimation errors—the disparity between the original and compressed KV values. To address these issues,

GEAR [148] enhances KV cache quantization by integrating error-reduction techniques. This approach includes: (i) quantizing the majority of caches of similar magnitudes to ultra-low precision, (ii) using a low-rank matrix to efficiently approximate the quantization residuals, and (iii) employing a sparse matrix for a negligible ratio of entries of large magnitudes to correct individual errors from outliers. This composite approximation strategy separates the coherent from the incoherent parts of the approximation error: the low-rank matrix captures the majority of the coherent basis of the quantization error, while the sparse matrix addresses the inconsistencies present in individual outliers. This optimization enables near-lossless KV cache compression, achieving a peak memory reduction of up to 2.29 \times . **Dynamic Memory Compression (DMC)** [228] adaptively compresses the KV cache by choosing either to add the current key and value representations directly to the cache or to blend them with the top item in the cache using a weighted average. Evaluations on Llama 2 [302] models with 7B, 13B, and 70B parameters demonstrate that DMC enables LLMs to maintain downstream performance on benchmarks such as MMLU [120] and HumanEval [42] close to that of the original LLM. For instance, DMC Llama 2 7B with a 4 \times compression ratio achieves an MMLU accuracy of 43.9, nearly matching the original model’s 44.6. Besides, **Transformer-Lite** [171] addresses a specific issue where model inputs and outputs redundantly store the KV cache twice, significantly increasing memory usage despite the content being largely identical. To optimize this, they refine KV cache storage by allocating a suitably large tensor for each input KV cache based on the maximum sequence length required for the inference task. Sub-tensors are then generated from this main tensor at varying address offsets, serving as model input and output KV caches. This approach allows the new KV caches to be written directly to the correct locations during model inference, thereby eliminating the need for additional copying processes. **LLMaaS** [365] introduces the concept of a Large Language Model as a Service on mobile devices, enabling LLMs to manage all device applications. LLMaaS proposes LLMS, which uses chunk-wise KV cache compression and swapping to facilitate context switching across apps within a limited memory budget. By dividing the KV cache into independently compressed and swapped chunks, LLMS optimally balances device memory use and I/O bandwidth, outperforming token-level or context-level management.

4.2.2 Runtime Efficiency Optimization. Runtime efficiency involves minimizing inference latency by reducing computing workload, decreasing the transfer of weights between GPU and CPU memory in MoE, and distributing LLM weights across multiple trusted devices.

Reducing Computing Workload. The goal of decreasing computing workload aligns with enhancing memory efficiency through methods such as quantization and KV cache compression, as mentioned in the previous section. Less model weight precision or fewer model weights naturally reduce latency. For example, quantization on an RP 3B device can achieve an inference time of 1.06 seconds per sample at 8-bit, compared to 1.66 seconds per sample at 32-bit [251]. Additionally, DMC demonstrates that 4 \times compression can result in a 3.4 \times acceleration on Llama 2 7B, and on an A100, the generation of 1K tokens with 3K tokens of context [228].

Beyond quantization and KV cache compression, model collaboration, deploying smaller SLMs on devices with cloud-based LLM support, enhances runtime efficiency. Large LLMs will increase latency when directly deployed via mobile engines like llama.cpp due to a large number of computing operations. **LLMCad** [348] addresses this by using a real-time, memory-resident SLM for simple tokens such as determiners and punctuation. This SLM generates tokens, while a larger cloud-based LLM verifies and corrects them, speeding up the process. LLMcad boosts token generation by up to 9.3 \times , for instance, by utilizing Llama 2 7B as the memory-resident LLM and Llama 2 13B as the cloud-based LLM, reducing latency from 16.2 seconds to 3.9 seconds on the Xiaomi Pro during TruthfulQA [185] tasks.

Reducing MoE Switching Time. To reduce latency in MoE architectures caused by frequently switching experts in limited device memory, EdgeMoE [364] 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 $1.11\times$ to $2.78\times$ and significantly reducing latency. For instance, in a switch transformer model with 8 experts, latency drops from approximately 0.7s to 0.3s, 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 [389] 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 $1.73\times$ and $2.65\times$ 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 [184, 199, 225, 251, 368, 390]. Similarly, KV cache compression also helps achieve these efficiency gains [148, 171, 228, 365].
- 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 [364].
- 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 [348].
- 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 [389].

5 GENERIC, TASK-SPECIFIC, AND DOMAIN-SPECIFIC SMALL LANGUAGE MODELS

This section investigates small language models (with fewer than 7 billion parameters) in both general and specific domains. It details the methods of obtaining these small language models, the datasets, and the evaluation tasks,

Llama 3.2 1B Model			
Model Specs		Dataset Specs	
Developer	: Meta AI	Open Training Dataset?	: No
Params	: 1.23B	Training Data	: Unk
Release Date	: 9/25/2024	Training Data (tokens)	: 9T
License	: Llama 3.2 Community License	Data Freshness	: December 2023
Intended Use	: Commercial and Research use	Training Specs	
Language	: EN, DE, FR, IT, PT, HI, ES, TH	Pre-text tasks	: Knowledge Distillation
Input Modality	: Text	Pre-training methods:	: Pruning-based initialization KD from Llama 3.1 8B & 70B
Architecture	: Transformer	Hardware	: H100-80GB
Layer Number	: 16	GPU Hours	: 370K
Hidden Size	: 2048	Post-training methods:	: SFT, RS, DPO
Attention	: GQA	Evaluation Specs	
Attention Heads	: 32	Benchmark	: MMLU (English)
Activation	: SiLU	Motivation	: Test Multi-subject Knowledge
Architectural Tech	: Shared Embeddings	Metric	: Macro Avg
Tokenizer	: TikToken-based	Performance	: 32.2
Context length	: 128K	Responsibility	
Vocab Size	: 128256	Safety	: Red Teaming, Llama safeguards
Open Source?	: Yes	Evaluations	: Red Teaming, CBRNE, Child Safety, Cyber Attacks
Code	: https://huggingface.co/meta-llama/Llama-3.2-3B		

Fig. 17. Llama 3.2 1B model card

exploring the techniques for acquiring SLMs through compression, fine-tuning, or training from scratch. Additionally, we summarize the representative small language models, as detailed in Table 8 and 11.

5.1 Generic-domain SLMs

Overview. “The bigger, the better” dominates LLM development, yet these models struggle with on-device processing and energy efficiency, which are vital for privacy, customization, and sustainability. Small language models (SLMs), with fewer parameters, improve computational efficiency in pre-training, fine-tuning, and inference, reducing memory and energy demands—crucial in resource-limited settings. The localized and compact characteristics of SLMs enhance privacy, personalization, and response times, making them suitable for low-power edge devices. Therefore, SLMs are attracting increasing attention, and various models are being developed. Table 8 summarizes current representative generic-domain 40 SLMs/SLM families. Although all chosen SLMs have similar architectures, they 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 17 as an example, its parameter size and use of filtered high-quality training data, pruning-based initialization, knowledge distillation pre-training tasks, and training techniques such as Supervised Fine-Tuning (SFT), Rejection Sampling (RS), and Direct Preference Optimization (DPO) distinguish it from others.

From Table 8, we observe several trends in component choices for SLMs: (i) Recent SLMs frequently employ Grouped Query Attention (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. (ii) SiLU and SwiGLU are preferred over ReLU as activation functions due to their superior capabilities. SwiGLU’s parameters are learned during training, allowing the model to dynamically adapt to various tasks and datasets, enhancing its flexibility and establishing it as a state-of-the-art choice. Conversely, SiLU is favored in small language models mainly for its computational efficiency, as it introduces no extra parameters. (iii) RMS normalization is more commonly used than

Table 8. High-level Overview and Training Details of Generic-domain Small Language Models. #Params means Parameter amounts.

Model	#Params	Date	Paradigm	Domain	Training Datasets	Training Techniques
Llama 3.2 ²	1B; 3B	2024.9	Pre-train	Generic	no release (9T tokens)	GQA, SiLU, Multilingual Text and code, Shared embedding, Pruning, Distillation, SFT, RLFH, RS, DPO
Qwen 1 [21]	1.8B; 7B; 14B; 72B	2023.12	Pre-train	Generic	no release	MHA; RoPE; SwiGLU; RMSNorm
Qwen 1.5 [21]	0.5B; 1.8B; 4B; 7B; 14B; 32B; 72B	2024.2	Pre-train	Generic	no release	MHA; RoPE; SwiGLU; RMSNorm; Multilingual support
Qwen 2 [352]	0.5B; 1.5B; 7B; 57B; 72B	2024.6	Pre-train	Generic	no release	GQA; RoPE; SwiGLU; RMSNorm; Multilingual support
Qwen 2.5 [352]	0.5B; 1.5B; 3B; 7B; 14B; 32B; 72B	2024.9	Pre-train	Generic	no release	GQA; RoPE; SwiGLU; RMSNorm; Multilingual support; Larger corpus
Gemma [295]	2B; 7B	2024.2	Pre-train	Generic	Unknown	MHA, RoPE, GELU _{tanh}
Gemma 2 [296]	2B; 9B; 27B	2024.7	Pre-train	Generic	Unknown	GQA; RoPE, GELU _{tanh} ; Alternating Local and Global Attention; Logit Soft-Capping; RMSNorm for Pre and Post-Normalization
H2O-Danube3 [243]	500M; 4B	2024.7	Pre-train	Generic	Unknown	Three different training stages with different data mixes
Fox-1 [297]	1.6B	2024.6	Pre-train	Generic	Unknown (3T tokens)	GQA; Deep architecture
Rene ³	1.3B	2024.5	Pre-train	Generic	Dolma-1.7 [279]	Mamba-2 layers, sliding-window attention (SWA)
MiniCPM [128]	1.2B; 2.4B	2024.4	Pre-train	Generic	Dolma [279]; C4 [250]; Pile [78]; stack [156]; StarCoder [173]; UltraChat [79]; OssInstruct [330]; Evollnstruct [346]	Warmup-Stable-Decay (WSD) learning rate scheduler
OLMo [104]	1B; 7B	2024.2	Pre-train	Generic	Dolma [279] ⁴	SwiGLU; RoPE, Non-parameteric Layer Norm
TinyLlama [382]	1B	2024.1	Pre-train	Generic	SlimPajama [278] and StarCoder [173]	GQA, SiLU, Fully Sharded Data Parallel (FSDP), Flash Attention [67], xFormers [164]
Phi-1 [107]	1.3B	2023.6	Pre-train	Coding	CodeTextBook [107] ⁵	MHA, GELU _{tanh} , RoPE, FlashAttention
Phi-1.5 [178]	1.3B	2023.9	Pre-train	Generic	CodeTextBook [107]; Synthetic Datasets (20B)	MHA, GELU _{tanh} , RoPE, FlashAttention, Deep ZeRO Stage 2
Phi-2 [138]	2.7B	2023.12	Pre-train	Generic	CodeTextBook [107]; Synthetic Datasets (20T)	MHA, GELU _{tanh} , RoPE, FlashAttention, Deep ZeRO Stage 2
Phi-3 [1]	3.8B; 7B; 14B	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, Deep ZeRO Stage 2
OpenELM [214]	270M; 450M; 1.1B; 3B	2024.4	Pre-train	Generic	RefinedWeb [239], deduplicated PILE [97], a subset of RedPajama [61], and a subset of Dolma v1.6 [279]	No biases in FC layers; Pre-norm: RMSNorm; Pos encoding: RoPE; Attention: GQA; FFN: SwiGLU; Tokenizer: LLaMA-style
MobiLlama [299]	0.5B; 0.8B	2024.2	Pre-train	Generic	LLM360 Amber (Arxiv, Book, C4, Refined-Web, StarCoder, StackExchange, and Wikipedia)	GQA; SwiGLU; Parameter-sharing
MobileLLM [199]	125M; 350M	2024.2	Pre-train	Generic	Unclear (1T tokens)	SwiGLU FFN, deep and thin architectures, embedding sharing, and grouped query attention
StableLM [303]	3B; 7B	2023.4	Pre-train	Generic	RefinedWeb [239], subsets of the Pile [97], RedPajama [61] and the Stack [156], OpenWebText [101], OpenWebMath [238], and parts of CulturaX [229]	MHA; SiLU; Fine-tuning; DPO; Self-knowledge; RoPE; Layer-Norm; no Biases
StableLM 2 [24]	1.6B	2024.2	Pre-train	Generic	Pile [97]	MHA; GELU; Maximal Update Parameterization
Cerebras-GPT [78]	111M - 13B	2023.4	Pre-train	Generic	Pile [97]	MHA; GELU; Flash Attention [68]; RoPE [282]; ZeRO [252]
Pythia [27]	70M - 12B	2023.4	Pre-train	Generic	Pile [97]	MHA; GELU _{tanh} ; ALiBi Positional Embedding [246], Embedding LayerNorm [74]
BLOOM, BLOOMZ [162]	560M; 1.1B; 1.7B; 3B; 7.1B; 17B	2022.11	Pre-train	Generic	ROOTS [161] and 13 programming languages	MHA; GeLU; Learned Positional Embeddings
Galactica [293]	125M; 1.3B; 6.7B; 30B; 120B	2022.11	Pre-train	Scientific	Open-access scientific materials (106B tokens) but not released	MHA; ReLU
OPT [383]	125M; 350M; 1.3B; 2.7B; 5.7B	2022.5	Pre-train	Generic	Pile [97] and PushShift.io Reddit [23]	-
XGLM [186]	1.7B; 2.9B; 7.5B	2021.12	Pre-train	Generic	CC100-XL	-
GPT-Neo [29]	125M; 350M; 1.3B; 2.7B	2021.5	Pre-train	Generic	Pile [97]	-
Megatron-gpt2 [274]	355M; 2.5B; 8.3B	2019.9	Pre-train	Generic	Wikipedia [77], CC-Stories [304], RealNews [373], Openwebtext	-
MINITRON [224]	4B; 8B; 15B	2024.7	Pruning and Distillation	Generic	-	LR Scheduler: WSD; Pruning
Orca 2 [219]	7B	2023.11	Distillation	Generic	Orca 2 dataset	LLaMA-2-7B based; prompt erasing
Orca [223]	13B	2023.6	Distillation	Generic	FLAN-v2 [201]	Teachers are ChatGPT, GPT4, Explanation tuning; Progressive Learning
Dolly-v2 [62]	3B; 7B; 12B	2023.4	Instruction tuning	Generic	Databricks-dolly-15k [62]	-
LaMini-LM [153]	61M-7B	2023.4	Distillation	Generic	LaMini instruction dataset	-
Specialized FlanT5 [94]	250M; 760M; 3B	2023.1	Instruction Tuning	Generic (math)	GSM8K	Base model is FlanT5
FlanT5 [57]	80M; 250M; 780M; 3B	2022.10	Instruction Tuning	Generic	Muffin, T0-SF, SNI and CoT	Base model is T5
T5 [250]	60M; 220M; 770M; 3B; 11B	2019.9	Pre-train	Generic	C4 [250]	Encoder-decoder; Multilingual; No bias in LayerNorm; Layer-Norm external to residual path

traditional 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:

- The deep and thin architecture [199] highlights that deeper models are more effective than wider ones for improving performance.
- Embedding sharing [383] is significant because embedding layers make up a large part of the model’s parameters. For instance, with an embedding dimension of 512 and a vocabulary size of 32k, the input and output embedding layers contain 16 million parameters each. These layers account for over 20% of the total parameters in a 125M-parameter model. In sub-billion scale models, there is a return to the concept of sharing embeddings between the input and output layers, using the same weights for both, which makes the model more efficient and compact.
- Layer sharing [199] is advantageous for small Transformer models, increasing the number of hidden layers without extra storage costs.
- Shared FFNs [299] makeup 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.

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

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

- **Pile** [97]: 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 GiB.
- **C4 (Colossal Clean Crawled Corpus)** [250]: This dataset includes 350 billion tokens, representing a cleaned version of the Common Crawl web corpus, intended to capture a wide snapshot of the internet⁶.
- **The Stack** [156]: It contains 6 trillion tokens of source code from over 300 programming languages, useful for developing code-centric AI applications.
- **StarCoder** [173]: It features 35 billion tokens, predominantly Python code, aimed at programming language understanding and generation.
- **RedPajama** [61]: 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** [239]: 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** [23]: A around 5 billion tokens resource for social media data collection, analysis, and archiving, specifically of Reddit data, aiding research into social media dynamics.
- **CulturaX** [229]: It comprises 6.3 trillion tokens across 167 languages, supporting the development of models with extensive linguistic and cultural understanding.

From the analysis of these datasets, we can derive several critical insights regarding the development of SLMs: (i) Data Quality: Data quality is paramount in training effective SLMs. Most pre-training routines incorporate sophisticated filtering processes to exclude low-quality content, which may involve removing duplicates or irrelevant samples based on another LLM. For instance, the TinyStories corpus [86] is specifically curated to be simple and clear, resembling

⁶Available at <https://commoncrawl.org>

language suited for a three-year-old child, thereby facilitating the training of models that can comprehend and generate uncomplicated narratives effectively. RedPajama-V2 [61] is processed via the CCNet pipeline, including 30B documents with quality signals, and provides IDs for deduplicating documents to create 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 [15, 93, 107, 211]

5.1.2 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.

- **Direct Preference Optimization (DPO)** [249] introduces a novel methodology for optimizing language models based on human preferences, providing a simpler alternative to Reinforcement Learning from Human Feedback (RLHF). Unlike RLHF, DPO does not employ explicit reward modeling or reinforcement learning techniques. Instead, it uses a dynamic weighting mechanism to modify the log probabilities of preferred versus non-preferred responses. This approach helps to avoid model degradation issues commonly encountered with methods that rely solely on probability ratio targets. The DPO loss function is as follows:

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

Here, π_θ is the policy being optimized, and π_{ref} is the reference policy. The expectation is taken over a dataset D , comprising tuples (x, y_w, y_l) , where x is the input and y_w, y_l are the preferred and non-preferred outputs, respectively. The sigmoid function σ , and the scaling factor β adjust the log-ratio of probabilities from π_θ and π_{ref} , guiding the model towards producing outputs aligned with human preferences.

- **Reinforcement Learning from Contrast Distillation (RLCD)** [354] is designed to calibrate generative SLMs/LLMs towards embodying harmless and beneficial characteristics. The process commences with an unaligned LM and a series of initial prompts. The prompts are systematically modified into two variants $p+$ and $p-$, which are intended to promote and suppress, respectively, specific attributes such as helpfulness and harmlessness. Upon inputting these altered prompts into the LM, outputs $o+$ and $o-$ are generated, with $o+$ automatically marked as the preferred response. This automation expedites the training process by eliminating the requirement for subsequent evaluative scoring. The subsequent training adheres to the conventional Reinforcement Learning from Human Feedback (RLHF) framework. The model is trained on this artificially constructed paired preference data to cultivate a preference model. Subsequently, a reward model is extracted from this preference model. Proximal Policy Optimization (PPO) is then employed to refine the alignment of the original LM, ensuring that its outputs more closely reflect human values and expectations. This method effectively fine-tunes the model's responses, confirming their compliance with established ethical norms and user preferences.
- **Conditioned Reinforcement Learning Fine-Tuning (C-RLFT)**, proposed by OpenChat [315], aims to improve model performance by effectively incorporating low-quality data during SFT. Data quality plays a crucial role in model outcomes; for instance, OpenChat 6k [314] uses 6k high-quality ShareGPT data annotated by GPT-4 [2], excluding GPT-3.5 annotations, and outperforms Vicuna [53], which uses the full 70k ShareGPT dataset, and Alpaca [292] with 50k self-instruct samples. However, OpenChat 6k overlooks the potential of low-quality data, which can contribute knowledge and help the model distinguish between varying quality levels. C-RLFT addresses this by leveraging mixed-quality data with simple, coarse-grained reward labels that differentiate data quality—e.g., expert data with 1

credit and sub-optimal data with 0.1 credits. To mitigate imperfect reward signals, C-RLFT conditions data sources with distinct prompt tokens recognized by the model itself. Additionally, C-RLFT supports reward-free training and eliminates the need for costly human feedback, making it a flexible and efficient fine-tuning method for SLMs/LLMs. A similar strategy, **Data Mix** [243], involves training on English text in three stages, each with a different mix of data. In each stage, the proportion of noisy web data is gradually reduced in favor of higher-quality data.

- **Explanation Tuning**, proposed by Orca [223], 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 better ability of reasoning. Specifically, Orac adopts the training dataset sourced from FLAN-v2 [201], which includes five subsets covering various datasets. Five million questions from FLAN-v2 are initially processed through ChatGPT, and one million of these responses are further refined by GPT-4 to create the training set. Many models such as Orca 1 [223], StableBeluga [212] and Dolphin⁷ have benefited from Explanation Tuning, showing substantial improvements over traditional instruction-tuned models.
- **Progressive Learning**, proposed by Orca [223], 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.
- **Prompt Erasing** introduced by Orac 2 [219], 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.
- **Maximal Update Parameterization (μ P)** 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 [278] employs μ P to train its models efficiently.

5.1.3 Model Performance. To compare the performance of SLMs, we have extracted experimental results from two recent and concurrent studies published in June 2024, **OLMo** [104] and **MobiLlama** [299], and the recently proposed edge-device **Llama 3.2 1B & 3B** in September 2024⁸. 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** [120]: Evaluate broad knowledge across diverse fields such as humanities, science, technology, engineering, and management. It includes multiple-choice questions covering 57 tasks ranging from elementary mathematics to US history, computer science, law, and beyond, with a total of 14K test samples.
- (2) **HellaSwag** [372]: Assesses the model's ability to select the correct ending to scenarios from multiple options, testing common sense reasoning, including 10K test samples.

⁷<https://huggingface.co/datasets/cognitivecomputations/dolphin>

⁸<https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/>

Table 9. Comparative Performance of Various Models on Common Benchmarks: data from **MobiLlama** [299], **OLMo** [104], and **Llama 3.2**.

Model Size Range	Model	MMLU	HellaSwag	ARC	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
	LitLlama	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
1B-3B	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
	StableLM 1.6B	-	68.2	43.8	74.0	-
	Pythia 1B	-	44.7	33.1	69.1	-
	TinyLlama 1.1B	-	58.7	34.8	71.1	-
	OLMo-1B	-	62.5	34.5	73.7	-
	OLMo 1.2B	25.9	62.5	34.5	-	58.9
	Boomer 1B	25.4	31.6	22.3	-	51.0
	Pythia-Dedup 1B	24.3	49.6	29.1	-	54.0
	Falcon-RW 1B	25.4	63.1	35.1	-	61.9
>3B	Cerebras-GPT 1.3B	26.7	38.5	26.1	-	53.6
	Lamini 1.3B	28.5	38.1	26.6	-	50.6
	OPT 1.3B	24.6	54.5	29.6	-	59.7
	GPT-NEO 1.3B	24.8	48.5	31.3	-	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
	Gemma 2 2B	57.8	61.1	76.7	-	-
	Llama 3.2 1B	49.3	41.2	59.4	-	-
	Llama 3.2 3B	63.4	69.8	78.6	-	-
	Phi-3.5-mini 3.8B	69.0	81.4	87.4	-	-
>3B	Pythia 6.9B	-	63.8	44.1	75.1	-
	Falcon-7B	-	75.9	47.5	78.5	-
	LLaMA 7B	-	76.2	44.5	77.2	-
	Llama 2 7B	-	76.8	48.5	76.7	-
	MPT-7B	-	77.6	46.5	77.3	-
	RPJ-INCITE-7B	-	70.3	42.8	76.0	-
	OLMo-7B	-	76.4	48.5	78.4	-

- (3) **ARC** [58]: 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** [28]: 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** [262]: a dataset structured as a fill-in-the-blank task with binary options, designed to assess common-sense reasoning. The dataset includes 1,767 test samples by default splits.

Table 10. Comparison of MobiLlama 0.5B with large-base 1.2B, Llama2 7B, and Phi2-2.7B in terms of efficiency and resource consumption on low-end hardware devices [299].

Platform	Model	#Params	Precision	Avg Tokens/Sec	Avg Memory Consumption	Avg Battery Consumption /1k Tokens	CPU Utilization
RTX2080Ti	Llama2	7B	bf16	14.85	27793 MB	135.51 mAH	31.62%
	Phi2	2.7B	bf16	32.19	12071 MB	59.13 mAH	24.73%
	large-base	1.2B	bf16	50.61	6254 MB	18.91 mAH	18.25%
	MobiLlama	0.5B	bf16	63.38	3046 MB	8.19 mAH	14.79%
CPU-i7	Llama2	7B	4bit	5.96	4188 MB	73.5 mAH	49.16%
	Phi2	2.7B	4bit	22.14	1972 MB	27.36 mAH	34.92%
	large-base	1.2B	4bit	29.23	1163 MB	10.81 mAH	30.84%
	MobiLlama	0.5B	4bit	36.32	799 MB	4.86 mAH	24.64%
Snapdragon-685	Llama2	7B	4bit	1.193	4287 MB	10.07 mAH	77.41%
	Phi2	2.7B	4bit	2.882	1893 MB	14.61 mAH	56.82%
	large-base	1.2B	4bit	6.687	780 MB	6.00 mAH	17.15%
	MobiLlama	0.5B	4bit	7.021	770 MB	5.32 mAH	13.02%

Accuracy is used as the evaluation metric in the table. **Open Language Model (OLMo)** [104] is publicly available with its training data and code ⁹. **MobiLlama** [299] 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 show high efficiency on diverse hardware (Table 10). From Table 9, 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) **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, memory, and GPU utilization. (4) Popular techniques such as pruning, quantization, distillation, SFT, and DPO, utilized in **Llama 3.2**, have substantially enhanced SLM performance.

Insights: We draw several key insights from the development of generic-specific 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. For example, the Dolma project enhances data quality through multi-stage data filtering processes.
- Post-pretraining, meticulous fine-tuning is often required to enhance the safety of SLMs, involving strategies to better distill capabilities from LLMs. Common strategies include explanatory tuning, progressive learning, and prompt erasing.

⁹<https://allenai.org/olmo>

Table 11. High-level Overview and Training Details of Specific-domain Small Language Models

Model	#Params	Date	Base Models	Domain	Training Datasets	Train Techniques
Hippocrates [3]	7B	2024.4	Instruction Tuning (LLaMA2 [302], Mistral [143])	Healthcare	Medical Guidelines, PMC-Patients [394], and PubMedQA-contexts [146]	Continual pre-training, instruction tuning, RLHF
BioMedLM [30]	2.7B	2024.3	From scratch and Fine-tuning	Healthcare	PubMed [97]	FSDP
BioMistral [160]	7B	2024.2	Mistral [143]	Biomedicine	PubMed [97]	Continual pretraining
MentalLLaMA [355]	7B; 13B	2023.9	Instruction Tuning (LLaMA2 [302])	Healthcare	IMHI dataset	RLHF; PEFT
AdaLM [362]	34M	2021.6	Distillation (BERT [77] or MiniLM [321])	Healthcare	PubMed [97]	Continual pretraining, Adapt-and-Distill
Rho-1 [187]	1B, 7B	2024.4	TinyLlama-1.1B [382], Mistral-7B [143]	Science (Mathematics)	OpenWebMath [238]	Continual pretraining
ChemLLM [379]	7B	2024.4	Instruction Tuning (InternLM2)	Science (Chemistry)	ChemData	Continual training and fine-tuning
SciGLM [378]	6B	2024.3	Instruction Tuning (ChatGLM-6B)	Science	SciInstruct	Self-reflective instruction annotation
Llemma [20]	7B	2023.10	Code Llama 7B	Science (Mathematics)	Proof-Pile-2 [20]	Continual pre-training
OceanGPT [26]	2B, 7B, 14B	2023.10	LLaMA2 [302]	Science (Ocean)	Open-access literature, DoIN-STRUCT	Continual pre-training, Instruction tuning
AstroLLaMA [230]	7B	2023.9	Tuning (LLaMA-2-7B)	Science (Astronomy)	arXiv abstracts from Kaggle	Continual training
DARWIN [344]	7B	2023.8	LLaMa 7B	Science (physics, chemistry, and material)	SciQ [332], Scientific paper [344], FAIR [344]	Fine-tuning
MindLLM [359]	1.3B, 3B	2023.10	From-scratch and Supervised Fine-tuning	Law, Finance	Pile [97], Wudao [369], CBooks	Train on Bilingual Mixture Data, SFT

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, 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 LLMs 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 framework designed for the medical domain, offering unrestricted access to its training data, codebase, checkpoints, and evaluation protocols ¹⁰. It integrates specialized medical knowledge through an extensive pre-training corpus from three datasets: Medical Guidelines, PMC-Patients [394], and PubMedQA-contexts [146]. This corpus, with around 300 million training tokens, ensures the model's proficiency in medical terminology and practices. The Hippo series, a customized 7B model, undergoes continuous pre-training, instruction tuning, and reinforcement learning from human and AI feedback. It has specific datasets for instruction tuning and preference learning. Fine-tuned on Mistral and LLaMA2, Hippo surpasses existing 7B and 13B models, matching or exceeding 70B models in some cases. For example, Hippo-Mistral 7B can achieve 59.9% accuracy on MedQA, outperforming Meditron 70B [47] whose performance is 58.5%. It is evaluated on six medical downstream tasks including MedMCQA [237], PubmedQA [146], MedQA [145], and the USMLE-step1, USMLE-step2, and USMLE-step3 exams [145].

BioMedLM [30] introduces a 2.7 billion parameter GPT-style autoregressive model, trained exclusively on PubMed abstracts and full articles [97]. After fine-tuning, BioMedLM achieves robust biomedical question-answering performance, comparable with larger models, scoring 57.3% on MedMCQA (dev) and 69.0% on the MMLU medical genetics exam. It is also fine-tuned to provide useful answers to patient queries, demonstrating the potential of smaller models as a transparent, privacy-conserving, cost-effective, and environmentally-friendly foundation for specific NLP applications like biomedicine. The model is available on Hugging Face Hub ¹¹.

AdaLM [362] develops small, fast, and effective domain-specific pre-trained language models. This approach continues the training of a medical domain-specific SLM atop an existing pre-trained model. To address domain distribution shifts, four combined strategies of domain adaptation and compression were tested: direct domain-specific pretraining, distillation followed by adaptation, adaptation followed by distillation, and simultaneous adaptation of both large and small models during initial distillation (adapt-ant-distill). For biomedical applications, a 16GB corpus from PubMed ¹² abstracts was used to adapt a BERT_base model (12 layers, 768 hidden size) [77]. The model was evaluated on three downstream tasks: JNLPBA [60], EBM PICO [233], and ChemProt [157], demonstrating that adaptation and distillation are optimal for developing task-agnostic, domain-specific small models.

MentalLLaMA [355] models explainable mental health analysis as a text generation task and establishes the first IMHI dataset with 105K samples for instruction tuning of LLMs. It compiles data covering eight tasks from ten sources and uses expert-designed prompts with ChatGPT for explanations. The data undergo rigorous automatic and manual evaluations for accuracy and quality. Based on LLaMA2, MentalLLaMA includes a 7B variant and is the first open-source LLM for explainable mental health analysis on social media. Evaluated on the IMHI benchmark, MentalLLaMA matches state-of-the-art methods in accuracy and produces human-level explanations, showing strong generalizability on unseen tasks. The project code is available at ¹³.

5.2.2 SLMs for Science. **SciGLM** [378] is a scientific language model capable of collegiate-level scientific reasoning. Central to this method is a novel self-reflective instruction annotation framework addressing data scarcity in science. This framework uses GPT-4 [2] to generate step-by-step reasoning for unlabeled scientific problems, followed by

¹⁰<https://cyberiada.github.io/Hippocrates/>

¹¹<https://huggingface.co/stanford-crfm/BioMedLM>

¹²<https://pubmed.ncbi.nlm.nih.gov/>

¹³<https://github.com/SteveKGYang/MentalLLaMA>

Table 12. Prompts for self-reflective instruction annotation framework

Chain-of-Thought	[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.

self-reflective critique and revision. Self-reflective critique is a method for annotating scientific questions that involve three steps with designed prompts in Table 12: (i) employing a Chain of Thought (CoT) prompt, (see Prompt 1 in the table), to generate step-by-step answers; (ii) using a reflective prompt, (see Prompt 2), for responses with incorrect solutions, aiding GPT-4 in generating correct answers by analyzing its previous errors; and (iii) incorporating the actual answer directly into the prompt (see Prompt 3) to further assist in resolving the question. To eliminate low-quality questions and answers resulting from incomplete OCR, an instruction-quality classifier is trained and applied for further revisions. Leveraging this, the SciInstruct dataset is curated, encompassing physics, chemistry, mathematics, and formal proofs, enhancing the scientific and mathematical reasoning abilities of the ChatGLM-6B [83] language model. Notably, SciGLM enhances the performance of the base model (ChatGLM3-6B-Base) by 3.06% in average scientific QA accuracy across CEval-Hard [134], CEval-Sci [134], MMLU-Sci [120], SciEval [283], and SciBench [322]. For broader community benefit, SciInstruct, SciGLM, the self-reflective framework, and fine-tuning code are available at ¹⁴.

Llemma [20], an SLM tailored for mathematical reasoning, evolves from the Code Llama framework [260]. Trained using a standard auto-regressive language modeling objective, Llemma’s 7B model processes 200B tokens, including a 55B-token subset from the proposed Proof-Pile-2 dataset. This dataset comprises a rich mix of scientific papers, web data on mathematics, and mathematical code, with a content cutoff in April 2023. By continual pre-training on Proof-Pile-2, Llemma significantly enhances its few-shot performance across key mathematical benchmarks: MATH [120], GSM8k [59], OCWCourses [166], MMLU-STEM [120], and SAT. This continual training enables Llemma to outperform all comparable open-weight language models. For instance, it achieves a 71.9% accuracy on the SAT, significantly higher than Code Llama’s 40.6%, underscoring the profound impact of targeted continual pre-training in boosting mathematical problem-solving capabilities.

ChemLLM [379] introduces a language model framework for chemistry. It incorporates **ChemData**, a dataset crafted for instruction tuning. This dataset transforms structured chemical data into dialogue formats suitable for training. Additionally, it includes **ChemBench**, a robust benchmark consisting of 4,100 multiple-choice questions. These questions span nine fundamental chemical tasks and aim to minimize linguistic style bias in model outputs. It is trained on the **InternLM2-Base-7B** model [35], initially enhancing general language ability with a multi-corpus of 1.7 million Q&A pairs from Hugging Face. In the second phase, it is fine-tuned using a mix of ChemData for chemical knowledge and the multi-corpus to retain general functionality. As a result, ChemLLM excels in various interdisciplinary chemical tasks through fluid dialog interactions, achieving core chemical task results comparable to GPT-4 [2] and showing competitive performance with similar-sized LLMs. For instance, in the Mol2caption task, ChemLLM achieves a score of 92.6, outperforming GPT-3.5 and slightly below the score attained by GPT-4 [2].

¹⁴<https://github.com/THUDM/SciGLM>

AstroLLaMA [230] effectively addresses the gap in the performance of LLMs in specialized fields like academic astronomy by introducing a domain-specific model. This 7-billion-parameter model, fine-tuned on over 300,000 astronomy abstracts from arXiv available on Kaggle¹⁵ using LLaMA-2 [302], adheres to causal language modeling objectives from its pre-training phase. It achieves a perplexity 30% lower than Llama2, indicating substantial domain adaptability. Despite fewer parameters, AstroLLaMA surpasses standard foundational models in generating insightful and scientifically relevant outputs. With its significant potential for further fine-tuning, AstroLLaMA is publicly available¹⁶, enhancing research in astronomy through tasks like automated paper summarization and conversational agent development.

5.2.3 SLMs for Finance and Law. **MindLLM** [359] introduces a series of bilingual, lightweight language models, developed from scratch, with configurations of 1.3 billion and 3 billion parameters. It incorporates experiences from large model development, covering data construction, model architecture, evaluation, and applications. The primary English pre-training data is sourced from the Pile dataset [97], and the Chinese pre-training data combines resources from WuDao [369], CBook¹⁷, and various Chinese websites. MindLLM consistently matches or surpasses other large models in public benchmarks. Bilingual training from scratch is advocated as a superior method for capacity learning and avoiding catastrophic forgetting, featuring a specialized instruction tuning framework designed to enhance smaller models' capabilities effectively. Additionally, supervised fine-tuning in law and finance explores specific vertical domains. In law, publicly available legal data, scenario-based Q&A, legal references from LaW-GPT [124], and NLP-based legal tasks from DISC-LawLLM [371] are used. Due to the absence of public objective benchmarks in the legal field, the robust ChatGPT model serves as an evaluator for reasoning on manually constructed data. In finance, EastMoney¹⁸ is selected as the data source. Financial sentiment data is formatted into a text classification task, where inputs include financial news and outputs are labels for five categories: positive, negative, neutral, very negative, and very positive. MindLLM-1.3B and 3B models demonstrate superior performance, achieving accuracy rates of 47.79% and 46.40%, respectively, to similar-sized models and outperform ChatGLM2-6B (45.79%) and Open-LLaMA-7B (28.38%) after fine-tuning.

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 [230, 355, 378, 379]. These datasets are often annotated using LLMs like GPT-4 and used to train or fine-tune general models such as LLaMa-2-7B [3, 30]. To ensure the data quality, specialized annotation frameworks are developed, such as Zhang et al. [378].
- In domains with abundant corpora, training a general model from scratch and fine-tuning it using SFT [359] 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 [362].

¹⁵<https://www.kaggle.com/Cornell-University/arxiv>

¹⁶<https://huggingface.co/universeTBD/astrollama>

¹⁷<https://github.com/FudanNLPLAB/CBook-150K>

¹⁸<https://www.eastmoney.com/default.html>

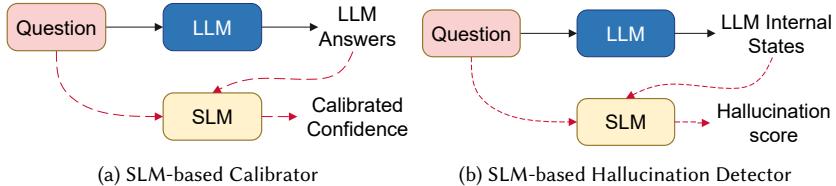


Fig. 18. Architectures of Enhancing Calibration and Hallucination Detection of LLMs.

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; and (v) evaluating LLMs using SLMs. A summary 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 [142, 310]. 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.

Enhancing Calibration and Hallucination Detection of LLMs As illustrated in Figure 18 (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** [306] uses an auxiliary DeBERTaV3 model [117] to assess LLM confidence in open-question scenarios, aiming to improve uncertainty expression and response adjustment. Similarly, Zhao et al. [392] has developed a self-supervised approach that generates risk scores for each response to calibrate LLM confidence, utilizing a small BERT model [77] to synchronize LLM outputs with other weak supervision sources. As shown in Figure 18 (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. [349] develop a lightweight detector that analyzes token contributions to hallucinations, outperforming both model-free baselines and quality estimation classifiers. Furthermore, **SAPLMA** [19] 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% to 83%.

Enhancing Retrieval-Augmented Generation Generally, as shown in Figure 19, 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** [289] is a small proxy model that assesses the necessity for LLM retrieval by

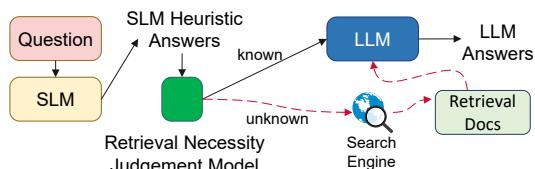


Fig. 19. Architecture of SLM as a Heuristic RAG Prober.

Manuscript submitted to ACM

Table 13. SLMs help LLMs in different aspects

Aspect	Representative work	Key point
SLM for reliable LLM generations	APRICOT [306]	Trains a small auxiliary model to predict LLM's confidence using only textual inputs and outputs.
	POLAR [392]	Uses a small BERT model to calibrate LLM responses and assign risk scores.
	Hallucination Detector in NMT [349]	Uses lightweight classifiers to detect hallucinations in Neural Machine Translation.
	SAPLMA [19]	Uses a BERT Small Language Model as a classifier to assess the truthfulness of statements accurately.
	Question Decomposer [339]	Distilled SLM decomposes complex questions to aid reasoning.
	SuperICL [347]	SLM Plug-ins provide confidence and prediction for contextual exemplars to aid in-context learning.
	SuperContext [357]	Specific SLM enhances ICL by providing confidence and predictions to overcome out-of-domain challenges.
	Self-RAG [16]	A proxy model labels special tokens during RAG data generation for fine-tuning.
	SKR [324]	Trains a small model to detect its self-knowledge for better use of external knowledge.
	SlimPLM [289]	Detects missing knowledge in LLMs with a slim proxy model, enhancing the LLM's knowledge integration.
SLM for extracting LLM prompts	In-Context RALM [254]	Trains a RoBERTa-based reranker for top-k BM25 documents using LM signals to enhance LM gains.
	CRAG [351]	Trains a lightweight evaluator to assess document quality and triggers actions based on confidence levels.
	Prompt Extraction [386]	Small model trained to predict confidence of extracted system prompt from adversarial prompts.
	Prompt Stealing Attacks [266]	Uses small models fine-tuned as parameter extractors to facilitate hierarchical prompt reconstruction.
SLM for Fine-tuning LLMs	Output2prompt [376]	Uses a sparse encoder-decoder-based T5 small model to reverse-engineer LLM inputs from outputs.
	Model Purifying [175]	Uses SLMs to ensemble with LLMs, mitigating negative effects from uncurated data.
	\mathcal{LP} [215]	Learning Percentage as a difficulty metric.
SLM for LLM applications	Emulated Fine-tuning [218]	Emulates pre-training and fine-tuning at different scales by summing base log probabilities with behavior deltas.
	CROSSLM [73]	SLMs enhance LLMs by generating task-specific high-quality data.
	SLCoLM [290]	Uses SLM predictions to guide the LLM generation process in Chinese Entity Relation Extraction.
	HEF [361]	Uses SLMs as flexible plugins to improve LLM's nuanced understanding.
	Contrastive decoding [176]	Enhances text quality by maximizing the difference between expert and amateur log probabilities.
SLM for LLM evaluation	SLIDE [391]	Utilizes SLMs trained via contrast learning to distinguish and score responses in dialogue scenarios effectively.

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**) [324] enables SLMs to autonomously decide when LLMs should operate independently, based on their self-assessment of knowledge limitations. Further, **SELF-RAG** [16] improves the factual accuracy and quality of LLM outputs through on-demand retrieval and self-reflection. This method employs a small critic 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** [144] employs SLMs to calculate the relevance of documents to a query x^{que} using perplexity, as formalized by the equation:

$$r_k = -\frac{1}{N_c} \sum_i \log p_{SLM}(x_i^{que} | x_k^{doc}), \quad k \in \{1, 2, \dots, K\} \quad (6)$$

where x_i^{que} is the i -th 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** [351] employs SLMs as evaluators of document relevance in the same way. *In addition, other research employs SLMs as re-rankers to refine the order of documents provided by initial retrieval efforts such as BM25 [258].* **In-Context RALM** [254] positions SLMs as rankers, optimizing the document sequence with a fine-tuning process on RoBERTa [195] as defined by the loss function:

$$\min_{ranker} \sum_{i=1}^k -\log p_{rank}(d_i | x_{\leq s_j}) \cdot p_\theta(y | d_i; x_{\leq s_j}) \quad (7)$$

where $x_{\leq s_i}$ is a prefix sampled from the training data, $y = x_{s_i+1}, \dots, x_{s_i+s}$ represents the text to be generated in the next stride, $p_\theta(y | d_i; x_{\leq s_i})$ denotes the probability of the LLM generating y given d_i and $x_{\leq s_i}$, and $p_{rank}(d_i | x_{\leq s_j})$ is the ranking score of d_i .

Enhancing Reasoning Capabilities of LLMs As illustrated in Figure 20, 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** [347] 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** [357] 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 [339]. This strategy not only increases efficiency but also reduces deployment costs when SLMs and LLMs are used collaboratively, transforming complex tasks into manageable segments.

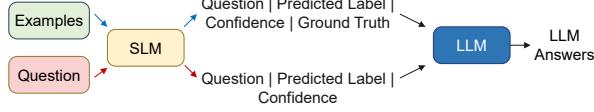


Fig. 20. SLM transfers knowledge into ICL.

Alleviate Copyright and Privacy Issues of LLMs As depicted in Figure 21, 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 full-scale LLMs. To harness the combined benefits of both models, Li et al. [175] integrates untrusted LLMs with benign SLMs using the CP- Δ

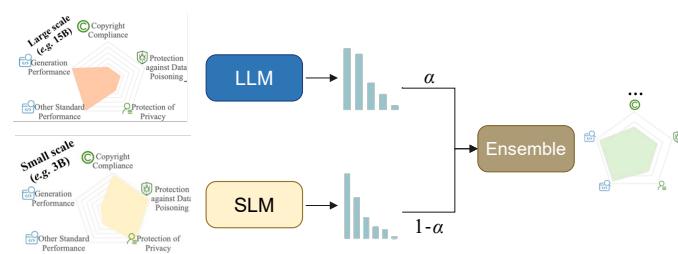


Fig. 21. Architecture of SLM-based Data Protection

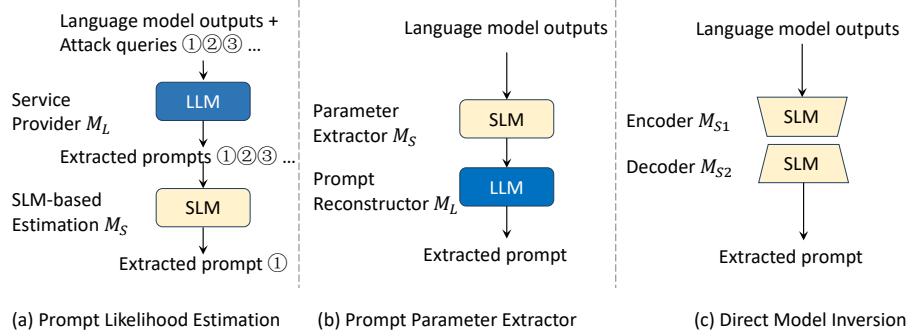


Fig. 22. SLM for LLM Prompt Extraction Paradigm. M_S denotes small language models and M_L denotes large language models. (a) SLM-based prompt estimation tries various attack prompts; M_S selects the most likely extracted one. (b) SLM-based Parameter Extractor identifies the type of input prompt. (c) SLM-based Model Inversion uses M_S to invert the LLM output back into the input.

KL algorithm to mitigate adverse effects while preserving performance. The equation is:

$$p(y|x) = \frac{p_l(y|x) \cdot p_s(y|x)}{Z(x)} \quad (8)$$

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:

$$z_p(\cdot|x) \propto \alpha z_l(\cdot|x) + (1 - \alpha) z_s(\cdot|x) \quad (9)$$

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 22.

SLM-based prompt likelihood estimation, as illustrated in Figure 22 (a), Zhang et al. [386] proposes 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 [118] is then used to select the most likely secret prompts from the output.

SLM-based prompt parameter extraction, as shown in Figure 22 (b), Sha and Zhang [266] utilizes an SLM as a Parameter Extractor to extract prompt parameters from LLM outputs. They employ a specialized BERT model [77] 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 22 (c), the method of using an SLM as a Direct Inversion Model is designed to reverse-engineer LLM outputs back to their original prompts [376]. They train a sparse encoder-decoder T5 model [250] with 222M parameters on the Instructions-2M dataset [221], 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 $p(x|y_1, \dots, y_n; M_{S1}, M_{S2})$, 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 [302] checkpoint on 32 NVIDIA A100 GPUs with 80GB memory using bfloat16 format requires approximately 70 hours [219]. 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 23.

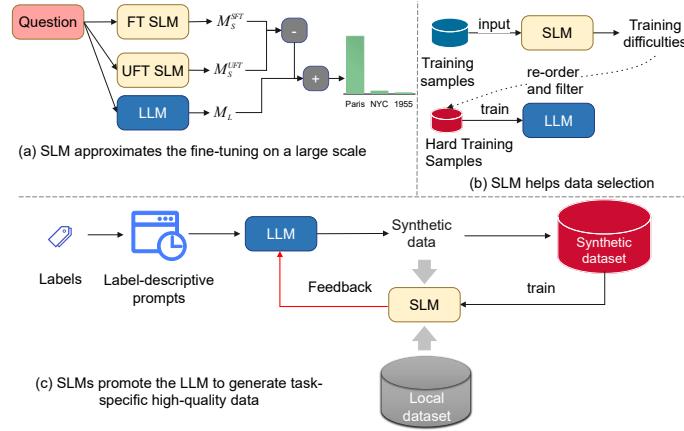


Fig. 23. SLM for LLM Fine-tuning.

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 23 (a), Emulated Fine-Tuning (EFT) [218] 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 [9] while capturing most benefits of fine-tuning larger models such as Falcon-180B, benefiting applications such as dialogue, question-answering, and code generation. Similarly, **Proxy-tuning** [189] 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.

SLMs play a role in selecting high-quality fine-tuning data for LLMs. Figure 23 (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. [287] and further advanced by Mekala et al. [215], the *learning percentage* $LP(i)$ is a metric used to curate high-quality datasets with hard samples:

$$LP(i) = \frac{P_{i-1} - P_i}{P_0 - P_n} \quad (10)$$

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.

SLMs enhance the quality of LLM-generated data for specific tasks. As depicted in Figure 23 (c), CROSSLM [73] 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 to produce 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 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) [176] improves coherence and lexical diversity by leveraging the differential capabilities between a large model, OPT-13B [383], and a smaller model, OPT-125M. As illustrated in Figure 24, CD improves content quality by sampling generation based on the difference in log probabilities, $\log p_{\text{EXP}} - \log p_{\text{AMA}}$, 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.

In knowledge injection, general LLMs based on open-domain data may lack the domain-specific knowledge necessary for specialized tasks in fields such as law and medicine. In contrast, domain-specific SLMs can provide crucial domain knowledge and adapt this knowledge in a format that LLMs prefer. To address this, BLADE [168] augments black-box LLMs with small

domain-specific models. The structure of BLADE comprises both the black-box LLMs, which offer strong language comprehension and reasoning capabilities, and small domain-specific LMs, which retain specialized insights. As illustrated in Figure 25, the BLADE approach involves three critical steps: 1) pre-training the SLM on domain-specific data to encapsulate the necessary expertise, 2) fine-tuning the model using knowledge instruction data to refine its application to task-specific requirements, and 3) employing joint Bayesian optimization to harmonize the capabilities of the general-purpose LLM with the small LM, enhancing overall performance.

In relation extraction, a field limited by scarce labeled data and prevalent long-tail categories, the “**Train-Guide-Predict**” framework [290] 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

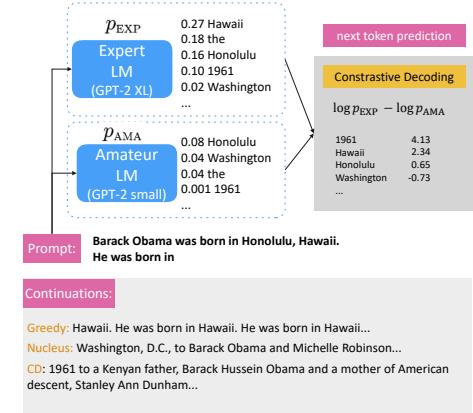


Fig. 24. Contrastive Decoding [176].

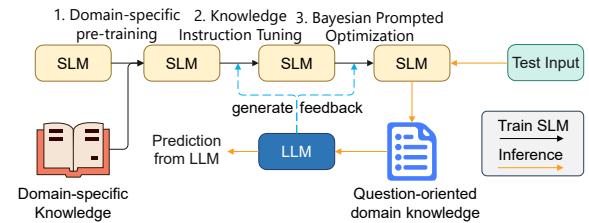


Fig. 25. BLADE Framework [168].

framework leverages the strengths of both models: it utilizes SLMs for acquiring task knowledge and guiding the LLM's generative process with initial SLM predictions, enhancing the LLM's handling of underrepresented categories.

In generating empathetic responses, LLMs are proficient in expressive capabilities but often lack a nuanced understanding of emotions and cognition. Thus, the Hybrid Empathy Framework (HEF) [361] enhances this by integrating Small Empathy Models (SEMs) to deepen the emotional and cognitive insight of LLMs. This framework uses a two-tiered emotion prediction approach: SEMs first determine primary emotions, which then guide the LLM to focus on these emotions and the key emotional triggers. This strategy leads to more accurate and empathetic responses.

6.5 SLM for LLM Evaluation

SLMs can also enhance the evaluation of LLMs, especially in dialogue generation, where handling diverse potential responses is inherently challenging. In dialog evaluation, generating both dialog and reference responses computationally is complex, making accurate assessment difficult due to the multiple plausible but semantically different responses possible for a single dialog context. Sole reliance on LLMs for evaluation can lead to problems such as dependency on prompt wording and inconsistent results. Therefore, a specifically trained SLM can address these issues. **SLIDE** framework [391] employs contrastive learning to fine-tune an SLM to effectively distinguish between positive and negative responses. This trained SLM is subsequently combined with an LLM to assign a score to each response, optimizing the evaluation process. The scoring method used is formalized as follows:

$$score = \begin{cases} score_{SLM}, & \text{if } score_{SLM} \geq 0.5 \\ score_{LLM}, & \text{if } score_{LLM} < 0.5 \\ \frac{score_{SLM} + score_{LLM}}{2}, & \text{otherwise} \end{cases} \quad (11)$$

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.

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.

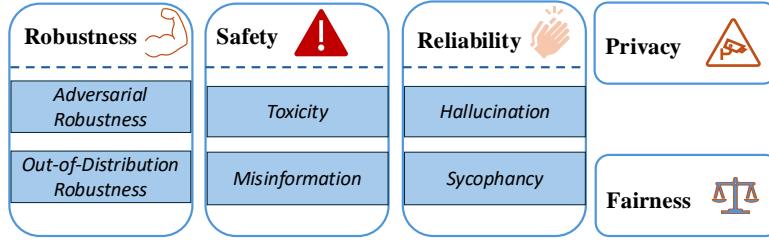


Fig. 26. Scenarios we discuss in this section. The taxonomy is inspired by previous works [284, 312]. Please note that the trustworthy scenarios listed here are not exhaustive.

7 TRUSTWORTHINESS IN SMALL LANGUAGE MODELS

Language models have become ubiquitous in our daily lives, and we increasingly rely on them. However, they pose risks regarding their limitations in some trustworthy dimensions like privacy and fairness. These concerns are especially critical in high-stakes domains such as healthcare [116] and finance [180]. Consequently, numerous studies have emerged to evaluate the trustworthiness of LMs [81, 85, 123, 158, 158, 220, 226, 242, 312, 328, 370]. Though there are already a lot of works exploring LMs' trustworthiness, most of them focus on LLMs. In this section, we investigate works that contain LMs of around 7B size and below. Note that we do not include the specific attack methods [36, 46, 135, 401] or work [356] that only focuses on early pre-trained LMs like BERT [77] as they are already covered in previous survey papers [72, 102, 110, 255]. Inspired by previous works [284, 312], we discuss the following five key trustworthy scenarios: *robustness*, *privacy*, *reliability*, *safety*, and *fairness*, as shown in Figure 26. We consider two scenarios for robustness: Adversarial (Adv) Robustness [313] and Out-of-Distribution (OOD) Robustness [34, 190]. For safety, we explore two key concerns: Misinformation [307] and Toxicity [331]. For reliability, we focus on Hallucination [132] and Sycophancy [269]. 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. Next, we briefly introduce some representative works.

Holistic Evaluation of Language Models (HELM) [182] 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 [323] introduces a dataset to evaluate how LMs act when they face content that should not be answered. Wang et al. [323] also label the output of several LMs output on their dataset and then use the labeled data to train some classifiers. PromptRobust [397] 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 [170] 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. [220] evaluates 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 [169] 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 [65] introduces around two thousand carefully 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 [88] 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 [284] is a comprehensive

Table 14. Comparison of Different Works that Evaluate the Trustworthiness Issues in LMs. Please note that for the "No. of LMs" attribute, compressed or pruned LMs are not included in the count.

Paper	Adv Robustness	OOD Robustness	Toxicity	Misinformation	Hallucination	Sycophancy	Privacy	Fairness	Have Compressed SLMs
HELM [182]	✓	✗	✓	✓	✗	✗	✗	✓	✗
Do-Not-Answer [323]	✗	✗	✓	✓	✗	✗	✓	✓	✗
PromptRobust [397]	✓	✗	✗	✗	✗	✗	✗	✗	✗
HaluEval [170]	✗	✗	✗	✗	✓	✗	✗	✗	✗
Mo et al. [220]	✓	✗	✓	✗	✓	✓	✓	✓	✗
PrivLM-Bench [169]	✗	✗	✗	✗	✗	✗	✓	✗	✗
FFT [65]	✗	✗	✓	✓	✓	✗	✗	✓	✗
ROBBIE [88]	✗	✗	✓	✗	✗	✗	✗	✓	✗
TrustLLM [284]	✓	✓	✓	✓	✓	✓	✓	✓	✗
RAmBLA [31]	✓	✗	✗	✗	✓	✗	✗	✗	✗
JailbreakBench [39]	✗	✗	✓	✓	✗	✗	✓	✗	✗
Xie et al. [345]	✗	✗	✓	✗	✓	✗	✗	✗	✗
OR-Bench [64]	✗	✗	✓	✓	✗	✗	✓	✗	✗
SORRY-Bench [343]	✗	✗	✓	✓	✗	✗	✓	✗	✗
BeHonest [51]	✗	✗	✗	✓	✓	✓	✗	✗	✗
Hong et al. [123]	✓	✓	✓	✗	✗	✗	✓	✓	✓
RUPBench [328]	✓	✗	✗	✗	✗	✗	✗	✗	✗
Nakka et al. [226]	✗	✗	✓	✗	✗	✗	✓	✓	✗

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 [31] 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 [39] 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. [345] tests online safety analysis methods, filling the gap where no methods focus on the generation phase. OR-Bench [64] 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 [343] 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 [51] evaluates the honesty of LMs from three aspects: Self-Knowledge, Non-Deceptiveness, and Consistency. They use many different metrics for each aspect. For example, sycophancy rate and 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 [302] and Llama-3 [84] series. Hong et al. [123] examines 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 [328] 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. [226] investigates 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.

We summarize all the works above in Table 14. Please note that the dimensions listed in the table reflect only those relevant to our current focus; additional dimensions may be discussed in those works, but not included in the table. For example, TrustLLM [284] also explores Machine Ethics.

8 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 small language models.

8.1 Developing Efficient SLM Model Architecture

Although Transformers [308] 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 [105], a promising alternative, has emerged. It adapts the Multi-Head Attention mechanism to dynamically scale attention based on task demands, boosting efficiency, especially for simpler sequences. Mamba retains the content-aware capabilities of Transformers but scales linearly with input length, improving efficiency in both training and inference. Hence, developing domain-specific SLMs based on Mamba offers a promising research direction. Additionally, exploring other model architectures that enhance performance while reducing computational demands is also important.

8.2 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.

8.3 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.

8.4 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 status. 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 Retrieval-Augmented Generation (RAG) and Mixture of Experts (MoE) are employed. The adoption of RAG in SLMs shows significant promise [192], 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 prompting engineering to maximize SLM performance, while accounting for SLMs' constraints, warrants further investigation.

8.5 Applications of SLMs: OOD, Personalization, Lifelong Learning

In real-world applications, SLMs often encounter out-of-distribution samples, 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:

- **Enhancing OOD with Integrated RAG and Self-Adaptive Techniques.** SLMs often encounter out-of-distribution (OOD) samples due to limited capabilities and training data. To enhance inference performance, Retrieval-Augmented

Generation (RAG) and self-adaptive methods are utilized. RAG provides additional information, while self-adaptive methods generate pseudo labels for self-training. However, inaccuracies in pseudo-labels can reduce their effectiveness. A promising future direction is to integrate RAG with self-adaptive techniques to improve pseudo-label accuracy.

- **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 [396] 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.

8.6 SLMs Assisting LLMs

In Section 6, we introduced existing works on the use of SLMs to assist LLMs. For instance, EFT [218] 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 [289] detects missing knowledge in LLMs using a slim proxy SLM to accelerate knowledge injection; Contrastive Decoding [176] 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 [175, 209]. 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 [16, 189, 218, 289] 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 [314]. However, directly eliminating low-quality content can diminish certain functionalities of LLMs, such as versatility [315]. Therefore, it is crucial to define more refined data quality assessment criteria across dimensions such as factuality, safety, and diversity [334]

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 [200], 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 Retrieval-Augmented Generation (RAG) using LLMs, differing query requirements between LLMs and search engines pose a challenge. The query for LLMs is often abstract and difficult to handle by search engines, which 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 noises, 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.
- **Cloud LLM-Local SLM Collaboration.** Cloud LLMs are commonly used, but privacy concerns limit them to processing all request aspects, necessitating the use of local SLMs to handle sensitive data [380]. This setup allows for the specialized capabilities of SLMs to be leveraged on-device while still benefiting from the generalized functionalities of cloud-based LLMs. Researching optimal designs for collaboration that balance and enhance the capabilities of cloud-based LLMs and local SLMs is a crucial future direction.
- **Optimizing Specialized Training for Enhanced Model Performance.** Specialized training and fine-tuning strategies are crucial for enhancing language model performance in specific tasks, such as coding. Models like DeepSeek-Coder illustrate the benefits of focused training, achieving significant gains in specialized domains. However, these models often struggle with broader or more complex requests, revealing a key trade-off. Future research could investigate integrating these specialized models with general SLMs and LLMs to merge specialization benefits with broader model versatility.

8.7 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 [284], is essential to thoroughly evaluate the trustworthiness of SLMs and understand their reliability across various applications.
- **Developing Trustworthy SLMs.** In addition to understanding the trustworthiness of SLMs, developing trustworthy SLMs and improving their trustworthiness is also of critical importance. Essentially there are three directions worthy of investigation: (i) One promising direction is how to train trustworthy SLMs from scratch; (ii) As one can also obtain SLMs by compression LLMs, another promising direction is how to obtain trustworthy SLMs during compression or

quantization of LLMs. In particular, if the LLM is trustworthy, how to preserve its trustworthiness during compression? If the LLM is non-trustworthy, how do we make SLM trustworthy during compression? (iii) Given a non-trustworthy SLM, how to fine-tune it to make it more robust?

9 CONCLUSION

This paper provides a comprehensive survey of SLMs with up to 7 billion parameters. We first clarify the definition of SLMs due to the current lack of clarity. We then outline foundational concepts essential for building SLMs. Subsequently, our review focuses on enhancement techniques such as knowledge distillation and quantization, along with strategies for adapting LLMs to SLM contexts. The subsequent section surveys representative SLMs, discussing their preferred datasets and architectural choices. We also examine their applications across various tasks as well as deployment strategies on devices and explore their role in augmenting the capabilities of LLMs. Another critical aspect discussed is their trustworthiness. The paper concludes with key insights intended to guide future research on small language models. Specifically, we highlight the following promising future directions.

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 preprint arXiv:2404.14219* (2024).
- [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 preprint arXiv:2303.08774* (2023).
- [3] Emre Can Acikgoz, Osman Batur İnce, Rayene Bench, Arda Anıl Boz, İlker Keser, Aykut Erdem, and Erkut Erdem. 2024. Hippocrates: An Open-Source Framework for Advancing Large Language Models in Healthcare. *arXiv preprint arXiv:2404.16621* (2024).
- [4] Harshavardhan Adepu, Zhanpeng Zeng, Li Zhang, and Vikas Singh. 2024. FrameQuant: Flexible Low-Bit Quantization for Transformers. *arXiv preprint arXiv:2403.06082* (2024).
- [5] Abien Fred Agarap. 2018. Deep Learning using Rectified Linear Units (ReLU). *CoRR* abs/1803.08375 (2018). arXiv:1803.08375 <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*.
- [7] Meta AI. 2024. *Llama 3.2: Revolutionizing edge AI and vision with open, customizable models*. <https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/> Accessed: 2024-9-25.
- [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 [cs.CL] <https://arxiv.org/abs/2305.13245>
- [9] 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 preprint arXiv:2311.16867* (2023).
- [10] Guilherme FCF Almeida, José Luiz Nunes, Neele Engelmann, Alex Wiegmann, and Marcelo de Araújo. 2024. Exploring the psychology of LLMs' moral and legal reasoning. *Artificial Intelligence* 333 (2024), 104145.
- [11] 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.
- [12] 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. 10865–10873.
- [13] 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 preprint arXiv:2305.10403* (2023).
- [14] AI Anthropic. 2024. The claudie 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card* 1 (2024).
- [15] Viraat Aryabumi, Yixuan Su, Raymond Ma, Adrien Morisot, Ivan Zhang, Aycı̄ Locatelli, Marzieh Fadaee, Ahmet Üstün, and Sara Hooker. 2024. To Code, or Not To Code? Exploring Impact of Code in Pre-training. *arXiv preprint arXiv:2408.10914* (2024).
- [16] 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*. <https://openreview.net/forum?id=hSyW5go0v8>
- [17] Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefer, and James Hensman. 2024. SliceGPT: Compress Large Language Models by Deleting Rows and Columns. In *The Twelfth International Conference on Learning Representations*. <https://openreview.net/forum?id=vXkardq6db>

- [18] 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 preprint arXiv:2108.07732* (2021).
- [19] 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 2023*. 967–976.
- [20] 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 preprint arXiv:2310.10631* (2023).
- [21] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen Technical Report. *arXiv:2309.16609* [cs.CL] <https://arxiv.org/abs/2309.16609>
- [22] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268* (2016).
- [23] 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. 830–839.
- [24] 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 1.6 b technical report. *arXiv preprint arXiv:2402.17834* (2024).
- [25] Milan Bhan, Jean-Noel Vittaut, Nicolas Chesneau, and Marie-Jeanne Lesot. 2024. Self-AMPLIFY: Improving Small Language Models with Self Post Hoc Explanations. *arXiv preprint arXiv:2402.12038* (2024).
- [26] 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 preprint arXiv:2310.02031* (2023).
- [27] 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:2304.01373* [cs.CL] <https://arxiv.org/abs/2304.01373>
- [28] 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. 7432–7439.
- [29] 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> If you use this software, please cite it using these metadata..
- [30] 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 preprint arXiv:2403.18421* (2024).
- [31] 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 preprint arXiv:2403.14578* (2024).
- [32] Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. 2022. Inpars: Data augmentation for information retrieval using large language models. *arXiv preprint arXiv:2202.05144* (2022).
- [33] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 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. https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcba4967418bf8ac142f64a-Paper.pdf
- [34] 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.
- [35] Zheng Cai, Maosong Cao, Haojong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, et al. 2024. Internlm2 technical report. *arXiv preprint arXiv:2403.17297* (2024).
- [36] 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)*. 2633–2650.
- [37] Samuel Carreira, Tomás Marques, José Ribeiro, and Carlos Grilo. 2023. Revolutionizing Mobile Interaction: Enabling a 3 Billion Parameter GPT LLM on Mobile. *arXiv:2310.01434* [cs.CL] <https://arxiv.org/abs/2310.01434>
- [38] Wei-Cheng Chang, X Yu Felix, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. [n. d.]. Pre-training Tasks for Embedding-based Large-scale Retrieval. In *International Conference on Learning Representations*.
- [39] 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 preprint arXiv:2404.01318* (2024).
- [40] Dong Chen, Yueling 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. 11249–11257.

- [41] 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 2023*. 6805–6820.
- [42] 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 preprint arXiv:2107.03374* (2021).
- [43] Tianyi Chen, Tianyu Ding, Badal Yadav, Ilya Zharkov, and Luming Liang. 2023. Lorashear: Efficient large language model structured pruning and knowledge recovery. *arXiv preprint arXiv:2310.18356* (2023).
- [44] Wei Chen, Zhiyuan Li, and Mingyuan Ma. 2024. Octopus: On-device language model for function calling of software APIs. arXiv:2404.01549 [cs.CL] <https://arxiv.org/abs/2404.01549>
- [45] Wenhui Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks. *Transactions on Machine Learning Research* (2023). <https://openreview.net/forum?id=YfZ4ZPt8zd>
- [46] 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*. 11215–11221.
- [47] 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 preprint arXiv:2311.16079* (2023).
- [48] Zhiyu Chen, Wenhui 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*. 3697–3711.
- [49] 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*. 6279–6292.
- [50] 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 preprint arXiv:2406.11277* (2024).
- [51] 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 preprint arXiv:2406.13261* (2024).
- [52] 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)*. 35–64.
- [53] 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. <https://lmsys.org/blog/2023-03-30-vicuna/>
- [54] 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:2401.06432 [cs.LG] <https://arxiv.org/abs/2401.06432>
- [55] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research* 24, 240 (2023), 1–113.
- [56] 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*. 1478–1489.
- [57] 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. *Journal of Machine Learning Research* 25, 70 (2024), 1–53.
- [58] 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 preprint arXiv:1803.05457* (2018).
- [59] 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 preprint arXiv:2110.14168* (2021).
- [60] Nigel Collier, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Jin-Dong Kim. 2004. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*. 73–78.
- [61] Together Computer. 2023. RedPajama: an Open Dataset for Training Large Language Models. <https://github.com/togethercomputer/RedPajama-Data>
- [62] 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. <https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-lm>
- [63] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 deep learning track. CoRR abs/2102.07662 (2021). arXiv:2102.07662 <https://arxiv.org/abs/2102.07662>
- [64] Justin Cui, Wei-Lin Chiang, Ion Stoica, and Cho-Jui Hsieh. 2024. OR-Bench: An Over-Refusal Benchmark for Large Language Models. *arXiv preprint arXiv:2405.20947* (2024).
- [65] 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 preprint arXiv:2311.18580* (2023).

- [66] Luigi Daniele and Suphavadeeprasit. 2023. Amplify-Instruct: Synthetically Generated Diverse Multi-turn Conversations for efficient LLM Training. *arXiv preprint arXiv:(coming soon)* (2023). <https://huggingface.co/datasets/LDJnr/Cappybara>
- [67] Tri Dao. [n. d.]. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *The Twelfth International Conference on Learning Representations*.
- [68] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems* 35 (2022), 16344–16359.
- [69] Tri Dao and Albert Gu. 2024. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060* (2024).
- [70] Rocktim Jyoti Das, Liqun Ma, and Zhiqiang Shen. 2023. Beyond size: How gradients shape pruning decisions in large language models. *arXiv preprint arXiv:2311.04902* (2023).
- [71] 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*. 1073–1081.
- [72] 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*. 1693–1706.
- [73] 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 preprint arXiv:2312.05842* (2023).
- [74] 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.). <https://openreview.net/forum?id=dXiGWqBoxaD>
- [75] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems* 36 (2024).
- [76] 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.
- [77] 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, Volume 1 (Long and Short Papers)*. 4171–4186.
- [78] 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 (2023).
- [79] 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 preprint arXiv:2305.14233* (2023).
- [80] Tinghe Ding. 2024. MobileAgent: enhancing mobile control via human-machine interaction and SOP integration. *arXiv:2401.04124* [cs.HC] <https://arxiv.org/abs/2401.04124>
- [81] Ricardo Dominguez-Olmedo, Moritz Hardt, and Celestine Mendler-Dünner. 2023. Questioning the survey responses of large language models. *arXiv preprint arXiv:2306.07951* (2023).
- [82] 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*. 441–451.
- [83] 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 (Volume 1: Long Papers)*. 320–335.
- [84] 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 preprint arXiv:2407.21783* (2024).
- [85] Kazuki Egashira, Mark Vero, Robin Staab, Jingxuan He, and Martin Vechev. 2024. Exploiting LLM Quantization. *arXiv preprint arXiv:2405.18137* (2024).
- [86] Ronen Eldan and Yuanzhi Li. 2023. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759* (2023).
- [87] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. 2018. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks* 107 (2018), 3–11.
- [88] 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. <https://aclanthology.org/2023.emnlp-main.230>
- [89] William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research* 23, 120 (2022), 1–39.
- [90] 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 preprint arXiv:2305.09955* (2023).

- [91] 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.
- [92] Elias Frantar, Saleh Ashkboos, Torsten Hoefer, and Dan Alistarh. 2023. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. In *The Eleventh International Conference on Learning Representations*.
- [93] 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). <https://yaofu.notion.site/How-does-GPT-Obtain-its-Ability-Tracing-Emergent-Abilities-of-Language-Models-to-their-Sources-b9a57ac0fcf74f30a1ab9e3e36fa1dc1>
- [94] Yao Fu, Hao Peng, 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.
- [95] Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal* 12, 2 (1994), 23–38.
- [96] 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 Transactions on Information Systems* 42, 1 (2023), 1–27.
- [97] 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 preprint arXiv:2101.00027* (2020).
- [98] 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 (Volume 1: Long Papers)*, 2843–2853.
- [99] 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*. <https://openreview.net/forum?id=uNrfDPMyo>
- [100] Alex Gichamba, Tewodros Kederalah Idris, Brian Ebliyau, Eric Nyberg, and Teruko Mitamura. 2024. ColBERT Retrieval and Ensemble Response Scoring for Language Model Question Answering. *arXiv:2408.10808* [cs.CL] <https://arxiv.org/abs/2408.10808>
- [101] Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. 2019. Openwebtext corpus.
- [102] 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.
- [103] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 6904–6913.
- [104] 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 preprint arXiv:2402.00838* (2024).
- [105] Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752* (2023).
- [106] 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*.
- [107] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojgan Javaheripour, 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:2306.11644* [cs.CL] <https://arxiv.org/abs/2306.11644>
- [108] Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, et al. 2024. DeepSeek-Coder: When the Large Language Model Meets Programming—The Rise of Code Intelligence. *arXiv preprint arXiv:2401.14196* (2024).
- [109] 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*.
- [110] 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* (2022).
- [111] Song Guo, Jiahang Xu, Li Lyra Zhang, and Mao Yang. 2023. Compresso: Structured pruning with collaborative prompting learns compact large language models. *arXiv preprint arXiv:2310.05015* (2023).
- [112] Zhen Guo, Peiqi Wang, Yanwei Wang, and Shangdi Yu. 2023. Improving Small Language Models on PubMedQA via Generative Data Augmentation. *arXiv:2305.07804* [cs.CL] <https://arxiv.org/abs/2305.07804>
- [113] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems* 28 (2015).
- [114] 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:2308.04711* [cs.CL] <https://arxiv.org/abs/2308.04711>
- [115] Tim Hartill, Nesan Tan, Michael Witbrock, and Patricia J. Riddle. 2023. Teaching Smaller Language Models To Generalise To Unseen Compositional Questions. *arXiv:2308.00946* [cs.CL] <https://arxiv.org/abs/2308.00946>
- [116] 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 preprint arXiv:2310.05694* (2023).
- [117] 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*. <https://openreview.net/forum?id=sE7-XhLxHA>
- [118] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654* (2020).

- [119] 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. *Knowledge-Based Systems* 256 (2022), 109835.
- [120] 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*. <https://openreview.net/forum?id=d7KBjmI3GmQ>
- [121] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* (2016).
- [122] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).
- [123] Junyuan Hong, Jinhao Duan, Chenhui Zhang, Zhangheng Li, Chulin Xie, Kelsey Lieberman, James Diffenderfer, Brian R. Bartoldson, Ajay Kumar Jaiswal, Kaidi Xu, Bhavya Kailkhura, Dan Hendrycks, Dawn Song, Zhangyang Wang, and Bo Li. 2024. Decoding Compressed Trust: Scrutinizing the Trustworthiness of Efficient LLMs Under Compression. In *Proceedings of the Forty-first International Conference on Machine Learning, ICML*. <https://openreview.net/forum?id=e3Dpq3WdMv>
- [124] Yutong Meng, Yuhao Wang, Hongcheng Liu, Yusheng Liao. 2023. XieZhi: Chinese Law Large Language Model. https://github.com/LiuHC0428/LAW_GPT.
- [125] 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.
- [126] 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 2023*. 8003–8017.
- [127] 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 preprint arXiv:2106.09685* (2021).
- [128] 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 preprint arXiv:2404.06395* (2024).
- [129] 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 preprint arXiv:2405.17849* (2024).
- [130] Feiran Huang, Zhenghang Yang, Junyi Jiang, Yuanchen Bei, Yijie Zhang, and Hao Chen. 2024. Large Language Model Interaction Simulator for Cold-Start Item Recommendation. *arXiv preprint arXiv:2402.09176* (2024).
- [131] Jiaxin Huang, Shixiang Shan Gu, Le Hou, Yuxin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. [n. d.]. Large Language Models Can Self-Improve. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.
- [132] 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 preprint arXiv:2311.05232* (2023).
- [133] 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 preprint arXiv:2402.04291* (2024).
- [134] 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. *Advances in Neural Information Processing Systems* 36 (2024).
- [135] Yangsibo Huang, Samiyak Gupta, Mengzhou Xia, Kai Li, and Danqi Chen. 2023. Catastrophic Jailbreak of Open-source LLMs via Exploiting Generation. *arXiv preprint arXiv:2310.06987* (2023).
- [136] Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. [n. d.]. Poly-encoders: Architectures and Pre-training Strategies for Fast and Accurate Multi-sentence Scoring. In *International Conference on Learning Representations*.
- [137] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation* 3, 1 (1991), 79–87.
- [138] Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sébastien 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. *Microsoft Research Blog* (2023).
- [139] 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 2023*. 15459–15469.
- [140] Ananya Harsh Jha, Tom Sherborne, Evan Pete Walsh, Dirk Groeneveld, Emma Strubell, and Iz Beltagy. 2024. Just CHOP: Embarrassingly Simple LLM Compression. *arXiv:2305.14864* [cs.CL] <https://arxiv.org/abs/2305.14864>
- [141] 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 preprint arXiv:2405.10616* (2024).
- [142] 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 2023*. 1827–1843.
- [143] 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 preprint arXiv:2310.06825* (2023).
- [144] 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*. <https://openreview.net/forum?id=9YvfRpmyw>

- [145] Di Jin, Eileen Pan, Nassim Oufattolle, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences* 11, 14 (2021), 6421.
- [146] 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.
- [147] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1601–1611.
- [148] Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. 2024. Gear: An efficient kv cache compression recipe for near-lossless generative inference of llm. *arXiv preprint arXiv:2403.05527* (2024).
- [149] 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 preprint arXiv:2001.08361* (2020).
- [150] Jeonghoon Kim, Jung Hyun Lee, Sungdong Kim, Joonsuk Park, Kang Min Yoo, Se Jung Kwon, and Dongsoo Lee. 2024. Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. *Advances in Neural Information Processing Systems* 36 (2024).
- [151] Minsoo Kim, Sihwa Lee, Janghwan Lee, Sukjin Hong, Du-Seong Chang, Wonyong Sung, and Jungwook Choi. 2024. Token-scaled logit distillation for ternary weight generative language models. *Advances in Neural Information Processing Systems* 36 (2024).
- [152] 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 preprint arXiv:2306.07629* (2023).
- [153] Yoon Kim and Alexander M Rush. 2016. Sequence-Level Knowledge Distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 1317–1327.
- [154] Young Jin Kim, Raffy Fahim, and Hany Hassan Awadalla. 2023. Mixture of Quantized Experts (MoQE): Complementary Effect of Low-bit Quantization and Robustness. *arXiv preprint arXiv:2310.02410* (2023).
- [155] Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. 2024. DistiLLM: Towards Streamlined Distillation for Large Language Models. *arXiv preprint arXiv:2402.03898* (2024).
- [156] 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 preprint arXiv:2211.15533* (2022).
- [157] Martin Krallinger, Obdulia Rabal, Saber A Akhondi, Martín Pérez Pérez, Jesús Santamaría, Gael Pérez Rodríguez, Georgios Tsatsaronis, Ander Intxaurrondo, José Antonio López, Umesh Nandal, et al. 2017. Overview of the BioCreative VI chemical-protein interaction Track. In *Proceedings of the sixth BioCreative challenge evaluation workshop*, Vol. 1. 141–146.
- [158] Divyanshu Kumar, Anurakt Kumar, Sahil Agarwal, and Prashanth Harshangi. 2024. Fine-Tuning, Quantization, and LLMs: Navigating Unintended Outcomes. *arXiv preprint arXiv:2404.04392* (2024).
- [159] Tom Kwiatkowski, Jenni Maria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics* 7 (2019), 452–466. https://doi.org/10.1162/tacl_a_00276
- [160] 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 preprint arXiv:2402.10373* (2024).
- [161] 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. *Advances in Neural Information Processing Systems* 35 (2022), 31809–31826.
- [162] 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. (2023).
- [163] 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 2024)*. 2835–2843.
- [164] Benjamin Lefauveaux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, Daniel Haziza, Luca Wehrstedt, Jeremy Reizenstein, and Grigory Sizov. 2022. xFormers: A modular and hackable Transformer modelling library. <https://github.com/facebookresearch/xformers>.
- [165] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *ArXiv e-prints* (2016), arXiv–1607.
- [166] 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. *Advances in Neural Information Processing Systems* 35 (2022), 3843–3857.
- [167] Guangyan Li, Yongqiang Tang, and Wensheng Zhang. 2024. LoRAP: Transformer Sub-Layers Deserve Differentiated Structured Compression for Large Language Models. *arXiv preprint arXiv:2404.09695* (2024).
- [168] 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* [cs.CL] <https://arxiv.org/abs/2403.18365>

- [169] 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*. 54–73.
- [170] Junyi Li, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023. HalluEval: 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, Singapore, 6449–6464. <https://doi.org/10.18653/v1/2023.emnlp-main.397>
- [171] 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* [cs.CL] <https://arxiv.org/abs/2403.20041>
- [172] Pingzhi Li, Xiaolong Jin, Yu Cheng, and Tianlong Chen. 2024. Examining Post-Training Quantization for Mixture-of-Experts: A Benchmark. *arXiv preprint arXiv:2406.08155* (2024).
- [173] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Lucioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2023. StarCoder: may the source be with you! *arXiv:2305.06161* [cs.CL] <https://arxiv.org/abs/2305.06161>
- [174] Shengrui Li, Xueting Han, and Jing Bai. 2024. Nuteprune: Efficient progressive pruning with numerous teachers for large language models. *arXiv preprint arXiv:2402.09773* (2024).
- [175] 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 preprint arXiv:2402.14845* (2024).
- [176] 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 (Volume 1: Long Papers)*. 12286–12312.
- [177] Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190* (2021).
- [178] 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* [cs.CL] <https://arxiv.org/abs/2309.05463>
- [179] 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 preprint arXiv:2310.15929* (2023).
- [180] 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.
- [181] Wing Lian, Guan Wang, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". 2023. SlimOrca: An Open Dataset of GPT-4 Augmented FLAN Reasoning Traces, with Verification. <https://huggingface.co/Open-Orca/SlimOrca>
- [182] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Alexander Cosgrove, Christopher D Manning, Christopher Re, Diana Acosta-Navas, Drew Arad Hudson, Eric Zelikman, Esin Durmus, Faisal Ladakh, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue WANG, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Andrew Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2023. Holistic Evaluation of Language Models. *Transactions on Machine Learning Research* (2023). <https://openreview.net/forum?id=iO4LZibEqW>
- [183] 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.
- [184] 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. *Proceedings of Machine Learning and Systems* 6 (2024), 87–100.
- [185] 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 (Volume 1: Long Papers)*. 3214–3252.
- [186] Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuhui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, Ramakanth Pasunuru, Sam Shleifer, Punit Singh Koura, Vishrav Chaudhary, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Zornitsa Kozareva, Mona Diab, Veselin Stoyanov, and Xian Li. 2022. Few-shot Learning with Multilingual Generative 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, Abu Dhabi, United Arab Emirates, 9019–9052. <https://doi.org/10.18653/v1/2022.emnlp-main.616>

- [187] 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 preprint arXiv:2404.07965* (2024).
- [188] Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Daya Guo, et al. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434* (2024).
- [189] Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A Smith. 2024. Tuning language models by proxy. *arXiv preprint arXiv:2401.08565* (2024).
- [190] Jiahuo Liu, Zheyan Shen, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. 2021. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624* (2021).
- [191] 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*. 452–461.
- [192] 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 V*. 1. 388–393.
- [193] 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 preprint arXiv:2312.15685* (2023).
- [194] Yinpeng Liu, Jiawei Liu, Xiang Shi, Qikai Cheng, and Wei Lu. 2024. Let's Learn Step by Step: Enhancing In-Context Learning Ability with Curricular Learning. *arXiv preprint arXiv:2402.10738* (2024).
- [195] 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 preprint arXiv:1907.11692* (2019).
- [196] 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 preprint arXiv:2403.06840* (2024).
- [197] Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyriolidis, and Anshumali Shrivastava. 2024. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems* 36 (2024).
- [198] 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 preprint arXiv:2305.17888* (2023).
- [199] 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 preprint arXiv:2402.14905* (2024).
- [200] 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 2024*. 11065–11082.
- [201] 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.
- [202] Shayne Longpre, Robert Mahari, Anthony Chen, Naana Obeng-Marnu, Damien Sileo, William Brannon, Niklas Muennighoff, Nathan Khazam, Jad Kabbara, Kartik Perisetta, et al. 2023. The data provenance initiative: A large scale audit of dataset licensing & attribution in ai. *arXiv preprint arXiv:2310.16787* (2023).
- [203] 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 preprint arXiv:2402.19173* (2024).
- [204] 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*. 2645–2652.
- [205] Renqian Luo, Lai 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.
- [206] 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 preprint arXiv:2402.17764* (2024).
- [207] Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems* 36 (2023), 21702–21720.
- [208] 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*. 5303–5315.
- [209] 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 preprint arXiv:2303.08559* (2023).
- [210] 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.
- [211] 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*. <https://openreview.net/forum?id=KIPJKST4gw>
- [212] Dakota Mahan, Ryan Carlow, Louis Castricato, Nathan Cooper, and Christian Laforte. [n. d.]. Stable Beluga models. [<https://huggingface.co/stabilityai/StableBeluga2>] [<https://huggingface.co/stabilityai/StableBeluga2>]

- [213] 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 preprint arXiv:2403.14852* (2024).
- [214] Sachin Mehta, Mohammad Hossein Sekhavat, Qingqing Cao, Maxwell Horton, Yanzi Jin, Chenfan Sun, Seyed Iman Mirzadeh, Mahyar Najibi, Dmitry Belenko, Peter Zatoulakal, et al. 2024. Openelm: An efficient language model family with open training and inference framework. In *Workshop on Efficient Systems for Foundation Models II@ ICML2024*.
- [215] Dheeraj Mekala, Alex Nguyen, and Jingbo Shang. 2024. Smaller language models are capable of selecting instruction-tuning training data for larger language models. *arXiv preprint arXiv:2402.10430* (2024).
- [216] 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 preprint arXiv:2403.03853* (2024).
- [217] Go Min-su. 2024. Deep Learning Bible - 8. Large Language Models. WikiDocs. <https://wikidocs.net/237419>
- [218] Eric Mitchell, Rafael Rafailev, 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*. <https://openreview.net/forum?id=Eo7kv0slr>
- [219] Arindam Mitra, Luciano Del Corro, Shweta 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 preprint arXiv:2311.11045* (2023).
- [220] Lingbo Mo, Boshi Wang, Muhan 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 (Volume 1: Long Papers)*. 2775–2792.
- [221] 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*. <https://openreview.net/forum?id=t9dWHpGkPj>
- [222] Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive Text Embedding Benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2014–2037. <https://doi.org/10.18653/v1/2023.eacl-main.148>
- [223] 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 preprint arXiv:2306.02707* (2023).
- [224] 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 preprint arXiv:2407.14679* (2024).
- [225] Rithesh Murthy, Liangwei Yang, Juntao Tan, Tulika Manoj Awalgona, 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. MobileAI Bench: Benchmarking LLMs and LMMs for On-Device Use Cases. *arXiv:2406.10290* [cs.CL] <https://arxiv.org/abs/2406.10290>
- [226] 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 preprint arXiv:2406.05364* (2024).
- [227] 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*. 1–13.
- [228] Piotr Nawrot, Adrian Łaniczki, Marcin Chochowski, David Tarjan, and Edoardo Ponti. 2024. Dynamic Memory Compression: Retrofitting LLMs for Accelerated Inference. In *Forty-first International Conference on Machine Learning*. <https://openreview.net/forum?id=tDRYrAkOB7>
- [229] 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 2024)*. 4226–4237.
- [230] Tuan Dung Nguyen, Yuan-Sen Ting, Ioana Ciucu, Charles O'Neill, Ze-Chang Sun, Maja Jabłńska, 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*. 49–55.
- [231] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).
- [232] A Noorian. 2024. A BERT-based sequential POI recommender system in social media. *Computer Standards & Interfaces* 87 (2024), 103766.
- [233] Benjamin Nye, Junyi Jessy Li, Roma Patel, Yinfei Yang, Iain J Marshall, Ani Nenkova, and Byron C Wallace. 2018. A corpus with multi-level annotations of patients, interventions and outcomes to support language processing for medical literature. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, Vol. 2018. NIH Public Access, 197.
- [234] OpenAI. 2024. Hello GPT-4o. <https://openai.com/index/hello-gpt-4o/> Accessed: 2024-5-13.
- [235] 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. *Advances in neural information processing systems* 35 (2022), 27730–27744.
- [236] Shankar Padmanabhan, Yasumasa Onoe, Michael Zhang, Greg Durrett, and Eunsol Choi. 2024. Propagating knowledge updates to lms through distillation. *Advances in Neural Information Processing Systems* 36 (2024).
- [237] Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. MedMCQA : A Large-scale Multi-Subject Multi-Choice Dataset for Medical domain Question Answering. *arXiv:2203.14371* [cs.CL] <https://arxiv.org/abs/2203.14371>
- [238] 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*. <https://openreview.net/forum?id=jKHmjlpViu>

- [239] 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 preprint arXiv:2306.01116* (2023).
- [240] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277* (2023).
- [241] Zhiyuan Peng, Xuyang Wu, Qifan Wang, and Yi Fang. 2023. Soft prompt tuning for augmenting dense retrieval with large language models. *arXiv preprint arXiv:2307.08303* (2023).
- [242] Ethan Perez, Sam Ringer, Kamile Lukosiute, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, Andy Jones, Anna Chen, Benjamin Mann, Brian Israel, Bryan Seethor, Cameron McKinnon, Christopher Olah, Da Yan, Daniela Amodei, Dario Amodei, Dawn Drain, Dustin Li, Eli Tran-Johnson, Guro Khundadze, Jackson Kernion, James Landis, Jamie Kerr, Jared Mueller, Jeeyoon Hyun, Joshua Landau, Kamal Ndousse, Landon Goldberg, Liane Lovitt, Martin Lucas, Michael Sellitto, Miranda Zhang, Neerav Kingsland, Nelson Elhage, Nicholas Joseph, Noemi Mercado, Nova DasSarma, Oliver Rausch, Robin Larson, Sam McCandlish, Scott Johnston, Shauna Kravec, Sheer El Showk, Tamera Lanham, Timothy Telleen-Lawton, Tom Brown, Tom Henighan, Tristan Hume, Yuntao Bai, Zac Hatfield-Dodds, Jack Clark, Samuel R. Bowman, Amanda Askell, Roger Grosse, Danny Hernandez, Deep Ganguli, Evan Hubinger, Nicholas Schiefer, and Jared Kaplan. 2023. Discovering Language Model Behaviors with Model-Written Evaluations. In *Findings of the Association for Computational Linguistics: ACL 2023*. 13387–13434.
- [243] Pascal Pfeiffer, Philipp Singer, Yauhen Babakhin, Gabor Fodor, Nischay Dhankhar, and Sri Satish Ambati. 2024. H2O-Danube3 Technical Report. *arXiv preprint arXiv:2407.09276* (2024).
- [244] Karmvir Singh Phogat, Sai Akhil Puranam, Sridhar Dasaratha, Chetan Harsha, and Shashishekhar Ramakrishna. 2024. Fine-tuning Smaller Language Models for Question Answering over Financial Documents. *arXiv:2408.12337* [cs.CL] <https://arxiv.org/abs/2408.12337>
- [245] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems 5* (2023), 606–624.
- [246] 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*. <https://openreview.net/forum?id=R8sQPPGCv0>
- [247] 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 preprint arXiv:2311.12275* (2023).
- [248] 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.
- [249] Rafael Rafailev, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems 36* (2024).
- [250] 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. *Journal of machine learning research* 21, 140 (2020), 1–67.
- [251] 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* [cs.CL] <https://arxiv.org/abs/2310.03971>
- [252] 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.
- [253] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2383–2392.
- [254] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-Context Retrieval-Augmented Language Models. *Transactions of the Association for Computational Linguistics* 11 (2023), 1316–1331.
- [255] 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 (Volume 1: Long Papers)*. Association for Computational Linguistics, 15762–15782. <https://aclanthology.org/2023.acl-long.878>
- [256] Al Mamunur Rashid, George Karypis, and John Riedl. 2008. Learning preferences of new users in recommender systems: an information theoretic approach. *Acm Sigkdd Explorations Newsletter* 10, 2 (2008), 90–100.
- [257] Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2024. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems 36* (2024).
- [258] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.
- [259] Jihyeon Roh, Minho Kim, and Kyoungman Bae. 2024. Towards a small language model powered chain-of-reasoning for open-domain question answering. *ETRI Journal* 46, 1 (2024), 11–21. <https://doi.org/10.4218/etrij.2023-0355> arXiv:<https://onlinelibrary.wiley.com/doi/pdf/10.4218/etrij.2023-0355>
- [260] Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémie Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950* (2023).
- [261] Caitlin Sadowski and Greg Levin. 2007. Simhash: Hash-based similarity detection.

- [262] 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.
- [263] Scott Sanner, Krisztian Balog, Filip Radlinski, Ben Wedin, and Lucas Dixon. 2023. Large language models are competitive near cold-start recommenders for language-and item-based preferences. In *Proceedings of the 17th ACM conference on recommender systems*. 890–896.
- [264] 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. *Advances in Neural Information Processing Systems* 36 (2024).
- [265] Rico Sennrich, Jannis Vamvas, and Alireza Mohammadshahi. 2023. Mitigating Hallucinations and Off-target Machine Translation with Source-Contrastive and Language-Contrastive Decoding. *arXiv preprint arXiv:2309.07098* (2023).
- [266] Zeyang Sha and Yang Zhang. 2024. Prompt stealing attacks against large language models. *arXiv preprint arXiv:2402.12959* (2024).
- [267] Yuzhang Shang, Zhihang Yuan, Qiang Wu, and Zhen Dong. 2023. PB-LLM: Partially Binarized Large Language Models. *arXiv:2310.00034 [cs.LG]* <https://arxiv.org/abs/2310.00034>
- [268] 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.
- [269] 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 preprint arXiv:2310.13548* (2023).
- [270] Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150* (2019).
- [271] Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202* (2020).
- [272] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuhuan 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.
- [273] 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*. 1893–1903.
- [274] Mohammad Shoeybi, Mostafa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-Lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053* (2019).
- [275] Parshin Shojaei, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, and Chandan K Reddy. 2024. Llm-sr: Scientific equation discovery via programming with large language models. *arXiv preprint arXiv:2404.18400* (2024).
- [276] 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. *Advances in Neural Information Processing Systems* 35 (2022), 14274–14289.
- [277] Karan Singh, 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.
- [278] 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. <https://cerebras.ai/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama>. <https://huggingface.co/datasets/cerebras/SlimPajama-627B>
- [279] Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Author, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Harsh Jha, Sachin Kumar, Li Lucy, Xinxin Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. 2024. Dolma: an Open Corpus of Three Trillion Tokens for Language Model Pretraining Research. *arXiv preprint* (2024).
- [280] Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, and Denny Zhou. 2021. Fast WordPiece Tokenization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2089–2103.
- [281] 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 preprint arXiv:2307.03744* (2023).
- [282] 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.
- [283] 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.
- [284] Lichao Sun, Yue Huang, Haoran Wang, Siyuan Wu, Qihui Zhang, Chujiie Gao, Yixin Huang, Wenhan Lyu, Yixuan Zhang, Xiner Li, et al. 2024. Trustllm: Trustworthiness in large language models. *arXiv preprint arXiv:2401.05561* (2024).
- [285] 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*.
- [286] 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 preprint arXiv:2004.02984* (2020).
- [287] Swabha Swamydipta, 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.

- [288] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge. arXiv:1811.00937 [cs.CL] <https://arxiv.org/abs/1811.00937>
- [289] 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 preprint arXiv:2402.12052* (2024).
- [290] 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 preprint arXiv:2402.14373* (2024).
- [291] 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 preprint arXiv:2402.02791* (2024).
- [292] Rohan Taori, Ishaa Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford Alpaca: An Instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca.
- [293] 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 preprint arXiv:2211.09085* (2022).
- [294] CodeGEMMA Team. 2024. Codegemma: Open code models based on gemma. *arXiv preprint arXiv:2406.11409* (2024).
- [295] 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 preprint arXiv:2403.08295* (2024).
- [296] Gemma Team, Morgane Rivière, 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 preprint arXiv:2408.00118* (2024).
- [297] TensorOpera Team. 2024. *TensorOpera Unveils Fox Foundation Model: A Pioneering Small Language Model (SLM) for Cloud and Edge*. <https://blog.tensoropera.ai/tensoropera-unveils-fox-foundation-model-a-pioneering-open-source-slm-leading-the-way-against-tech-giants/> Accessed: 2024-6-13.
- [298] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.
- [299] 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 preprint arXiv:2402.16840* (2024).
- [300] 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 preprint arXiv:2402.12253* (2024).
- [301] 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 preprint arXiv:2302.13971* (2023).
- [302] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaï, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).
- [303] Jonathan Tow, Marco Bellagente, Dakota Mahan, and Carlos Riquelme. [n. d.]. StableLM 3B 4E1T. <https://huggingface.co/stabilityai/stablelm-3b-4e1t> (<https://huggingface.co/stabilityai/stablelm-3b-4e1t>)
- [304] Trieu H Trinh and Quoc V Le. 2018. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847* (2018).
- [305] Adina Trufinescu. 2024. Discover the New Multi-Lingual High-Quality Phi-3.5 SLMs. <https://techcommunity.microsoft.com/t5/ai-azure-ai-services-blog/discover-the-new-multi-lingual-high-quality-phi-3-5-slms/ba-p/4225280>.
- [306] Dennis Ulmer, Martin Gubri, Hwaran Lee, Sangdoo Yun, and Seong Joon Oh. 2024. Calibrating Large Language Models Using Their Generations Only. *arXiv preprint arXiv:2403.05973* (2024).
- [307] Sander Van Der Linden. 2022. Misinformation: susceptibility, spread, and interventions to immunize the public. *Nature medicine* 28, 3 (2022), 460–467.
- [308] A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).
- [309] 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.
- [310] 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.
- [311] 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.
- [312] Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, Sang T. Truong, Simran Arora, Mantas Mazeika, Dan Hendrycks, Zinan Lin, Yu Cheng, Sanmi Koyejo, Dawn Song, and Bo Li. 2023. DecodingTrust: A Comprehensive Assessment of Trustworthiness in GPT Models. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.
- [313] 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 preprint arXiv:2111.02840* (2021).
- [314] Guan Wang, Sijie Cheng, Qiyi Yu, and Changling Liu. 2023. OpenLLMs: Less is More for Open-source Models. <https://doi.org/10.5281/zenodo.8105775>
- [315] 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*. <https://openreview.net/forum?id=AOJyfhWYHf>

- [316] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huajie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. 2023. Bitnet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453* (2023).
- [317] Jianling Wang, Haokai Lu, James Caverlee, Ed H Chi, and Minmin Chen. 2024. Large Language Models as Data Augmenters for Cold-Start Item Recommendation. In *Companion Proceedings of the ACM on Web Conference 2024*. 726–729.
- [318] 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 (Volume 1: Long Papers)*. Association for Computational Linguistics, 11897–11916. <https://doi.org/10.18653/v1/2024.acl-long.642>
- [319] 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 (Volume 1: Long Papers)*. 5546–5558.
- [320] Wenxiao Wang, Wei Chen, Yicong Luo, Yongliu Long, Zhengkai Lin, Liy Zhang, Binbin Lin, Deng Cai, and Xiaofei He. 2024. Model compression and efficient inference for large language models: A survey. *arXiv preprint arXiv:2402.09748* (2024).
- [321] 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 [cs.CL]* <https://arxiv.org/abs/2002.10957>
- [322] Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramanian, 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*. <https://openreview.net/forum?id=bqJEGioLr>
- [323] 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 2024*. Association for Computational Linguistics, 896–911. <https://aclanthology.org/2024.findings-eacl.61>
- [324] 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 2023*. 10303–10315.
- [325] Yubo Wang, Xueguang Ma, and Wenhui Chen. 2023. Augmenting black-box llms with medical textbooks for clinical question answering. *arXiv preprint arXiv:2309.02233* (2023).
- [326] 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*. Association for Computing Machinery, 4100–4109. <https://doi.org/10.1145/3534678.3539073>
- [327] 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*. 3876–3887.
- [328] Yuqing Wang and Yun Zhao. 2024. RUPBench: Benchmarking Reasoning Under Perturbations for Robustness Evaluation in Large Language Models. *arXiv preprint arXiv:2406.11020* (2024).
- [329] 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*. <https://openreview.net/forum?id=gEZrGCozdqR>
- [330] Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2023. Magicoder: Source code is all you need. *arXiv preprint arXiv:2312.02120* (2023).
- [331] 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 2021*. Association for Computational Linguistics, 2447–2469. <https://doi.org/10.18653/v1/2021.findings-emnlp.210>
- [332] 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.
- [333] Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaquin Zhang, and Yunxin Liu. 2024. AutoDroid: LLM-powered Task Automation in Android. *arXiv:2308.15272 [cs.AI]* <https://arxiv.org/abs/2308.15272>
- [334] 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*. <https://openreview.net/forum?id=GLGYYqPwvj>
- [335] 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*. 1652–1656.
- [336] Minghao Wu, Abdul Waheed, Chiyu Zhang, Muhammad Abdul-Mageed, and Alham Fikri Aji. 2024. 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, St. Julian's, Malta, 944–964. <https://aclanthology.org/2024.eacl-long.57>
- [337] Qinzhuo Wu, Weikai Xu, Wei Liu, Tao Tan, Jianfeng Liu, Ang Li, Jian Luan, Bin Wang, and Shuo Shang. 2024. MobileVLM: A Vision-Language Model for Better Intra-and Inter-UI Understanding. *arXiv preprint arXiv:2409.14818* (2024).
- [338] 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*. 3566–3575.
- [339] Zhuofeng Wu, He Bai, Aonan Zhang, Jiatao Gu, VG Vydiswaran, Navdeep Jaitly, and Yizhe Zhang. 2024. Divide-or-Conquer? Which Part Should You Distill Your LLM? *arXiv preprint arXiv:2402.15000* (2024).

- [340] 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 preprint arXiv:2403.01969* (2024).
- [341] 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*. <https://openreview.net/forum?id=09iOdaeOzp>
- [342] 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.
- [343] 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 preprint arXiv:2406.14598* (2024).
- [344] 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 preprint arXiv:2308.13565* (2023).
- [345] 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 preprint arXiv:2404.08517* (2024).
- [346] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Dixin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244* (2023).
- [347] 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 preprint arXiv:2305.08848* (2023).
- [348] 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 \[cs.NI\]](https://arxiv.org/abs/2309.04255) <https://arxiv.org/abs/2309.04255>
- [349] Weijia Xu, Sweta Agrawal, Eleftheria Briakou, Marianna Martindale, and Marine Carpuat. 2023. Understanding and Detecting Hallucinations in Neural Machine Translation via Model Introspection. *Transactions of the Association for Computational Linguistics* 11 (2023), 546–564.
- [350] 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 preprint arXiv:2402.11295* (2024).
- [351] Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884* (2024).
- [352] An Yang, Baosong Yang, Bin Yuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671* (2024).
- [353] 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 preprint arXiv:2407.01885* (2024).
- [354] 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*. <https://openreview.net/forum?id=v3XXtxWKi6>
- [355] Kailai Yang, Tianlin Zhang, Ziyan Kuang, Qianqian Xie, Jimin Huang, and Sophia Ananiadou. 2024. MentalLLaMA: interpretable mental health analysis on social media with large language models. In *Proceedings of the ACM on Web Conference 2024*. 4489–4500.
- [356] 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 2023*. Association for Computational Linguistics, Toronto, Canada, 12731–12750. <https://doi.org/10.18653/v1/2023.findings-acl.806>
- [357] 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*. <https://openreview.net/forum?id=bAMPOUf227>
- [358] Yifei Yang, Zouying Cao, and Hai Zhao. 2024. Laco: Large language model pruning via layer collapse. *arXiv preprint arXiv:2402.11187* (2024).
- [359] Yizhe Yang, Huashan Sun, Jiawei Li, Runheng Liu, Yinghao Li, Yuhang Liu, Heyan Huang, and Yang Gao. 2023. Mindlm: Pre-training lightweight large language model from scratch, evaluations and domain applications. *arXiv preprint arXiv:2310.15777* (2023).
- [360] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600* (2018).
- [361] 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 preprint arXiv:2402.11801* (2024).
- [362] 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 preprint arXiv:2106.13474* (2021), 460–470.
- [363] Mert Yazan, Suzan Verberne, and Frederik Situmeang. 2024. The Impact of Quantization on Retrieval-Augmented Generation: An Analysis of Small LLMs. *arXiv preprint arXiv:2406.10251* (2024).
- [364] 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 \[cs.LG\]](https://arxiv.org/abs/2308.14352) <https://arxiv.org/abs/2308.14352>
- [365] Wangsong Yin, Mengwei Xu, Yuanchun Li, and Xuanzhe Liu. 2024. LLM as a System Service on Mobile Devices. [arXiv:2403.11805 \[cs.OS\]](https://arxiv.org/abs/2403.11805) <https://arxiv.org/abs/2403.11805>
- [366] 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*. <https://openreview.net/forum?id=N8N0hgNDRt>

- [367] 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 preprint arXiv:2407.02485* (2024).
- [368] Zhongzhi Yu, Zheng Wang, Yuhua 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 preprint arXiv:2406.15758* (2024).
- [369] 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.
- [370] Xiaohan Yuan, Jinfeng Li, Dongxia Wang, Yuefeng Chen, Xiaofeng Mao, Longtao Huang, Hui Xue, Wenhui Wang, Kui Ren, and Jingyi Wang. 2024. S-Eval: Automatic and Adaptive Test Generation for Benchmarking Safety Evaluation of Large Language Models. *arXiv preprint arXiv:2405.14191* (2024).
- [371] Shengbin Yue, Wei Chen, Siyuan Wang, Bingxuan Li, Chenchen Shen, Shujun Liu, Yuxuan Zhou, Yao Xiao, Song Yun, Wei Lin, et al. 2023. Disc-lawlm: Fine-tuning large language models for intelligent legal services. *arXiv preprint arXiv:2309.11325* (2023).
- [372] 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*. 4791–4800.
- [373] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. *Advances in neural information processing systems* 32 (2019).
- [374] Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Advances in Neural Information Processing Systems* 32 (2019).
- [375] Cheng Zhang, Jianyi Cheng, George A Constantinides, and Yiren Zhao. 2024. LQER: Low-Rank Quantization Error Reconstruction for LLMs. *arXiv preprint arXiv:2402.02446* (2024).
- [376] Collin Zhang, John X Morris, and Vitaly Shmatikov. 2024. Extracting Prompts by Inverting LLM Outputs. *arXiv preprint arXiv:2405.15012* (2024).
- [377] Chen Zhang, Dawei Song, Zheyu Ye, and Yan Gao. 2023. Towards the law of capacity gap in distilling language models. *arXiv preprint arXiv:2311.07052* (2023).
- [378] 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 preprint arXiv:2401.07950* (2024).
- [379] 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 preprint arXiv:2402.06852* (2024).
- [380] 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 preprint arXiv:2403.03129* (2024).
- [381] 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 preprint arXiv:2305.18403* (2023).
- [382] Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. TinyLlama: An Open-Source Small Language Model. *arXiv:2401.02385* [cs.CL] <https://arxiv.org/abs/2401.02385>
- [383] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuhui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068* (2022).
- [384] 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*. 2612–2616.
- [385] 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*.
- [386] Yiming Zhang, Nicholas Carlini, and Daphne Ippolito. 2024. Effective Prompt Extraction from Language Models. *arXiv:2307.06865* [cs.CL] <https://arxiv.org/abs/2307.06865>
- [387] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. 2024. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems* 36 (2024).
- [388] 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*.
- [389] 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* [cs.LG] <https://arxiv.org/abs/2312.00388>
- [390] 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 preprint arXiv:2403.01136* (2024).
- [391] 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 preprint arXiv:2405.15924* (2024).
- [392] 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 preprint arXiv:2306.16564* (2023).
- [393] Youpeng Zhao, Ming Lin, Huadong Tang, Qiang Wu, and Jun Wang. 2024. Merino: Entropy-driven Design for Generative Language Models on IoT Devices. *arXiv:2403.07921* [cs.LG] <https://arxiv.org/abs/2403.07921>

- [394] 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 preprint arXiv:2202.13876* (2022).
- [395] 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 (Volume 1: Long Papers)*. 3277–3287.
- [396] 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 preprint arXiv:2408.03533* (2024).
- [397] 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 preprint arXiv:2306.04528* (2023).
- [398] Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2023. A survey on model compression for large language models. *arXiv preprint arXiv:2308.07633* (2023).
- [399] 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* [cs.CL] <https://arxiv.org/abs/2308.11807>
- [400] Yuanyuan Zhuang and Jaekyeong Kim. 2021. A bert-based multi-criteria recommender system for hotel promotion management. *Sustainability* 13, 14 (2021), 8039.
- [401] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043* (2023).
- [402] 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 Transactions on the Web* 17, 1 (2022), 1–36.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009