Domain Specialization as the Key to Make Large Language Models Disruptive: A Comprehensive Survey

CHEN LING, Emory University, USA and NEC Labs America, USA

XUJIANG ZHAO*†, NEC Labs America, USA

JIAYING LU*, Emory University, USA

CHENGYUAN DENG*, NEC Labs America, USA and Rutgers University, USA

CAN ZHENG*, NEC Labs America, USA and University of Pittsburgh, USA

JUNXIANG WANG*, NEC Labs America, USA

TANMOY CHOWDHURY* and YUN LI, George Mason University, USA

HEJIE CUI, Emory University, USA

XUCHAO ZHANG, Microsoft, USA

TIANJIAO ZHAO, AMIT PANALKAR, DHAGASH MEHTA, STEFANO PASQUALI, Blackrock, Inc., USA

WEI CHENG, HAOYU WANG, YANCHI LIU, ZHENGZHANG CHEN, NEC Labs America, USA

HAIFENG CHEN, CHRIS WHITE, NEC Labs America, USA

QUANQUAN GU, University of California, Los Angeles, USA

JIAN PEI, Duke University, USA

CARL YANG and LIANG ZHAO†, Emory University, USA

Large language models (LLMs) have significantly advanced the field of natural language processing (NLP), providing a highly useful, task-agnostic foundation for a wide range of applications. However, directly applying LLMs to solve sophisticated problems in specific domains meets many hurdles, caused by the heterogeneity of domain data, the sophistication of domain knowledge, the uniqueness of domain objectives, and the diversity of the constraints (e.g., various social norms, cultural conformity, religious beliefs, and ethical standards in the domain applications). Domain specification techniques are key to make large language models disruptive in many

---

*Equal Contribution.
†Corresponding author.

Authors' addresses: Chen Ling, chen.ling@emory.edu, Emory University, Atlanta, GA, USA and NEC Labs America, Princeton, NJ, USA; Xujiang Zhao, xuzhao@nec-labs.com, NEC Labs America, Princeton, NJ, USA; Jiaying Lu, jiaying.lu@emory.edu, Emory University, Atlanta, GA, USA; Chengyuan Deng, cd751@rutgers.edu, NEC Labs America, Princeton, NJ, USA and Rutgers University, New Brunswick, NJ, USA; Can Zheng, caz51@pitt.edu, NEC Labs America, Princeton, NJ, USA and University of Pittsburgh, Pittsburgh, PA, USA; Junxiang Wang, junwang@nec-labs.com, NEC Labs America, Princeton, NJ, USA; Tanmoy Chowdhury, tchowdh6@gmu.edu; Yun Li, yli38@gmu.edu, George Mason University, Fairfax, VA, USA; Hejie Cui, hejie.cui@emory.edu, Emory University, Atlanta, GA, USA; Xuchao Zhang, xuchaozhang@microsoft.com, Microsoft, Redmond, WA, USA; Tianjiao Zhao, Amit Panalkar, Dhagash Mehta, Stefano Pasquali, Blackrock, Inc., Atlanta, GA, USA; Wei Cheng, Haoyu Wang, Yanchi Liu, Zhengzhang Chen, NEC Labs America, Princeton, NJ, USA; Haifeng Chen, Chris White, NEC Labs America, Princeton, NJ, USA; Quanquan Gu, qgu@ucla.edu, University of California, Los Angeles, Los Angeles, CA, USA; Jian Pei, j.pei@duke.edu, Duke University, Durham, USA, NC; Carl Yang, j.carlyang@emory.edu; Liang Zhao, liang.zhao@emory.edu, Emory University, Atlanta, USA, GA.

---

applications. Specifically, to solve these hurdles, there has been a notable increase in research and practices conducted in recent years on the domain specialization of LLMs. This emerging field of study, with its substantial potential for impact, necessitates a comprehensive and systematic review to better summarize and guide ongoing work in this area. In this article, we present a comprehensive survey on domain specification techniques for large language models, an emerging direction critical for large language model applications. First, we propose a systematic taxonomy that categorizes the LLM domain-specialization techniques based on the accessibility to LLMs and summarizes the framework for all the subcategories as well as their relations and differences to each other. Second, we present an extensive taxonomy of critical application domains that can benefit dramatically from specialized LLMs, discussing their practical significance and open challenges. Last, we offer our insights into the current research status and future trends in this area.

Additional Key Words and Phrases: Large Language Models, Natural Language Processing, Domain Specialization

1 INTRODUCTION

The evolution of natural language processing (NLP) and artificial intelligence (AI) models has witnessed a remarkable trajectory, beginning with the rule-based systems of the 1950s and 1960s, transitioning to statistical models in the 1990s, followed by the emergence of neural networks in the 2010s. Owing to the success of self-attention and Transformer-based neural network architecture [152], Pre-trained Language Models (PLMs) emerged and swiftly gained popularity in the late 2010s due to their ability to learn universal language representations from large-scale data in an unsupervised manner, which can be beneficial for many downstream NLP tasks such as commonsense reasoning [173], multiple-choice question answering [131], and story generation [13], while avoiding training new models from scratch. In the last few years, with the fast growth of large corpus and hardware capacities, researchers have found scaling up model and training data can continuously improve the model capacity, following the scaling law [65], eventually resulting in Large Language Models (LLMs) [166], such as GPT-3 [11] (175B parameters), PaLM [20] (540B parameters), and LLaMA [149] (65B parameters). LLMs, significantly outperforming smaller models in understanding and generating human-like text, have emerged as a promising AI research trend. Their potential to revolutionize natural and social sciences through efficient literature analysis, novel hypothesis generation, and complex data interpretation could accelerate research, enhance the discovery process, and facilitate interdisciplinary collaboration.

While LLMs hold great promise as general task solvers, effectively extending their functionality beyond mere "chatbot" roles poses significant challenges. This has led to the emergence of "domain specialization of LLMs". Specifically, **domain specialization** of Large Language Models (LLMs) is defined as *the process of customizing general-purpose LLMs according to specific domain contextual data, augmented by domain-specific knowledge, optimized by the domain's objective, and regulated by domain-specific constraints.* This shift towards domain specialization of LLMs is motivated by several compelling reasons. First, there are significant differences in conversation and language styles in different fields, roles, and tasks ranging from medical prescriptions to legal sentences, to online chatting, etc. The acquisition of such capabilities and experience even require human beings many years of training, a lot of which are hands-on and proprietary. Moreover, different fields, institutions, and teams have their own "business models" about which response will maximize their own utility function for their tasks, which is not directly replaceable by a single general-purpose LLMs solver with no customization. More importantly, the requirement of domain knowledge for professional-level

usage also need to be very in-depth, in-real-time, and accurate, none of which can be easily achieved by pre-trained LLMs. Many domain knowledge resources are proprietary assets and core competitiveness of the organizations that can never be leaked to general-purpose LLMs. Last but not the least, languages are constrained by social norms, cultural conformity, religious beliefs, legal requirements, and ethical practice, all of which are changing parameters in different locations, countries, populations, races, communities, etc., which make general-purpose LLMs impossible to be a one-fits-all solver without any customization.

Domain Specialization of LLMs is a critical yet challenging problem that requires inventing and integrating effective techniques to address the serious challenges. Particularly, there are three significant challenges.

Challenge 1: Difficulty keeping an LLM updated with the latest knowledge. The power of LLMs is attributed mainly to their massive training corpus. Yet, it also indicates LLMs tend to have a knowledge cut-off and lack sufficient access to the latest information, events, or discoveries. In many specialized domains, new discoveries, regulations, and best practices continuously emerge, making it difficult for LLMs to stay up-to-date. For instance, more than 30 thousand mainstream news articles are published every day [157]. For social media analysis and fact-checking, LLMs may not handle them since the knowledge extracted from the training corpus is offline. This indicates that regular re-training or continuous learning mechanisms are required to maintain LLMs' relevance and accuracy in these dynamic fields. However, ensuring the model freshness can be resource-intensive, as it necessitates continuous high-quality and up-to-date data collection, processing, and computationally intensive model re-training.

Challenge 2: Difficulty in learning all specialized knowledge of different domains in one LLM. LLMs, by default, possess general knowledge across a wide range of topics and may have seen and obtained specific knowledge for most domains. However, more popular or widely-discussed topics may be over-represented, while very domain-specific topics can usually be under-represented, which makes it difficult to be effectively learned for domain-specific tasks. In addition, domain-specific tasks often involve complex concepts, specialized terminology, and intricate relationships between entities. Without proper guidance, LLMs may generate plausible-sounding but inconsistent answers to similar queries (i.e., LLM's hallucination) or slightly rephrased questions [5]. This issue arises because LLMs are designed to predict the most likely word sequences based on the input rather than providing a definitive answer based on a structured knowledge base. Researchers have found users can guide the model to produce more relevant, accurate, and task-specific responses, enhancing the overall utility and effectiveness of AI systems across numerous domains by providing LLMs with a few task-specific demonstrations [166]. Nevertheless, providing LLMs with adequate demonstrations is not trivial since user instructions can often be vague, incomplete, or ambiguous, making it difficult to discern the intended meaning or desired outcome. Let alone LLMs tend to have a finite context window, typically determined by the maximum token length they can process (e.g., ChatGPT can only handle 4097 tokens).

Challenge 3: Intensive model and computational complexity required for downstream task learning. To better adapt to specific domain applications, downstream task learning is historically a commonly used practice to specialize language models. However, different from traditional language models, adapting an LLM to downstream tasks needs vast amounts of high-quality, task-specific data. Acquiring, cleaning, and pre-processing such data can be time-consuming and resource-intensive. Moreover, the sheer complexity of LLMs makes it challenging to identify the most appropriate down-stream task learning strategy, as the choice of hyperparameters, learning rate, and training duration can significantly impact the model's performance. Chen et al. [16] have also discussed down-stream task learning for LLMs may lead to severe *catastrophic forgetting* since the LLM with a complex architecture is more likely to forget previously learned knowledge and overfits to target domains. In addition to the data requirement and complex model architecture, LLMs typically consist of billions of parameters, e.g., both Generative Pre-trained Transformer 3 (GPT-3)

[11] and Pathways Language Model (PaLM) [20] contains more than 100 billion parameters, which require substantial computational power to train. Fine-tuning or re-training these models necessitates access to high-performance GPUs or specialized hardware, such as TPUs, which can be expensive and difficult to obtain, especially for individual researchers or smaller organizations.

Over the past few years, significant research has been conducted on domain specialization techniques for LLMs. Many methods focus on generic technical contributions, adaptable to specific domains with minor modifications and access to domain-specific information. However, cross-referencing these techniques across different application domains remains a challenge, as does the absence of a systematic standardization and summary of methods for evaluating various domain specialization techniques. This lack of clarity creates obstacles for non-AI professionals and obfuscates existing bottlenecks, pitfalls, open problems, and potential future research directions. To surmount these obstacles and harness artificial intelligence for more effectively accomplishing tasks across various domains, this survey paper offers a comprehensive and systematic review of the current state-of-the-art LLM domain specialization. The major contributions of this paper include:

- **A systematic categorization and taxonomy of LLMs domain specialization techniques**: We comprehensively classify existing methods based on different levels (i.e., black-box, grey-box, and white-box) of accessibility to the LLM and organize their corresponding techniques into a taxonomy. We discuss details, relationships, pros, and cons among different subcategories. The proposed taxonomy is designed to assist domain experts in identifying the most suitable techniques for their target problem settings.
- **A comprehensive categorization and summarization of major application domains**: We debut the taxonomy of representative application domains that domain-specialized LLMs can enhance. The practical significance and open challenges for each application domain or subdomain are elucidated, allowing for easy mapping to the proposed technique taxonomy. Researchers and various domain experts could cross-reference additional application domains for evaluating their newly proposed methods while expanding their advanced techniques to encompass new application domains.
- **An insightful discussion of the current status of research in this area and future trends**. Based on the comprehensive and systematic survey and investigation of existing domain specialization techniques and applications, an overall picture and trends of LLM domain specialization have been outlined and discussed. The open challenges and The paper concludes by presenting fresh insights into the bottlenecks, open problems, as well as a discussion of possible future directions.

1.1 Related Surveys

This section briefly outlines previous surveys relevant to the domain specialization of LLMs in three categories: (1) fundamental overview of PLMs and LLMs; (2). techniques on domain adaptation and generalization of PLMs; and (3). Specializing language models for specific domains.

Fundamental overview of PLMs and LLMs. While comprehensive reviews [101, 123] of PLMs and their use in diverse NLP tasks exist, they don't necessarily apply to LLMs due to differences between the two. Given the recent growth in popularity and effectiveness of LLMs, several review papers have emerged, addressing various LLM aspects. Some focus on fundamental LLM components [86, 173, 186], others on the history and potential applications of generative AI [13, 180], and a few [100] on enhancing LLMs with reasoning capabilities. However, a comprehensive review and technical taxonomy of LLM domain specialization are yet to be provided.

Domain adaptation and generalization of PLMs. Surveys [31, 42] examine how to effectively and efficiently adapt PLMs to specific domains, such as adding a layer to the model or updating the model parameters. However, most of these techniques don't apply to LLMs because of the inaccessibility of their architecture and parameter space. Also, updating knowledge in LLMs is challenging due to computational costs and the need for efficient optimization strategies.

Specializing language models for specific domains. Recent review papers have emphasized the benefits and necessity of customizing LLMs for specific domains. Risks linked with applying generic LLMs to areas like medical education have been noted in [24, 133], including lack of originality and inaccuracies. Practical considerations for legal domain-specific language models have also been suggested in [139]. In the finance sector, initial steps towards a finance-specialized LLM have shown improved performance on financial tasks without compromising general benchmarks [169]. These advances highlight the need for a comprehensive review and technical taxonomy of domain specialization techniques to assist different sectors in effectively employing LLMs for their unique tasks.

2 TAXONOMY OF DOMAIN SPECIALIZATION

Large language models are typically referred to as large-scale pre-trained language models (PLMs) based on the Transformer architecture [101, 123]. Empirical evidence suggests that scaling pre-trained language models, such as increasing the model size or data size, frequently results in enhanced model capacity for downstream tasks. In this section, we begin by reviewing the fundamental concepts of PLMs and proceed to present a comprehensive taxonomy of existing techniques aimed at specializing large language models for specific domains.

2.1 Background

Specifically, PLM is a type of neural network pre-trained on a large corpus of text data to learn linguistic patterns, structures, and semantics. The input and output of PLMs can be described as follows. In LLMs, the *input* is a text sequence that serves as context for understanding and processing. To clarify the task, a *prompt*, an additional sentence or query, is often included. These prompts, designed based on the NLP task, provide a premise or task explanation. For instance, in text summarization, a prompt like "Summarize the key points in the following passage:" could precede the input passage. The *output* is the text sequence or prediction generated in response to the input. Depending on the task, this could be an answer to a question or a sentiment label, and may require post-processing like token decoding or label extraction for final presentation. As LLMs are typically scaled-up versions of PLMs, they follow the similar architecture design of PLMs, which come in three main flavors: *encoder-only*, *encoder-decoder*, and *decoder-only* architectures. This brief introduction will provide an overview of these PLMs architectures, and discuss their differences and commonalities.

- *Encoder-only Language Models* process input text into vector representations without an explicit decoding phase to generate new text. Instead, they transform and embed text into a high-dimensional space. These models are primarily designed to capture and understand the patterns and semantics in the input data. They are extensively used for tasks such as text classification, sentiment analysis, and clustering. One of the notable examples is BERT [30], which extracts context-rich embeddings for downstream tasks by pre-training on a masked language modeling objective.
- *Encoder-Decoder Language Models* consist of an encoder that processes input text into vector representations and a decoder that generates output text from these representations. They employ cross-entropy loss as the objective function, comparing the actual and predicted target sequences. These PLMs are often used for sequence-to-sequence tasks like machine translation and summarization, with T5 [125] being a notable example.

- *Decoder-only Language Models*, like GPT [124], are autoregressive language models that generate the next word in a sequence based on previous words. They map a sequence of tokens to a vector representation and generate contextually relevant content autoregressively, calculating the probability of the next token based on the context. This autoregressive modeling approach is particularly suitable for text-generation tasks.

2.2   Taxonomy of Domain Specialization Techniques
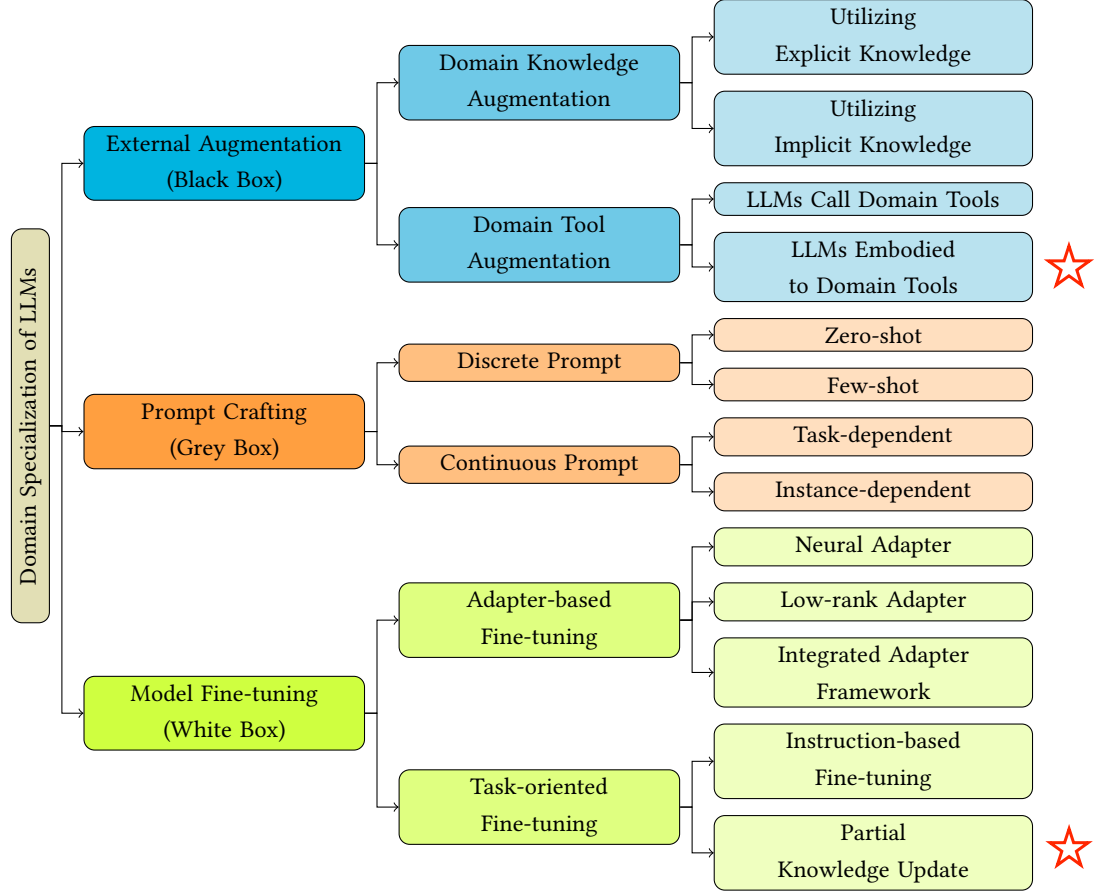
Fig. 1. The taxonomy for current techniques on LLM domain specialization.

Domain specialization of LLMs can be understood as tailoring broad, universally-trained LLMs to operate optimally within a specific field or domain. To tackle the three challenges of domain specialization mentioned in Section 1, respectively, the approaches in LLM domain specialization can be categorized into three corresponding classes of approaches: *external augmentation*, *prompt crafting*, and *model fine-tuning*. These classes correspond to *assumptions* of different levels of accessibility to LLMs, namely, no access (black box), partial access (grey box), and full access (white box). The black box *assumption* typically indicates we only have access to the model API (e.g., ChatGPT) without knowing any information but the generated output; the grey box *assumption* denotes we have limited information (e.g., the probability of generated tokens in GPT-3 API), such information can guide us to design and fine-tune a suitable

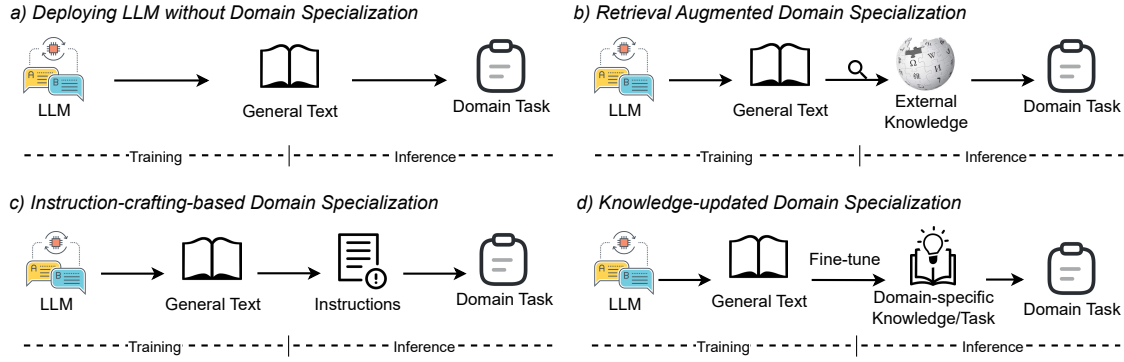Fig. 2. This exposition discusses different approaches for tailoring LLMs to domain-specific tasks: (a) using an LLM trained on general corpora without modifications, (b) enhancing the LLM's performance through retrieving relevant external knowledge, (c) utilizing domain-specific and task-relevant instructions to improve LLM's capabilities, and (d) updating the LLM's internal knowledge with domain-specific text and tasks.

prompt to elicit domain knowledge better; and the white box *assumption* indicates we have full access to the LLM (e.g., LLaMA and its variants), including the parameter setting, training data, and the model architecture.

Other than the LLM accessibility-based taxonomy, one way to categorize LLM domain specialization methods is based on the training strategy used, such as fine-tuning an existing model with domain-specific data, training a model from scratch specifically for the domain, or employing a mixed training strategy. An additional taxonomy could be based on the intervention level: pre-training intervention involves modifying the pre-training process to encourage domain-specific knowledge, the fine-tuning intervention involves adaptations during the fine-tuning stage, and inference-time intervention involves modifying the model's behavior during the actual application to generate more domain-specific outputs. Furthermore, the taxonomy can be established based on the evaluation and feedback mechanism: fixed evaluation sets a constant benchmark, dynamic evaluation involves continuous performance assessment with changing benchmarks, and user feedback-based evaluation uses direct user input as a signal to specialize the model's responses.

In this survey, we categorize existing approaches based on the LLM's **accessibility** and provide an overview of each approach in Figure 2. To be more specific, *1) External augmentation (black box)* does not necessarily require access to the LLM's inner parameter space, making it the most accessible for users with limited resources (e.g., computational resources, domain-specific data). As shown in Figure 2 (b), by using external resources or tools, domain-specific knowledge is incorporated into the input prompt, generated output, or both, effectively adapting the LLM's performance without modifying its internal structure. *2) prompt crafting (grey box)* involves designing various types of prompts by accessing the gradient or loss values of LLMs, allowing for finer control over the model's behavior. *3) model fine-tuning (white box)* demands the most access and resources, as it involves updating the LLM's parameters to incorporate domain-specific knowledge directly into the model. (Figure 2 (d)).

Relations between approaches in different categories.

- **Different levels of specialization**: Each approach operates at a different level of specialization (i.e., black box, grey box, and white box). Augmenting with external knowledge provides a focused injection of domain-specific information while prompt engineering works at the input level, shaping the model's inference process. Fine-tuning modifies the LLM's internal parameters, leading to more profound changes in the model's behavior.

- **Trade-offs**: The approaches exhibit different trade-offs regarding computational cost, ease of implementation, and generalization. Augmenting with external information and crafting task-specific instructions are often less computationally expensive than knowledge updates of LLMs but may not yield the same level of performance improvement. Fine-tuning and neural adapters can provide more substantial performance gains but can be more challenging to implement and may suffer from reduced generalization capabilities if overfitting occurs.
- **Complementary nature**: The three approaches can be used independently or in combination to achieve better performance on domain-specific tasks. For instance, external knowledge can be integrated with a fine-tuned LLM to leverage both specialized knowledge and optimized parameters. Similarly, carefully designed prompts can be used alongside neural adapters to guide the model's output while taking advantage of the newly learned domain-specific knowledge.

Common Framework. Researchers can utilize these methods independently or in combination to achieve optimal performance on specific tasks while considering the unique requirements and constraints of each approach. In this paper, we provide a common framework underlying the black box, grey box, and white box methods for domain specialization of LLMs, which is a process consisting of four core stages: *Definition*, *Augmentation*, *Optimization*, and *Evaluation*.

(1) **Definition**: This is the first step where the specific domain, the objectives within that domain, and any constraints are clearly defined. Whether we fine-tune a model (white box), craft prompts (grey box), or augment inputs/outputs (black box), it requires a clear understanding of the domain we are specializing for. This also helps in identifying the specific data, knowledge, and resources relevant to the domain that could be used in the following steps.

(2) **Augmentation**: This stage involves the incorporation of domain-specific knowledge into the model, or its inputs/outputs. In a white box approach, this could involve fine-tuning the model with domain-specific data. For a grey box approach, it might involve using gradients or loss values to craft prompts that steer the model toward domain-specific responses. In a black box method, it could involve using external tools or resources to modify the input prompt or the generated output to make it more domain-specific.

(3) **Optimization**: Once the model or its inputs/outputs are augmented with domain knowledge, the next step is to optimize the model's performance to best fulfill the domain objectives. This can be done through methods like gradient descent for a white box approach, prompt engineering for a grey box approach, or post-processing and filtering of outputs for a black box approach.

(4) **Evaluation**: The final stage involves testing the specialized model's performance against predefined benchmarks, gathering feedback, and refining the model based on this feedback. This could involve running the model on a domain-specific test set or getting feedback from domain experts.

3 EXTERNAL AUGMENTATION FOR DOMAIN SPECIALIZATION

Retrieval augmentation aims to enhance LLMs by retrieving relevant information from external sources, without fine-tuning model parameters. There are two primary categories: (1) **Domain Knowledge Augmentation**, where LLMs are provided with domain-specific context from an external knowledge source, and (2) **Domain Tool Augmentation**, which integrates LLMs with external systems or tools, often via APIs. Domain Knowledge Augmentation supplements the model's responses with external information, while Domain Tool Augmentation expands the model's capabilities for tasks it couldn't perform otherwise. Domain Knowledge improves depth and accuracy within a specific field, while Domain Tools allow the model to perform tasks beyond its inherent abilities. This section discusses both approaches, their limitations, and their advantages.

3.1 Domain Knowledge Augmentation

Domain knowledge, in the broadest sense, is a comprehensive understanding of a specific field or subject area. It includes concepts, principles, facts, and patterns that are unique to a particular domain. The knowledge can be represented in various forms, including a set of documents, a domain-specific knowledge graph, or a neural network that contains parametric domain knowledge. Domain knowledge augmentation in LLM specification refers to the process of enriching an LLM's performance in specific domains by incorporating additional information from domain knowledge. Two categories of external knowledge typically can facilitate LLMs in their domain specialization: **explicit knowledge** refers to knowledge that is clearly defined, easily expressed, and structured in a manner that can be directly understood and utilized; and **implicit knowledge** refers to knowledge that is not directly stated or easily expressed, but is embedded within the data or the system, often in a latent, non-obvious form.

3.1.1 Utilizing Explicit Knowledge with LLM. A conventional method for customizing language models to domain-specific tasks is to retrieve domain-specific information from external context. When presented with an explicit knowledge source containing domain-specific information, it is crucial for LLMs to prioritize the context if the data source holds task-relevant details that contradict the model's memorized knowledge. This strategy ensures that model predictions are anchored in the context, allowing for the refinement or correction of specific model predictions without the need for frequent retraining.

Current techniques often employ a neural retriever to acquire task-relevant information from either a large corpus (e.g., Wikipedia) or a knowledge base (e.g., Wikidata) [10, 26, 44, 57, 74, 75, 83, 88, 143]. Specifically, given a task-specific query, early works [10, 74, 83, 143] designed neural retrievers to vectorize the query and all information in the external knowledge source to search for relevant information based on various similarity metrics (e.g., cosine similarity) in the latent space. The searched information can then be concatenated with the query for downstream tasks. With the prevalence of LLMs, researchers have been using LLMs to replace the neural network-based retriever [26, 57, 126], and one work [57] demonstrated that coupling a rather lightweight LLM (around 11 billion parameter size) with an external knowledge base can achieve similar performance when using a 540B LLM (i.e., PaLM). Furthermore, in order to enhance the transparency and explainability of the retrieval, He et al. [44] proposed to leverage LLMs to decompose the information retrieval process with detailed reasoning steps, and Lu et al. [88] explored to utilize LLMs to verify whether information obtained by a pre-trained neural-network-based retriever is relevant or not.

3.1.2 Utilizing Implicit Knowledge with LLM. Implicit domain knowledge in machine learning refers to latent, non-obvious information embedded within data or the system, often represented as vectorized knowledge or embeddings learned during pre-training. Such embeddings capture intricate data patterns, symbolizing domain knowledge in an abstract form. Previous research [36, 39, 99, 155] suggests the use of attention mechanisms to enable PLMs to retrieve task-related information from this implicit knowledge. These studies transform task-specific queries into latent embeddings, calculating attention scores between the query vector and each knowledge entry. A softmax function is used to generate a weight or probability distribution across all knowledge entries concerning the input query. The retrieved memory vector is then obtained via a weighted sum of the memory entries, using attention weights. This method enhances traditional neural networks with implicit knowledge, permitting the model to access relevant, current information during inference.

While LLMs can store a substantial amount of information in their parameters to generate high-quality responses, augmentation with implicit knowledge isn't always required. Unlike explicit knowledge, implicit knowledge requires

extra processing, such as transforming domain-specific data into latent vectors, making it less practical. Despite the limited work in augmenting LLMs with implicit knowledge, researchers are exploring its potential, including its use in storing instructional knowledge about a domain. This approach involves creating an instruction cycle that retrieves the next input prompt from implicit knowledge, parses the LLM's output to recover variable assignments, and stores these back into the memory for retrieving the next instruction. Augmenting LLMs with this instruction cycle allows them to process large inputs and potentially solve complex domain-specific problems [137].

3.1.3 Open Challenges. By incorporating external knowledge, LLMs function like librarians, finding relevant information without needing to memorize all resources. This enhances performance in specialized tasks without extensive retraining, enabling more adaptable and efficient AI systems capable of lifelong learning and knowledge updating. However, augmenting LLMs with external knowledge for domain-specific tasks presents several open challenges.

(1) *Seamless integration*: Seamless integration of external knowledge into LLMs is crucial, whether the knowledge is explicit or implicit. Existing methods typically concatenate retrieved knowledge to the LLM's input or intermediate layers. However, it's important for the LLM to have the option of accepting or rejecting retrieved information, given that such information may be incomplete or conflicting.

(2) *Scalability and adaptability*: Designing systems capable of scaling to manage large amounts of domain-specific data and adapting to new or changing information is challenging. With rapidly expanding knowledge bases, computing pairwise knowledge similarity will become increasingly computationally unfeasible.

3.2 Domain Tool Augmentation

Domain tools refer to specialized software, libraries, or frameworks that are developed specifically for a particular domain or field (*e.g.*, NCBI Web APIs for genomics question answering [64], automated formal theorem prover for mathematical proofs [61], sandbox environment for social behavior simulation [112], etc.). These tools are designed to handle domain-specific tasks, data, or knowledge effectively, which often incorporate algorithms, techniques, or data structures that are tailored to the unique requirements of that domain. However, the utilization of these domain tools often demands strict adherence to input formats or extensive training, making them less accessible to general users. On the other hand, LLMs are artificial general intelligence models that exhibit intelligence and cognitive capabilities across a wide range of tasks and domains. Despite their versatility, current LLMs are constrained in tasks that require domain specialization. These limitations [100, 135] include (1) unstable result formats depending on the random seeds, generation hyperparameters, and input contents [136]; (2) inability to access up-to-date information [108] since LLMs are solely capable of acquiring information from their training data; (3) a tendency to make up facts observed by researchers [59]; (4) lack of precision in certain tasks such as arithmetic [37].

Researchers propose a collaborative integration approach to overcome the limitations of solely using either domain tools or LLMs for complex domain-specific tasks. This approach combines the strengths of both, utilizing domain-specific knowledge, algorithms, and functionalities from the tools, while offering a user-friendly interface through LLMs. This collaboration optimizes the use of domain-specific resources and eases user engagement by allowing LLMs to guide effective tool usage.

LLMs Call Domain Tools. One straightforward way for domain tool augmentation is to allow LLMs to call domain tools. Essentially, this type of approach follows a multi-stage pipeline, given an LLM $f_\Theta(\cdot)$ and a domain tool $\mathcal{T}(\cdot)$: (1) elicit an executable command c for the domain tool from the LLM by curated or constructed prompts p, denoted as
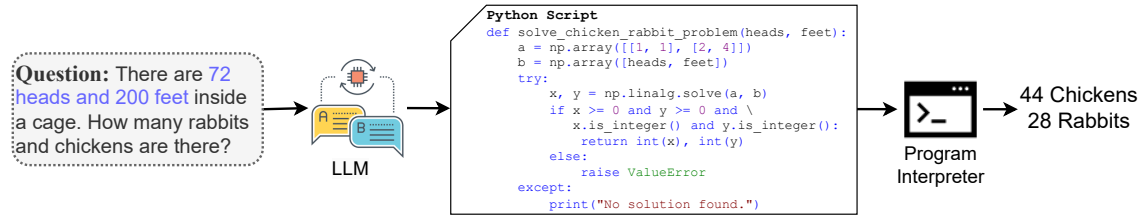
Question: There are 72 heads and 200 feet inside a cage. How many rabbits and chickens are there?

LLM

```
Python Script
def solve_chicken_rabbit_problem(heads, feet):
    a = np.array([[1, 1], [2, 4]])
    b = np.array([heads, feet])
    try:
        x, y = np.linalg.solve(a, b)
        if x >= 0 and y >= 0 and \
            x.is_integer() and y.is_integer():
            return int(x), int(y)
        else:
            raise ValueError
    except:
        print("No solution found.")
```

Program Interpreter

44 Chickens
28 Rabbits

Fig. 3. A toy example for LLMs call domain tools.

$c = f_\Theta(p)$. (2) execute the command c in the domain tool and get the outputs, denoted as $r = \mathcal{T}(c)$. (3) post-process the domain tool outputs by pre-defined rules or the LLM, denoted by $y = post - process(r)$.

This pipeline provides a general diagram and can be easily expanded to multi-LLMs multi-tools collaboration scenarios. The key technical challenge is to ensure the instruction-following and validity of generated commands c from LLMs, so that domain tools can accurately solve desired tasks. Most existing works propose to utilize zero-show or few-shot prompting for executable commands generation (please refer to Sec. 4 for more details). Figure 3 shows a toy example, where the task is to solve an arithmetic question "*There are 72 heads and 200 feet inside a cage. How many rabbits and chickens are there?*". To elicit LLMs to generate an executable Python program, we can formulate the prompt as "*Please write a Python script to solve the arithmetic question. Question: {question_text}*". Then, a snippet of scripts is returned by LLM as the executable command c for the Python interpreter. Finally, the Python interpreter responds with the program outputs "*44, 28*", and further post-processed into desired results "*44 Chickens and 28 Rabbits*".

Depending on the types of domain tools, LLMs can generate corresponding commands that adhere to the syntax and format requirements to call them. Many domain tools provide APIs for easy and precise access. Early exploration in this direction is to elicit elicit search engine queries (*e.g.*, WebGPT [108], WizInternet [69], GopherCite [97]) written in natural language or database queries (*e.g.*, Binder-SQL [19], DIN-SQL [118], BIRD [78]) in the programming language. Later, researchers study how to elicit LLMs to write executable codes that can be executed in program interpreters such as Python [17, 37, 148], Wolfram [168], and so on. Other than the widely-used search queries, database queries, and Python programs, there exist many domain-specialized APIs that have unique syntax. For instance, chatGPT plugin system [109] introduces how to utilize tools for travel booking, restaurant reservation, e-commerce shopping, and workflow automation. These API calling scripts are typically generated by zero-shot or few-shot prompting techniques, as stated in the toy example.

Some complex tasks may involve more than one type of tool to accomplish. Following this vibe, researchers start to generalize LLMs as **task planners** (also mentioned as "API selectors" or "controllers") that call multiple types of domain tools. Other than generating executable commands for each used tool, these approaches focus on how to decompose a complex task into a set of concrete subtasks and how to coordinate between multiple tools. For instance, DSP [61] proposes a Draft, Sketch, and Prove framework for automated theorem proofs where (1) an LLM or oracle is used to draft informal proofs described in a mixture of natural and mathematical languages from input statements, (2) another LLM is used to generate formal sketch from previous informal proof, and (3) off-the-shelf prover is used to prove the open conjectures inside each formal sketch. TaskMatrix.AI [81] proposes using LLMs to derive high-level solution outlines for domain-specific tasks, and automatically match some of the sub-tasks in the outlines to the off-the-shelf domain models/systems to complete them. HuggingGPT [140] proposes leveraging LLMs to act as the controllers to manage existing domain models to solve complicated domain tasks. Qin et al. [122] propose a general tool-augmented LLMs

framework to decompose complex tasks into several subtasks, dynamically adjust the execution plan, and effectively finish each subtask with appropriate tools.

LLMs Embodied to Domain Tools. LLMs can also be called by domain tools to serve as smart agents in interactive environments, namely *LLMs embodied to domain tools*. LLMs, when embodied in interactive robots, can serve as the decision-making module for domain-specific applications. For example, PROGPROMPT [144] investigates LLMs' ability to assist robots in completing tasks, when the robot's perception module observes surrounding objects and the LLM is prompted with available action specifications. Results indicate the LLM can generate situated actions for simulated household and real-world tabletop tasks. Furthermore, Murali et al. [107] employ LLMs as the primary component for identifying different speakers in multiparty conversations involving a social robot. The robotics community is progressively exploring these areas, studying LLM utility in human-robot interfaces, planning, grounding, and more [27, 80, 164]. Furthermore, researchers start to investigate how multiple LLMs can *interact with the environment* or *communicate and collaborate* together for real-world task-solving. Mind's eye [84] studies how LLMs can benefit from the interaction with simulated physics engines to inject grounded rationale for physics alignment tasks. CAMEL [76] proposes a communicative agent framework to assign different roles to LLM agents so that multiple AI agents can collaboratively communicate by chatting with each other in an instruction-following fashion to solve the specified task. A recent work [112] utilizes twenty-five LLMs as generative agents in a game-based sandbox environment to create believable simulations of human behavior for interactive applications.

3.2.1 Open Challenges. By leveraging the power of LLMs, domain tools can assist in a variety of tasks across multiple fields, including robotics, virtual agents, and problem-solving in real-world scenarios. This allows for more intuitive and seamless human-machine collaboration, leading to increased efficiency and adaptability in tackling complex problems.

Augmenting LLMs with domain tools poses several open challenges:

(1) *Automated integration*: At present, augmenting LLMs with domain-specific tools requires a significant amount of effort to ensure proper integration. A promising future direction involves utilizing LLMs as a unified interface through standardized protocols to connect various applications and services, thereby enabling seamless communication and interaction between them.

(2) *Getting rid of domain tools*: Another direction for the future development of LLMs is to focus on creating a powerful artificial general intelligence (AGI) model that is not dependent on external tools or domain-specific knowledge. An AGI model would have the potential to revolutionize the way we use language models, enabling more complex and sophisticated tasks to be performed with greater ease and efficiency.

4 PROMPT CRAFTING FOR DOMAIN SPECIALIZATION

While LLMs trained on large-scale corpora are powerful, further pre-training on prompts can enhance their ability to adhere to user intentions and generate accurate and less toxic responses [111, 161]. Prompts, or task-specific input texts designed to elicit specific model responses, help guide the LLM's content generation process and set expectations for the desired output. Approaches generally fall into two categories: (1) **Discrete Prompt** involves creating task-specific natural language instructions to prompt LLMs, eliciting domain-specific knowledge from their parameter space, and (2) **Continuous Prompt** uses learnable vectors to prompt LLMs, eliminating the need for manually designed text instructions. This section delves into both approaches and the merits and limitations of domain specialization.

4.1 Discrete Prompt

Recent works [11, 110] allow LLMs to quickly adapt to unseen domains by discrete prompting, and GPT-3 [11] is the first work that introduces how to perform an unseen task using an LLM with zero-shot/few-shot discrete prompts without updating the LLM's inner parameter. We give a formal definition of the discrete prompt framework below.

Problem Setup. Given an LLM $f_\Theta(\cdot)$ where Θ denotes pre-trained model parameters, the task is to elicit desired output $\boldsymbol{y}$ from LLM with a discrete prompt p and a test query, denoted as $\boldsymbol{y} = f_\Theta([p; \boldsymbol{c}])$, when freezing Θ. It is worth noting that both $\boldsymbol{y}$, p and $\boldsymbol{c}$ are sequences of tokens (i.e., natural language sentences). The rationale behind using discrete prompts is that they can serve as instructions to elicit the generalized reasoning abilities of LLMs. By following such instructions, LLMs can perform domain-specific tasks that they have not been specifically trained for. This approach allows LLMs to demonstrate their ability to apply previously learned knowledge to new and diverse situations, thus enhancing their overall effectiveness and utility.

Depending on the prompting types, discrete prompts can be divided into two categories: (1) zero-shot [7, 68, 134], where the prompt p consist of only the task description; and (2) few-shot [90, 102, 167], where the prompt p consists of the task description and few illustrative examples. The key difference between zero-shot and few-shot prompts is whether or not illustrative examples are provided.

4.1.1 Zero-shot Discrete Prompts. The zero-shot setting represents the cold-start scenario, where not a single supportive labeled example is available.

Figure 4 presents a toy example of how zero-shot discrete prompts work. The task description that compromises the prompt p can be curated by human users or automatically generated by templates, where the intent of the task and the expected outcomes are described in natural language. However, as stated in [136], post-process is sometimes required to extract the rigorous prediction results from the unbounded raw outputs. Researchers demonstrate that instruction alignment pre-training enables decent zero-shot performance on various unseen tasks [11, 134, 165], where different tasks can be represented in a unified sequence generation format. PADA [7] is one of the pioneering works that explore how to elicit the domain adaptation ability of LLMs for domains unseen during the training phase. PADA first generates the

> \# Task description
> Please determine if the two sentences entail, contradict, or are neutral to each other.
> \# Test query
> **Premise**: She emerged vigorous with Apgar of 7 and 8.
> **Hypothesis**: She had low APGAR scores.
> **Answer**:
> \# LLM response
> LLM: Contradiction

Fig. 4. An example (adapted from [70]) of zero-shot discrete prompts, where task description, and/or a test query are provided to LLMs. No illustrative examples are provided in zero-shot prompts.

target domain name followed by a set of domain-related features related to the test query, and then uses them together as the prompt to predict task labels. Follow-up works explore how to utilize zero-shot discrete prompt for domain adaptation in sentiment analysis [60], image classification [38], semantic segmentation [35], and rumor detection [82]. Later, Kojima et al. [68] extend the few-shot-Chain-of-Thoughts(Few-shot-CoT) [167] into zero-shot-CoT to elicit multi-step reasoning ability of LLMs. The core idea of Zero-shot-CoT is a two-stage prompting, where the 1st stage simply adds the same prompt "*Let's think step by step*" before each answer to derive the reasoning process sentences, and the 2nd stage takes the generated reasoning sentences to generate the final answer. Zero-shot-CoT has achieved

significantly stronger performance than the standard zero-shot prompting method on arithmetic, symbolic reasoning, and other logical reasoning tasks.

4.1.2 Few-shot Discrete Prompts. The few-shot setting reflects the characteristics of sparse training samples of many domain-specific applications (i.e., only a few annotated examples are available).

Task description

Please determine if the two sentences entail, contradict, or are neutral to each other. Below are several examples.

Example 1

Premise: ALT, AST, and lactate were elevated as noted above.
Hypothesis: The patient has abnormal lfts.
Answer: Entailment

Example 2

Premise: Chest x-ray showed mild congestive heart failure.
Hypothesis: The patient complains of cough.
Answer: Neutral

Test query

Premise: She emerged vigorous with Apgar of 7 and 8.
Hypothesis: She had low APGAR scores.
Answer:

LLM response

LLM: Contradiction

Fig. 5. An example (adapted from [70]) of few-shot discrete prompts, where task description, illustrative examples, and/or a test query are provided to LLMs. Compared to zero-shot prompts, a few supportive labeled samples are provided in few-shot prompts.

Figure 5 presents a toy example of how few-shot discrete prompts work. Different from zero-shot prompts, a few examples that further convey the task intention and provide illustrations of the desired output formats are included in the prompt p. Researchers have observed that few-shot prompts yield more stable output formats and more decent performance on downstream tasks [11, 110]. Chain-of-Thought (CoT) [167] improves the domain specialization ability of LLMs by introducing a series of intermediate reasoning steps for complex reasoning tasks, but it also brings extra cost in manually designing CoTs for each test example. As a follow-up, Auto-CoT [184] eliminates manual designs by appending the "*Let's think step by step*" prompt to the given task context and letting LLMs generate reasoning chains directly. Other than the natural language format instructions, CoCoGen [90] studies the programming language format instructions to tackle structured reasoning tasks.

More advanced techniques are then proposed to further improve discrete instruction of LLMs for domain specialization and customization. For instance, *ensemble-based instruction* [89, 159, 160] utilizes multiple different instructions to derive multiple model outputs and then aggregates these outputs to achieve better task performance. Another line of research proposes *recursive instruction* [2, 32, 33, 67, 172, 187] to breaks down a complex unseen task into a series of subtasks that are relatively easier to solve and then employs LLMs with specific instructions for each subtask.

4.1.3 Open Challenges. Utilizing discrete prompts helps LLMs leverage their inherent knowledge to adapt to new and diverse situations. This approach not only demonstrates the flexibility and adaptability of LLMs but also enhances their overall effectiveness and utility across a wide range of domains and tasks. However, crafting discrete prompts of LLMs for domain specialization poses several open challenges:

(1) *Effectiveness*: Often the discrete instructions are curated by domain experts or follow some types of templates. It is arguable whether the instructions used are the most effective ones. Therefore, there is a need for evaluation

of these instructions. This can be achieved through collaboration between domain experts and data scientists, who can analyze the performance of the LLMs and adjust the instructions accordingly. An automatic evaluation would be even better.

(2) *Scalability and adaptability*: Automated ways to generate and select/combine discrete instructions without excessive human intervention is another promising direction to improve discrete instructions of LLMs.

4.2 Continuous Prompt

Similar to discrete prompts, the continuous prompt is a sequence of tokens proposed to attach with the input sentence and guide LLMs with extra knowledge but can be learned from the downstream dataset by continuous *prompt tuning*. In this case, the continuous prompt serves as a *soft* parameterized prompt instead of the hard-coded instruction as discrete language phrases. Prompt tuning is to optimize the prompt that adapts an LLM to customized tasks or domains with the preservation of LLM's general language understanding ability. Here, one can solely update prompt-related parameters, whose quantity is around only 0.01% of the total number of the LLM's parameters, while freezing the LLM itself during the fine-tuning phase.
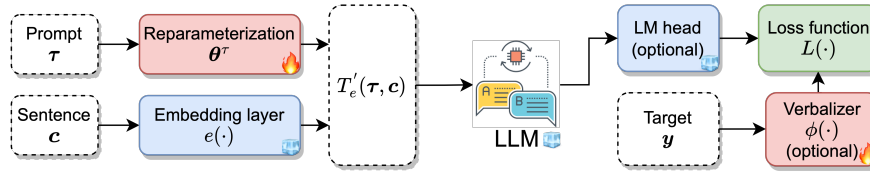

Fig. 6. An illustration of soft prompt tuning. Fire icon represents tunable modules and ice icon represents that parameters of those modules are frozen during tuning. Verbalizer are only used for classification task where a mapping from class label to label words is required, which can be one-one mapping, trainable tokens [43], or enhanced with extra knowledge [53].

A general framework of continuous prompt tuning (Figure 6) can be concisely described in the following stages: (1) Given an input sentence c and its corresponding target y, a template function $T(\cdot)$ organizes them along with a prompt τ of length m into a new sentence $T(\tau, c) = \{\tau_{0:i}, c, \tau_{i+1:m}\}$. (2) Subsequently, the sequence $T(\tau, c)$ is mapped into an embedding space using model's input layer $e(\cdot)$, resulting in the sequence of token embeddings: $T_e(\tau, c) = \{e(\tau_1), ..., e(\tau_i), e(\omega_1), ..., e(\omega_n), e(\tau_{i+1}), ..., e(\tau_m)\}$, where τ_i is the i-th token in the prompt and $T_e(\cdot)$ denotes the sequence in the embedding space. To perform prompt tuning, τ is considered as pseudo tokens without explicit semantic meanings, and thus $e(\tau_i)$ is replaced with a trainable tensor $h(\tau_i)$ reparameterized by θ^τ. This modifies the template to: $T'_e(\tau, c) = \{h(\tau_1), ..., h(\tau_i), e(x_1), ..., e(x_n), h(\tau_{i+1}), ..., h(\tau_m)\}$. (3) Finally, we can feed the embedding sequence to an LLM, and optimize the continuous prompts θ^τ using the downstream loss function $\mathcal{L}$ as follows:

$$\theta^{\tau\star} = \underset{\theta^\tau}{arg\,max}\ \mathcal{L}(f_\Theta(T'_e(\tau, c)), y),$$

where $f_\Theta(\cdot)$ is the LLM function parametrized by Θ. For a cloze-style input reformulated from general tasks, for example, the sentiment analysis task for the sentence "I like the movie!" can be rephrased as a cloze-completion problem: "I like the movie! It was [MASK].". The predicted words at the masked position are then employed for subsequent classification. In this case, a unique token [MASK] is integrated during the generation of the template in step (1), and a verbalizer ϕ is required to map class labels to words in the language model's vocabulary, e.g., *positive→ 'great'*, resulting in:

$$\theta^{\tau\star} = \underset{\theta^\tau}{arg\,max} \sum_c \log P\left([MASK] = \phi(y)|T'_e(\tau, c)\right)$$

The information condensed by the prompt falls into two categories: (1) **task-dependent prompt tuning**, and (2) **instance-dependent prompt tuning**. Each category encompasses general and specific enhancements for domain and task adaptation. Although some studies are based on PLMs, the advantages apply to LLMs, given the correlation between prompt tuning enhancements and model size [72] and successful implementations on large-scale PLMs. Moreover, it provides a parameter-efficient, fully controllable tuning method to adapt PLMs for more customized purposes.

4.2.1 Task-dependent Prompt Tuning. Task-dependent prompt tuning optimizes a shared prompt for all instances within a specific task, enabling it to encapsulate information from extensive datasets comprising thousands or millions of examples. However, training a naïve prompt is hard to converge and suboptimal for different scenarios, leaving room for improvement for specific tasks and domains.

Prompt Content Enhancement. We refer prompt content as the embedding values of continuous prompt, enhancements are developed in terms of task-specific initialization and prior knowledge transfer. Pilot works have validated that in contrast to many optimizers that begin with a random distribution applied in general ML tasks, the optimization process of soft prompt is significantly influenced by its initial value. For language models, word embeddings are pre-trained to be quite distinct. Consequently, a standard optimizer such as stochastic gradient descent (SGD) can only update the parameters in a limited vicinity, leading to the possibility of falling into a local minimum [1]. Therefore, a more effective initialization approach would involve using embeddings of concrete task-specific words.

One of the pioneering works, WARP [43] initializes the prompt by the embedding of special token "[MASK]". KnowPrompt [18] designed learnable prompts as virtual type words and virtual answer words, which are initialized by the aggregated representation of concrete label words and disassembling words based on their frequency in the dataset. In addition, random initialization has been proven to be the least efficient, especially for small model, while Prompt-tuning [72] presented no significant gap between initialization strategies when the model size grows to 11B, indicating that LLMs is robust for prompt's initialization values in general tasks.

Further studies have revealed that retrained prompts on source domains can enhance performance in unseen target domains, illustrating the ability of prompt transfer [153]. SPoT [153] initialize the prompt with a single generic source prompt learnt from multiple sources tasks, and then fine-tune it on target task in a classic way [72]. PPT [40] also pre-trains a prompt using self-supervised learning on extensive unlabeled corpora, which then serves as the initial prompt for the target task. Su et al. [145] demonstrated the transferability of continuous prompts in both cross-task as well as cross-model settings, and find that a well-initialized prompt can significantly accelerate training convergence. Furthermore, take the advantage of transferability, LFPT5 [121] employed soft prompt for lifelong learning. It continuously trains the prompt that simultaneously learns to solve the current task and generate training samples of previous tasks to overcome the *catastrophic forgetting*. Progressive prompts [128] introduce the prompt tuning into continuous learning. The prompt for current task is defined as the concatenation of prompts that are optimized on previous tasks and a tunable current prompt.

Prompt Construction Enhancement. We refer to the prompt construction about the positioning and length of the prompt, and combinations of additional templates or discrete prompts. Continuous prompts can be simply prepended, appended, and inserted to the original input sentences without extra language phrases. The pioneering study, WARP [43] adopted all three intersections with a "[MASK]" token for classification tasks. In Prefix-tuning [79], tunable prompts are prepended to the sentence embedding and the activation of all attention blocks, capitalizing on the left-to-right

nature of the autoregressive model: the prepended prompt can efficiently affect the subsequent words through attention. In addition, a recent work [72] prepend prompts at the input layer, achieving comparable results to fine-tuned models.

Template is widely used to leverage the adaptation performance [136], for example, reformulating an NLP task (e.g., sentence classification) into the masked words prediction task that is employed during LM pre-training. Based on the predefined task-specific templates, soft prompts can be inserted and thus offers flexibility for conditional tuning. KnowPrompt [18] designed a template appending to the input sentence with a "[MASK]" between subject and object for relation extraction and incorporates trainable prompts of "virtual type words" surrounding these two entities. Output embeddings of "virtual type words" are trained to align logically with the target relation at the masked position, conditioning the optimization with the information of entity type. KiPT [77] developed a knowledge extractor for event detection tasks, which identifies trigger words in sentences based on their semantic similarity to event concepts above a threshold. Identified trigger words as well as the corresponding event labels will then be prepended to a randomly-initialized soft prompt with the input sentence. KiPT also reformulates the sequence tagging tasks: trigger-identification and trigger classification, into the generative task by outputting structured event records.

4.2.2 Instance-dependent prompt tuning. A shared task-dependent prompt is static against the change in input sentence, which ignores semantic differences as well as specific knowledge of individual instances, and thus be suboptimal in fine-grained objectives. Instance-dependent prompt tuning however conditionally generates prompts for individual instances, incorporating both contextual information and task instructions.

Prompt Content Enhancement. Enhancement of prompt content for instance-dependent tuning focus on learning a joint and adaptive representation of tasks as well as instance context. IDPG [170] proposed an additional two-layer perceptron as a prompt generator, down and up project the sentence embedding to the adaptive soft prompt. ATTEMPT [3] first train multiple prompts on large-scale source tasks, and calculate an aggregated prompt base on a sentence-wise attention network, which will then be mixed with a newly initialized target task prompt as the final instance-dependent prompt. Jin et al., [63] assume that prompt tokens differently contribute the instance, and thus designed a look-up module to score the association of prompt tokens to instance tokens, which is then used to calculate the aggregated prompt embeddings. Bhardwaj et al., [8] generate context-aware prompts by a transformer-based sentence encoder, but further quantize the contextual prompt into a more compact representation to avoid optimization collapse. Levine et al., [73] learn the joint representation of prompt and input by a frozen T5 encoder following cross- and self-attention layers. Liu et al., [85] propose an instance-aware prompt that is applied to the intermediate layers of LM. The proposed prompt generator is a simple feed-forward layer with bottleneck architecture which take the embedding of [CLS] token or pooling of embeddings of sentence tokens.

Prompt Construction Enhancement. Similar to the construction enhancement, instance-dependent prompt tuning introduce instance-dependent knowledge as concrete words or learn adaptive prompt in terms of positioning and length. OntoPrompt [177] enriches the template with instance-related knowledge from external ontology as an additional text, and tune continuous prompts surrounding the "[MASK]" to help prediction. Recently, to give a comprehensive discussion of the effect of content and structure of prompts, dynamic prompting [175] proposed a unified framework to learn an instance-dependent prompt by dynamically defining prompt position, length, and values for each instance. It also proves the effectiveness of post-fix prompt, given most prior works prepend the prompt to the input sentence.

4.2.3 Open Challenges. Continuous prompt tuning presents a streamlined method to utilize the broad language understanding capacity of LLMs for specific tasks across different domains. It efficiently tackles issues inherent in

discrete prompt methods, such as (1) significant reliance on the prompt for LLM performance, where minor wording or template changes can greatly affect the result, (2) computational complexity in identifying the optimal natural language-based prompt from a large search space, and (3) the time-consuming and labor-intensive process of manually designing instructions, particularly in expertise-required domains. However, continuous prompt tuning has its limitations.

(1) *Interpretability* is often criticized as a weakness of soft prompt tuning. By discreting the optimal continuous prompts into nearby token vectors in LM's vacabulary, studies such as WARP [43] have found these prompts to be non-interpretable and lacking meaningful content. In KnowPrompt [18] and Prompt-tuning [72], prompt tokens are discovered in close proximity to domain-related terms. For example, Prompts trained on the BoolQ dataset revealed that *science*, *technology*, and *engineering* were the nearest neighbors of the optimal prompt, as approximately 20% of the questions pertain to the "Nature/Science" category [72]. However, the interpretability of continuous prompts as a coherent sequence is still unclear. In addition, continuous prompt is not confined to directing LLMs using compact textual information. OPTIMA [41] achieves domain adapting with prompt by tuning it help regularizes the decision boundary to be smooth around regions where source and target data distributions are similar with an adversarial learning framework.

(2) *Limited access* to LLMs poses a significant challenge for continuous prompt learning, especially as models with immense sizes (e.g., 540B PaLM) and models with only API access. This restriction hinders differential optimization on continuous embeddings. In this case, derivative-free prompt tuning that optimizes the soft prompt without gradients from LLMs is widely discussed. Black-box tuning (BBT) [146] proposed a gradient-free approach to searching the optimal prompt in a smaller intrinsic space. with Covariance Matrix Adaptation Evolution Strategy (CMA-ES) for non-convex optimization instead of Adam. Similarly, Clip-Tuning [14] use multiple deterministic clipping instances of the target LM to optimize an agent that learns the intrinsic prompt embedding. However, those methods still need access at least to the embedding layer, which is unsatisfactory for LLMs where only textual query is allowed. In this case, derivative-free approaches for discrete prompt search appear to be a more promising direction, and several studies have already achieved preliminary success [29, 119].

5 MODEL FINE-TUNING FOR DOMAIN SPECIALIZATION

LLMs, despite being trained on extensive general text data, might not encode adequate knowledge for specific tasks or domains. In such scenarios, fine-tuning the model on a smaller, domain-specific dataset can enhance its performance within that particular area. This fine-tuning can be divided into two main approaches. Adapter-based Fine-tuning and Task-oriented Fine-tuning. (1) **Adapter-based Fine-tuning:** This approach, as illus-

Fig. 7. The visualization of two approaches to fine-tune LLMs based on domain-specific knowledge, where the blue rectangle denotes the set of parameters in LLM. (a) the adapter-based LLM fine-tuning aims to fine-tune LLMs on specific domains with a small number of extra parameters (i.e., adapter); and (b) the task-oriented model fine-tuning aims to fine-tune LLMs based on specific tasks.

trated in Figure 7 (a), employs neural adapters or modular components to enhance the LLM's performance on domain-specific tasks without major modifications to the LLM's inner parameters. These adapters, typically integrated into the existing LLM architecture, allow for task-specific learning while keeping the original model largely intact. (2) **Task-oriented Fine-tuning:** As represented in Figure 7 (b), this method focuses on modifying the LLM's inner
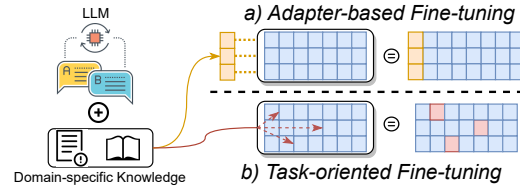
parameters to improve alignment with specific tasks. However, entirely updating all parameters of an LLM may be impractical due to hardware limitations and potential performance degradation. Therefore, the challenge for researchers lies in identifying which parameters require alteration within the expansive parameter space, or in efficiently updating a subset of these parameters. These two approaches allow LLMs to be tailored to specific tasks or domains, offering flexibility and efficiency in handling specialized applications.

5.1 Adapter-based Fine-tuning

Adapter-based fine-tuning aims to add a small number of extra parameters to the LLM for achieving better performance in specific tasks. Typically the additional parameters are encoded in simple modules to *guide* the language model's adaptation to target domains or tasks. The golden spots for the added modules include: (1) simple with a small number of parameters; (2) extensible to the original language models; (3) flexible with sequential training on each specific domain. Most of the proposed strategies with the above favorable properties are built on adapters [49, 129] under the umbrella of parameter-efficient fine-tuning.

5.1.1 Adapters. Adapters are trainable modules inserted between layers of a pre-trained model [49]. The key property of adapters highlights that the parameters of the original language model keep frozen, thus provide sustainable parameter sharing even with varying domains and tasks. Suppose $f_\Theta(\cdot)$ denotes the function of LLM parametrized with the set of parameters Θ and $g_{\Delta\Theta}(\cdot)$ denotes the function of adapters with parameter $\Delta\Theta$, then $f_\Theta \circ g_{\Delta\Theta}$ represents the fine-tuned language model by adapters. Let X be general input data with task performance metric ϕ, and D be the domain training data with domain-specific task performance ϕ_D (for both ϕ and ϕ_D, a higher value indicates better performance), the goal of adapters is to find $g_{\Delta\Theta}$ such that:

$$\phi(f_\Theta(X)) \approx \phi(f_\Theta \circ g_{\Delta\Theta}(X)) \qquad \phi_D(f_\Theta(D)) \le \phi_D(f_\Theta \circ g_{\Delta\Theta}(D))$$

Despite most empirical studies on cross-lingual or multi-task learning, some recent works explore unsupervised domain adaptation particularly using adapters. Unsupervised Domain Adaptation (UDA) using adapters has been explored in recent work, aiming to enhance the cross-lingual or multi-task learning capabilities of pre-trained models. The first attempt [183] targeted multi-domain adaptation with a two-step strategy: domain-fusion training with Masked Language Model (MLM) loss on a mixed corpus, followed by task fine-tuning with a task-specific loss on the domain corpus. Subsequently, UDApter was introduced, which also adopted the two-step training and fine-tuning approach, but segregated this into two adapter modules: a domain adapter and a task adapter. The domain adapter first learned domain-invariant representations, which were then concatenated with the task adapter whose parameters were frozen [92]. This was achieved using the architecture defined in AdapterFusion [114]. AdapterSoup further improved adaptation efficiency by adopting a weight-average of domain adapters only during the testing phase [22]. To select domain adapters, three strategies were explored: exhaustive combination, text clustering, and semantic similarity.

Though these works focused on domain specialization, they were evaluated on pre-trained language models like GPT-2 [22, 92, 183], indicating potential applicability to larger language models. To address this, LLaMA-adapter was designed for efficient adaptation on Large Language Models with Adapters (LLaMAs) using self-instruct demonstrations. The adapter architecture incorporated a zero-init attention mechanism, and the domain specialization capability was tested on instruction-following and multi-modal reasoning tasks [182].

As the application of adapters expands, several techniques, while not explicitly claimed as effective for domain specialization, have either demonstrated potential by offering favorable performance on downstream tasks or served

as integrated components in existing frameworks for domain specialization. Hence, adapters are usually classified based on their architectures into neural adapters and low-rank adapters. With the objective of facilitating user-friendly implementation, a growing body of work is dedicated to building comprehensive frameworks of different adapters [54, 115]. Certain studies have also shown that adapter integration can yield superior performance across a variety of downstream tasks.

Neural adapters. We call adapters with neural network architectures neural adapters. In their original design, [49] uses a composition of down-projection, GeLU non-linearity [47] and up-projection with the feed-forward layers as the backbone. Later [6] simplifies the architecture to a single hidden-layer feed-forward network and demonstrates the effectiveness on domain adaptation. The adapter modules are inserted after the multi-head attention and feed-forward layers in the transformer. These adapters have been named as bottleneck adapters or serial adapters. We use the latter throughout this paper when referring to [49].

The development of neural adapters naturally takes inspiration from neural network architecture design, such as ResNet, autoencoder, attention mechanism, etc. The adapters used in [114] have an additional residual connection. Soon after, [116] proposes MAD-X framework with invertible adapters, which are inserted adjacent to input and inverted to be fed into the output embeddings. At the high-level, invertible adapters can be considered a mimic of autoencoders. Tiny-attention adapter [185] explores the effectiveness of adapters using attention with tiny per-head dimensionality. Till now, most proposed architectures apply fully-connected layers for down-projection and up-projection in adapters. However, Compacters [66] considers parameterized hypercomplex multiplication layers [179] as an alternative, which has a similar form as a fully-connected layer, but learns a sum of Kronecker products. The main advantage is parameter efficiency. Another way of achieving this is inspired by network pruning, as proposed by SparseAdapter[46] to further reduce the training parameters by pruning at initialization. Note that SparseAdapter is a generic technique applicable to neural adapters. Congregating adapters via insertion can be considered as adaptation *inside* the language models, an alternative is adaptation *outside* the language models. *K*-adapters [158] proposes to train multiple adapters individually on various knowledge domains, then inject the learned knowledge with language models by concatenation. Recently, Sung et al. [147] raises a concern on the high training memory required because the backpropagation flows throug the language model with inserted adapters in entirety. They further propose ladder side-tuning, which only adds small modules on the side of the language model connected to the language model backbone via shortcuts. Both techniques use MLP for demonstration, but keep flexible with different adapter architectures.

Low-rank adapters. Low-rank adaptation (LoRA) [52] is inspired by the observation that large language models reside on an intrinsic subspace [?], where model parameters are efficiently updated. Therefore, learning in this subspace significantly reduces the amount of parameters. LoRA modules implant learnable SVD blocks as the subspace with a low matrix rank $r \ll d$, where d is the dimension of input data. The matrices are added in parallel to the pre-trained weights, thus keeping them frozen during the fine-tuning. Notably, LoRA shows superiority in further reducing the number of trained parameters and introducing no latency during inference.

A follow-up work on this line is DyLora [150], which addresses two issues of LoRA using dynamic search: fixed block size and exhaustive search on the optimal rank. Recently, another concern of LoRA was raised that low-rank modules have limited representation power, and further resolved by the Kronecker adapter (KronA) [34]. The essence is to substitute the SVD modules with a Kronecker product module with two matrices of smaller sizes. Despite not many follow-ups on the low-rank adapters, LoRA modules are included in various integrated adaptation frameworks [45, 54, 93, 162] as an important building block. More details on these frameworks follow below.

Integrated adapter framework. With the flourishing results on effective adapters as introduced above, it is a natural extension to incorporate several adapters of various families to boost their performance. AdapterFusion [114] employs a straightforward idea: train multiple adapters on different tasks and combine the learned embeddings from each adapter with a fusion layer. UniPELT [93] proposes to activate different combinations of methods that best suit the current data or task setup via a gating mechanism. The sub-modules included serial adapter [49], LoRA [52], Prefix-tuning [79] and Bitfit [178]. Orthogonal to UniPELT, AdaMix [162] stacks multiple adapters of the same type, but avoids more computational cost by training the activation with stochastic routing. AdaMix can be regarded as a general technique that applies to any adapter, despite their implementation on only serial adapters and LoRA.

The idea of learning a routing function on an inventory of adapters further inspires follow-up works. In the context of multi-task learning, Polytropon [117] jointly learns an inventory of adapters and a routing function to re-combine the fine-tuned adapters of various sizes shared among different tasks. Variants of this scheme are further studied by [12], including the replacement of the routing function with weights averaging, or multi-head routing function to achieve better expressivity. On the implementation-oriented aspect, AdapterHub [115] is the most comprehensive and easy-to-use library integrating all mainstream adapters. The only downside, however, is the absence of support of large language models. Recently, LLM-adapters [54] introduces a framework including open-access large language models such as LLaMA, OPT, GPT-J, etc. It subsumes four adapters as basic components (Serial adapter [49], MAD-X [116], Parallel adapter [176] and LoRA [52]) and remains extensible to new modules. The study of domain specialization further explores mathematical reasoning.

5.1.2 Open Challenges. The adapter's wide applications stem from its modular compatibility with language models, flexible design for integration, and efficient domain-specific data fine-tuning, advancing the adapter-based fine-tuning paradigm. However, these methods have drawbacks. Firstly, the performance of inserted modules can be sensitive to architectural design and size across different tasks and domains, risking insufficient representational power or overfitting on limited data. Secondly, additional modules enlarge the model size, imposing new resource demands and possibly extending inference time. Lastly, as Sung et al. note, the training memory needed by adapter-based methods remains substantial as backpropagation involves the entire model even when previous parameters are frozen [147]. Given these discussions, we outline the open challenges in applying adapters to LLMs for domain specialization:

(1) *Stability and universality:* The performance of adapters can be subject to various architecture or hyper-parameters applied even on pre-trained language models (PTLMs), thus imposes a question mark on the stability and universality. This concern further extends to LLMs. A deeper understanding on how different adapters match with different task settings would be a significant boost to broader applications of adapters.

(2) *Computational resources:* Adapters have shown remarkable results with a million-size of parameters on PTLMs. However, it remains unproven if they are enough for LLMs. If more adapter modules (more parameters) are required, then the issue of computational cost can be raised again. Another ideal spot on this issue is to reduce the training memory with novel architecture design or fine-tuning strategy.

5.2 Task-oriented Fine-tuning

Despite these incredible capabilities of LLMs trained on large text corpus, fundamentally improving the model performances beyond few-shot examples and auxiliary adapters still requires updating the inner parameters of LLMs on an extensive amount of high-quality domain-specific datasets. However, fine-tuning an LLM on any (domain) specific tasks poses two challenges: 1) updating LLM's global knowledge may destroy the in-context learning ability due to
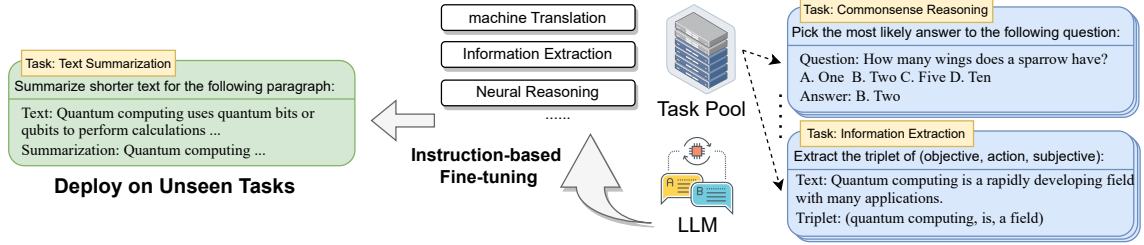
Fig. 8. The overview of fine-tuning an LLM with explicit instructions across various domains and datasets. Particularly, the LLM is fine-tuned on a collection of tasks (e.g., commonsense reasoning, information extraction, etc.) with detailed instructions, and the fine-tuned LLM is expected to obtain problem-solving skills.

reasons including but not limited to overfitting, catastrophic forgetting, and task-specific biases [163]. 2) fine-tuning LLMs is computationally expensive due to the vast parameter space and the deep model architecture. In this section, we review recent techniques on how to update the global knowledge of LLMs, which can be primarily categorized into two areas: **Instruction-based Fine-tuning** and **Partial Knowledge Update** to address both challenges, respectively.

5.2.1 Instruction-based Knowledge Update. Instruction-based Knowledge Update refers to updating an LLM's parametric knowledge by fine-tuning LLMs on a diverse set of tasks with explicit instructions or prompts, which is conceptually the same as Instruct Learning introduced in [111]. An illustration of fine-tuning an LLM with instructions is provided in Figure 8, where an LLM is fine-tuned on a collection of tasks across the whole NLP application domain, and the LLM is deployed on the held-out and unseen tasks. Wei et al. f[165] provided the very first attempt to fine-tune LLMs based on a collection of datasets described via instructions. Empirically, effective instructions can substantially improve zero-shot performance on unseen tasks. The instruction-tuned language model FLAN is fine-tuned on a 137B LLM over 60 NLP datasets using natural language instruction templates. The study shows that FLAN outperforms its unmodified counterpart and even surpasses both zero-shot and few-shot 175B GPT-3 on most unseen tasks. Subsequently, in recent works by [23, 56, 98], explicit instructions have been employed to fine-tune LLMs, with emphasis placed on (1) expanding the number of tasks, (2) enlarging the model's size, and (3) fine-tuning on chain-of-thought data. As a result, the fine-tuned LLM attains state-of-the-art performance on numerous benchmarks in both zero/few-shot NLP tasks.

Fine-tuning with Human Instructions. Fine-tuning with human instructions aims to guide LLMs towards generating safer, truthful, less toxic content in line with user intentions. Most LLMs utilize autoregressive approaches, making the generated content largely influenced by the training corpus distribution and less controllable. Reinforcement learning from human feedback (RLHF) is a notable technique for aligning LLM content with human needs [21]. In RLHF: 1) LLMs create multiple content options for a prompt, ranked by humans for quality, relevance, and desired output alignment; 2) an external reward model assigns scores to content based on rankings, capturing evaluator preferences; 3) model policy is updated using reinforcement learning techniques to maximize expected reward, fine-tuning the model to better align with human preferences; 4) this process of content generation, ranking, reward modeling, and policy optimization repeats in iterations, with the model continually learning from human feedback. Existing methods successfully apply RLHF to fine-tune LLMs on complex reasoning tasks using human instructions [113, 156].

Potential Limitations of Instruction-based Knowledge Update. Knowledge updates based on explicit instructions tend to perform well on Natural Language Understanding tasks but are limited to simpler instructions and struggle with tasks diverging from evaluation sets. Improving adaptability to diverse tasks often incurs catastrophic forgetting. A crucial

question is extending model knowledge and abilities without causing such forgetting. Recently, Huang et al. proposed a method that uses a pre-trained LLM to generate high-confidence, rationale-augmented answers for unlabeled questions, improving general reasoning without ground truth labels or explicit instructions [55]. Additionally, Scialom et al. are expanding LLM knowledge and abilities without forgetting previous skills by fine-tuning LLMs across various tasks and introducing an approach to counter catastrophic forgetting with Continual Learning via Rehearsal [138, 141].

5.2.2 Partial Knowledge Update. Other than leveraging task-specific instructions to fine-tune LLMs, a number of approaches emerge to conduct LLM fine-tuning by updating/editing a part of LLM parameters that link to specific knowledge without leveraging external guidance. Suppose $f_\Theta(\cdot)$ denotes the function of LLM parametrized with the set of parameters Θ and $\theta \in \Theta$ is the single parameter in Θ. Updating the inner knowledge of $f_\Theta(\cdot)$ based on a collection of training data D is denoted as:

$$\tilde{\Theta} = \Theta + \nabla f_\Theta(D) \odot T, \qquad T^{(i)} = \begin{cases} 1, & \text{if } \theta^{(i)} \in \Theta_T \\ 0, & \text{if } \theta^{(i)} \notin \Theta_T \end{cases} \tag{1}$$

where T denotes the mask vector and $T^{(i)} \in T$ denote the i-th element of T. The mask controls the amount of LLM's inner knowledge to be updated in each fine-tuning iteration, where we use $\Theta_T \subseteq \Theta$ to denote the parameters that need to be updated in Θ. In the conventional setting of fine-tuning pre-trained language models [50, 130, 189], $|\Theta| = |\Theta_T|$. However, updating all of the parameters is computationally prohibited and resource-consuming in the context of LLM. Empirically, $|\Theta| \gg |\Theta_T|$, which refers to the modification of only a small number of parameters. Existing parameter-efficient knowledge update can be categorized into three streams: i.e., **Knowledge Editing** aims at directly locating and updating a small subset of parameters in an LLM; **Gradient Masking** aims at masking out the gradients of non-relative parameters during the fine-tuning; and **Knowledge Distillation** focuses on obtaining a child model with domain-specific knowledge from LLMs.

Knowledge Editing. Recent research has seen success in updating LLMs with new memories to replace outdated information or add specialized domain knowledge. For instance, improving the ability to update an outdated prediction like "Boris Johnson is Prime Minister of the UK" can enhance an LLM's reliability and generalization. Various methods have been proposed to locate and edit an LLM's parametric knowledge [25, 28, 48, 95, 96]. De Cao et al. proposed a hyper-network trained to update LLM parameters with a single fact needing modification, avoiding fine-tuning to prevent performance degeneration [28]. However, later works found that hyper-network-based editing falters as the LLM scales up, proposing retrieval-based methods to store edits in explicit memory and reason over them to adjust LLM predictions [103, 104]. Other methods focus on localizing and understanding LLM internal mechanisms. Notable works identify crucial neuron activations for LLM factual predictions through attention mechanisms and causal interventions, successfully updating domain facts [25, 95, 96]. A recent method is proposed learning a map from textual queries to fact encodings in an LLM's internal representation, using these encodings as knowledge editors and probes [48].

Gradient Masking. Gradient masking is a technique used to selectively update specific parts of an LLM during the fine-tuning process. The main goal is to reduce computational overhead and potentially mitigate issues such as catastrophic forgetting or overfitting, particularly when adapting pre-trained models to smaller or specialized datasets. Gradient masking involves modifying the gradients during back-propagation by applying a masking function (Equation (1)). This function determines which parts of the model will be updated, effectively *masking* the gradients for certain

parameters and keeping them unchanged. The choice of parameters to mask can be based on various criteria, such as their relevance to the task, importance in the model, or contribution to the overall loss.

Earlier attempts [62, 178] have been made to efficiently fine-tune relatively small language models by utilizing various regularization techniques, their methods cannot easily adapt to fine-tuning LLMs. This is primarily due to the substantially larger amounts of data and computational resources required to train LLMs effectively, which can be several orders of magnitude more than what is needed for smaller language models. To add gradient masks to LLMs, CHILD-TUNING [171] utilizes the downstream task data to detect the most task-related parameters as the child network and freezes the parameters in non-child network to their pre-trained weights. Moreover, Zhang et al. [181] propose a Dynamic Parameter Selection algorithm for efficiently fine-tuning LLMs, which adaptively selects a more promising sub-network to perform staging updates based on gradients of back-propagation, which brings great improvement in domain-specific downstream tasks under low-resource scenarios.

Knowledge Distillation. While most works on LLM self-knowledge update focus on task-specific instructions and parameter efficiency, a promising area of research explores distilling domain-specific knowledge from LLMs into smaller networks to reduce inference latency and enhance domain-specific task solving ability. Muhamed et al. compressed a 1.5 billion parameter LLM to a 70 million parameter model for Click-through-rate prediction, introducing twin-structured BERT-like encoders and a fusion layer for a cross-architecture distillation from a single LLM, resulting in superior performance in both online and offline settings [106]. Similarly, [4, 94, 154] employ a knowledge distillation module for LLM fine-tuning, achieving faster convergence and better resource utilization. This module leverages pre-trained parameters for quick convergence and trains a small subset of parameters to address model over-parameterization. Furthermore, [51, 142] distill the step-by-step chain-of-thought reasoning abilities of larger models into smaller models.

5.2.3 Open Challenges. Fine-tuning LLMs with the latest data ensures that they provide relevant and accurate information, especially in domains where rapid changes occur, such as technology, medicine, and current events. Furthermore, we have observed different applications or users may have unique requirements or preferences. However, fine-tuning the large-scale LLMs also poses several open challenges:

(1) *Compliance with regulations*: In most cases, updating and fine-tuning LLMs are necessary to ensure compliance with specific regulations or guidelines, such as data protection laws or industry-specific requirements. The so-called *LLM alignment* can be accomplished during the fine-tuning phase.
(2) *Computational resources*: Fine-tuning or updating inner knowledge of LLMs necessitates access to high-performance GPUs or specialized hardware, which can be expensive and difficult to obtain, particularly for individual researchers or smaller organizations. Pursuing fine-tuning efficiency is still a practical yet essential problem.

6 APPLICATIONS OF LLM DOMAIN SPECIALIZATION

In this survey paper, we explore the applications of LLMs across a range of domain-specific tasks in social sciences (e.g., education, finance, law), natural sciences (e.g., biomedicine, earth science), and formal sciences (e.g., human-computer interaction, software engineering, and cyber security). To achieve domain specialization for LLMs in these diverse fields, readers can employ various techniques, such as external augmentation, instruction crafting, and knowledge update. These approaches can help tailor LLMs to specific tasks and challenges in each domain, enabling more accurate, relevant, and effective applications. Although each domain has its unique challenges and requirements, several common applications of specialized LLMs are shared across these fields:

- *Advanced information extraction*: They can identify entities, relationships, and events from domain-specific texts, such as recognizing genes in biomedical literature or detecting legal clauses in contracts.
- *Text generation and summarization*: They can generate high-quality, domain-specific content and create accurate summaries of complex domain-specific texts.
- *Data-driven predictions and recommendations*: They can analyze domain-specific data for forecasting and providing recommendations, like predicting financial trends or suggesting personalized medical treatment plans.
- *Conversational agents and expert systems*: They can be incorporated into conversational agents or expert systems for domain-specific guidance, such as virtual tutors or legal chatbots.
- *Automated code generation and analysis*: In software engineering, they can generate or analyze code, identify bugs, or suggest improvements based on natural language descriptions.

In this section, we dive deep to review existing techniques for specializing LLMs in domain-specific tasks and discuss potential open challenges in detail. Due to the space limitation, we only provide a brief introduction of each domain and leave the complete discussion in the supplementary material.

Biomedicine. Language models are becoming increasingly useful in the field of biology, from fundamental biomedical research [91, 132] to clinical healthcare support [58, 105, 127]. At the fundamental biomedicine science level, LLMs can be trained on vast amounts of domain-specializing data (e.g., genomic and proteomic) to analyze and predict biological functions, disease mechanisms, and drug discovery. LLMs can also aid in predicting protein structures and interactions, which are critical for understanding cellular processes and designing new drugs. At the clinical healthcare support level, pre-trained or medical corpus fine-tuned LLMs can be used for the natural language processing of medical records to identify patterns, make diagnoses, and provide personalized treatment recommendations. Also, LLMs can assist in medical image analysis in a multi-modality learning way, such as identifying specific features in X-rays or MRI scans. Overall, LLMs offer tremendous potential for advancing biology research and improving healthcare outcomes.

Earth Science. Earth science is an interdisciplinary domain focused on examining the interactions between physical and human systems across diverse spatial and temporal scales. This field incorporates methods from Earth observation, information science, spatial analysis, complexity theory, and simulation modeling to investigate phenomena like climate change, land-use change, natural disasters, environmental development, and urbanization. Spatial information is vital to Earth science, and geographic information science tools are invaluable for a wide range of interdisciplinary studies involving spatial data. Large language models like ChatGPT can act as question-answer systems, assisting those interested in Earth Science to gain pertinent knowledge, such as recommending the optimal earth observation dataset for specific research purposes, offering code examples like Google Earth Engine code for processing satellite data, providing high-quality responses to environmental-related questions [188], developing innovative idea [9], generating climate scenario [9]. LLMs can also be tailored to various Earth Science-related downstream tasks through methods such as fine-tuning, few-shot, or even zero-shot learning.

Finance and Law. Specializing LLMs in the financial and legal domains requires careful adaptation to the distinctive characteristics of these fields. In the financial domain [71, 87, 169, 174], models need to comprehend complex financial terminologies, economic trends, and regulatory norms to accurately generate content like financial reports, investment analyses, or risk assessments. Meanwhile, the legal domain [15, 120, 151] demands understanding and generation of intricate legal language, comprehension of laws, legal codes, and court rulings, while maintaining absolute precision and a xxformal tone. For both domains, model specialization often involves fine-tuning with domain-specific datasets,

incorporating explicit domain knowledge, and optimizing for domain-specific objectives like compliance with regulations, accuracy of information, or effectiveness of advice. However, it's crucial to maintain an ethical guardrail for these models, given the high stakes nature of both financial and legal decisions. The specialized models also need to keep abreast of the evolving landscapes of these domains, adapting to changes in laws, regulations, or financial trends.

Human Computer Interaction and Software Engineering. Specializing LLMs in the domains of human-computer interaction (HCI) and software engineering requires a deep understanding of the terminologies, workflows, and conventions unique to these areas. In the HCI domain, an LLM may be specialized to understand and respond to user inputs more effectively, potentially improving the design and usability of interfaces by offering more natural and intuitive interaction paradigms. This involves training the model on diverse data, ranging from human conversational data to user interaction logs. On the other hand, in the software engineering domain, an LLM can be specialized to aid in tasks such as code generation, bug detection, code review, and documentation. This involves training the model on large codebases, issue trackers, documentation, and other software-related data. These specialized models can provide valuable assistance to developers, enhance the software development process, and potentially revolutionize the way we interact with computers. Despite the promising applications, several challenges remain, including the complexity of these domains, the need for accurate and up-to-date data, and the balance between specialized and general knowledge.

7 OPEN CHALLENGES AND FUTURE WORKS

7.1 Open Challenges

After having explored the current approaches for domain specialization of large language models, categorized as white box, grey box, and black box methods, it is essential to acknowledge that despite the significant progress in this field, there remain several open challenges. These challenges pervade all categories of models, irrespective of their accessibility or the specific techniques employed for specialization. As we strive to create LLMs that can effectively understand and generate domain-specific content, it is these challenges that will shape the future trajectory of research in this field. Let us delve into these open challenges to better comprehend the complexities of domain specialization and the areas where further research is required.

- *Domain Complexity*: Each domain has its unique intricacies and complexities, which could range from highly specialized vocabularies, and nuanced terminologies to complex knowledge structures. For instance, the legal or medical field employs language and terms that are extremely domain-specific and follow certain syntax and structure rules. This complexity extends to the relationships between different entities and concepts within the domain. Accurately understanding and modeling this intricate domain knowledge is a significant challenge for all types of models.
- *Balancing General and Domain Knowledge*: An LLM, while needing to understand the specificities of a particular domain, also has to maintain its general knowledge to provide contextually appropriate responses. If a model is overly specialized, it may perform exceptionally within the targeted domain but fail to understand or generate coherent responses to prompts outside of it. Conversely, retaining too much general knowledge may dilute the domain-specific responses. Striking this balance between general and domain knowledge is a complex task.
- *Explainability and Trust*: As LLMs become more sophisticated, their decision-making process also becomes more opaque, raising the challenge of explainability. It is crucial for users, especially in high-stakes domains like healthcare, law, or finance, to understand how the model arrived at a certain output. Achieving this transparency

can help build trust in the system. The challenge lies in the trade-off between model complexity and explainability, as increasing one often decreases the other.

- *Adapting to Domain Evolution*: Domains are not static; they evolve over time with the introduction of new terminologies, concepts, and trends. For example, the ongoing COVID-19 pandemic introduced a slew of new medical terms and concepts. Therefore, an LLM that is specialized for a certain domain must continuously adapt to these changes to stay relevant and effective. Designing models that can keep pace with the evolving landscape of their specialized domain is a challenging task.
- *Scalability*: Domain specialization often involves training or fine-tuning the LLM with domain-specific data, crafting specific prompts, or using other domain-specific resources. While this might be feasible for a few domains, scaling this process to cover a wide range of domains or to handle large, complex domains is a significant challenge. It involves not just computational resources but also the availability of domain-specific data and expertise. The challenge is to create efficient and effective methods for domain specialization that can be scaled to cover many different domains.

7.2   Future Works

As we chart the frontier of large language model specialization, it's not only crucial to build upon and refine the existing black-box, grey-box, and white-box methods, but also to anticipate and explore innovative, out-of-the-box techniques that have the potential to transcend these conventional approaches. Leveraging the rapid advancement of AI technologies and the increased understanding of LLMs, we can envision a future where novel techniques will emerge that push the boundaries of what's possible in domain specialization, providing enhanced performance, greater flexibility, and more efficient utilization of resources.

- *Hybrid Approaches*: This could involve combining multiple methods depending on the stage or specific needs. For example, a model could start with a black-box approach, using external resources to augment input prompts, then proceed to a grey-box method where gradients or loss values are used to further refine the prompts, and finally employ a white-box approach to fine-tune the model based on the learned strategies and feedback. This hybrid approach could provide a balance between resource requirement and model performance and might be especially effective when dealing with scarce domain-specific data.
- *Meta-Learning or AutoML Techniques*: AutoML or meta-learning strategies could be used to automate the process of selecting the best strategies for domain specialization. For instance, a meta-learning approach might learn a policy to select the best data for fine-tuning, the best prompt engineering techniques, or the best layers to fine-tune for a given domain, based on previous experience with similar domains. This could significantly reduce the resources and expertise needed for domain specialization, and could potentially lead to more effective and efficient methods.
- *Incorporating More Explicit World Knowledge*: Instead of relying solely on text-based pre-training, future LLMs might leverage structured knowledge sources, like knowledge graphs, to augment their understanding of the domain. This could involve techniques like graph neural networks or attention mechanisms that operate on graph-structured data. For instance, a medical LLM could incorporate knowledge from a medical ontology graph to better understand the relationships between various medical terms and concepts. This could lead to more accurate and informative outputs, especially in domains where explicit structured knowledge is available.

- *Human-in-the-loop Learning*: This involves continuous interaction and feedback from human users or experts to guide the model's learning process. For instance, a legal LLM could be continuously updated based on feedback from legal professionals using the model. This feedback could be incorporated in the form of additional training data, changes to the model's reward function in a reinforcement learning framework, or modifications to the model's prompts. This could lead to a more dynamic and adaptable model that can evolve with the needs and knowledge of the users.
- *Active Learning*: This approach involves the model actively querying for information or feedback when it encounters a domain-specific concept it doesn't understand or has low confidence about. For instance, if a model trained on general news articles encounters a specialized medical term it doesn't understand, it could query a medical ontology or ask for clarification from a human user. The model could then incorporate this new information into its subsequent responses. This could make the model more effective at handling unfamiliar domain-specific topics, and could provide a more interactive and engaging user experience.

Each of these techniques provides a promising direction for future research in the domain specialization of large language models, and could help address some of the challenges and limitations of the current black-box, grey-box, and white-box methods.

8 CONCLUSION

In conclusion, the rapid advancement of LLMs has sparked significant interest in harnessing their potential to tackle domain-specific tasks in various natural, social, and formal science fields. However, several challenges, such as limited domain-specific expertise, knowledge elicitation, and model complexity, hinder the direct application of LLMs in these domains. This survey systematically categorizes and summarizes existing domain specialization techniques based on their access level to LLMs, along with a comprehensive overview of application domains that can benefit from specialized LLMs. By offering a detailed analysis of the advantages, disadvantages, and relationships among different techniques and domains, this survey aims to assist domain experts in identifying suitable techniques for their target problem settings, while also providing data scientists with a clear understanding of the practical significance and open challenges in various application domains. Moreover, the paper highlights the current status of research in this area, shedding light on future trends and potential avenues for interdisciplinary collaboration. As the field of LLM domain specialization continues to evolve, this survey serves as a valuable resource for researchers and practitioners, fostering further advancements and innovations in the application of artificial intelligence across diverse domains.

REFERENCES

[1] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. 2019. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*. PMLR, 242–252.

[2] Simran Arora, Avanika Narayan, Mayee F Chen, Laurel Orr, Neel Guha, Kush Bhatia, Ines Chami, and Christopher Re. 2023. Ask Me Anything: A simple strategy for prompting language models. In *The Eleventh International Conference on Learning Representations*.

[3] Akari Asai, Mohammadreza Salehi, Matthew E Peters, and Hannaneh Hajishirzi. 2022. Attempt: Parameter-efficient multi-task tuning via attentional mixtures of soft prompts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 6655–6672.

[4] Zhangir Azerbayev, Ansong Ni, Hailey Schoelkopf, and Dragomir Radev. 2022. Explicit Knowledge Transfer for Weakly-Supervised Code Generation. *arXiv preprint arXiv:2211.16740* (2022).

[5] Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023* (2023).

[6] Ankur Bapna, Naveen Arivazhagan, and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. *arXiv preprint arXiv:1909.08478* (2019).

[7] Eyal Ben-David, Nadav Oved, and Roi Reichart. 2022. PADA: Example-based Prompt Learning for on-the-fly Adaptation to Unseen Domains. *Transactions of the Association for Computational Linguistics* 10 (2022), 414–433.

[8] Rishabh Bhardwaj, Amrita Saha, and Steven C. H. Hoi. 2022. Vector-Quantized Input-Contextualized Soft Prompts for Natural Language Understanding. In *Conference on Empirical Methods in Natural Language Processing*.

[9] Som S Biswas. 2023. Potential Use of Chat GPT in Global Warming. *Annals of Biomedical Engineering* (2023), 1–2.

[10] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*. 2206–2240.

[11] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[12] Lucas Caccia, Edoardo Ponti, Lucas Liu, Matheus Pereira, Nicolas Le Roux, and Alessandro Sordoni. 2022. Multi-Head Adapter Routing for Data-Efficient Fine-Tuning. *arXiv preprint arXiv:2211.03831* (2022).

[13] Yihan Cao, Siyu Li, Yixin Liu, Zhiling Yan, Yutong Dai, Philip S Yu, and Lichao Sun. 2023. A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt. *arXiv preprint arXiv:2303.04226* (2023).

[14] Yekun Chai, Shuohuan Wang, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2022. Clip-Tuning: Towards Derivative-free Prompt Learning with a Mixture of Rewards. In *Conference on Empirical Methods in Natural Language Processing*.

[15] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The Muppets straight out of Law School. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. 2898–2904.

[16] Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. *arXiv preprint arXiv:2004.12651* (2020).

[17] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2022. Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks. *arXiv preprint arXiv:2211.12588* (2022).

[18] Xiang Chen, Ningyu Zhang, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2021. KnowPrompt: Knowledge-aware Prompt-tuning with Synergistic Optimization for Relation Extraction. *Proceedings of the ACM Web Conference 2022* (2021).

[19] Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Binding Language Models in Symbolic Languages. In *The Eleventh International Conference on Learning Representations*.

[20] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311* (2022).

[21] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems* 30 (2017).

[22] Alexandra Chronopoulou, Matthew E Peters, Alexander Fraser, and Jesse Dodge. 2023. AdapterSoup: Weight Averaging to Improve Generalization of Pretrained Language Models. *arXiv preprint arXiv:2302.07027* (2023).

[23] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416* (2022).

[24] Hejie Cui, Jiaying Lu, Shiyu Wang, Ran Xu, Wenjing Ma, Shaojun Yu, Yue Yu, Xuan Kan, Chen Ling, Joyce Ho, et al. 2023. A Survey on Knowledge Graphs for Healthcare: Resources, Applications, and Promises. *arXiv preprint arXiv:2306.04802* (2023).

[25] Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2021. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696* (2021).

[26] Zhuyun Dai, Vincent Y Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B Hall, and Ming-Wei Chang. 2022. Promptagator: Few-shot dense retrieval from 8 examples. *arXiv preprint arXiv:2209.11755* (2022).

[27] Ishita Dasgupta, Christine Kaeser-Chen, Kenneth Marino, Arun Ahuja, Sheila Babayan, Felix Hill, and Rob Fergus. 2023. Collaborating with language models for embodied reasoning. In *Second Workshop on Language and Reinforcement Learning*.

[28] Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing Factual Knowledge in Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

[29] Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P. Xing, and Zhiting Hu. 2022. RLPrompt: Optimizing Discrete Text Prompts with Reinforcement Learning. In *Conference on Empirical Methods in Natural Language Processing*.

[30] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[31] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv* (2022).

[32] Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. 2023. Compositional Semantic Parsing with Large Language Models. In *The Eleventh International Conference on Learning Representations*.

[33] Dheeru Dua, Shivanshu Gupta, Sameer Singh, and Matt Gardner. 2022. Successive Prompting for Decomposing Complex Questions. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

[34] Ali Edalati, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. 2022. KronA: Parameter Efficient Tuning with Kronecker Adapter. *arXiv preprint arXiv:2212.10650* (2022).

[35] Mohammad Fahes, Tuan-Hung Vu, Andrei Bursuc, Patrick Pérez, and Raoul de Charette. 2022. P{\O}DA: Prompt-driven Zero-shot Domain Adaptation. *arXiv preprint arXiv:2212.03241* (2022).

[36] Yang Feng, Shiyue Zhang, Andi Zhang, Dong Wang, and Andrew Abel. 2017. Memory-augmented Neural Machine Translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

[37] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2022. PAL: Program-aided Language Models. *arXiv preprint arXiv:2211.10435* (2022).

[38] Chunjiang Ge, Rui Huang, Mixue Xie, Zihang Lai, Shiji Song, Shuang Li, and Gao Huang. 2022. Domain Adaptation via Prompt Learning. *arXiv preprint arXiv:2202.06687* (2022).

[39] Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017. Improving neural language models with a continuous cache. In *International Conference on Learning Representations*.

[40] Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2022. PPT: Pre-trained Prompt Tuning for Few-shot Learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Dublin, Ireland, 8410–8423.

[41] Xu Guo, Boyang Li, and Han Yu. 2022. Improving the Sample Efficiency of Prompt Tuning with Domain Adaptation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates.

[42] Xu Guo and Han Yu. 2022. On the Domain Adaptation and Generalization of Pretrained Language Models: A Survey. *arXiv* (2022).

[43] Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: Word-level Adversarial ReProgramming. In *Annual Meeting of the Association for Computational Linguistics*.

[44] Hangfeng He, Hongming Zhang, and Dan Roth. 2022. Rethinking with Retrieval: Faithful Large Language Model Inference. *arXiv preprint arXiv:2301.00303* (2022).

[45] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366* (2021).

[46] Shwai He, Liang Ding, Daize Dong, Miao Zhang, and Dacheng Tao. 2022. Sparseadapter: An easy approach for improving the parameter-efficiency of adapters. *arXiv preprint arXiv:2210.04284* (2022).

[47] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* (2016).

[48] Evan Hernandez, Belinda Z Li, and Jacob Andreas. 2023. Measuring and Manipulating Knowledge Representations in Language Models. *arXiv preprint arXiv:2304.00740* (2023).

[49] 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.

[50] Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146* (2018).

[51] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander 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. *arXiv preprint arXiv:2305.02301* (2023).

[52] 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).

[53] Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Jingang Wang, Juanzi Li, Wei Wu, and Maosong Sun. 2022. Knowledgeable Prompt-tuning: Incorporating Knowledge into Prompt Verbalizer for Text Classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Dublin, Ireland.

[54] Zhiqiang Hu, Yihuai Lan, Lei Wang, Wanyu Xu, Ee-Peng Lim, Roy Ka-Wei Lee, Lidong Bing, and Soujanya Poria. 2023. LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models. *arXiv preprint arXiv:2304.01933* (2023).

[55] Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *arXiv preprint arXiv:2210.11610* (2022).

[56] Shaohan Huang, Li Dong, Wenhui Wang, Yaru Hao, Saksham Singhal, Shuming Ma, Tengchao Lv, Lei Cui, Owais Khan Mohammed, Qiang Liu, et al. 2023. Language is not all you need: Aligning perception with language models. *arXiv preprint arXiv:2302.14045* (2023).

[57] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299* (2022).

[58] Katharina Jeblick, Balthasar Schachtner, Jakob Dexl, Andreas Mittermeier, Anna Theresa Stüber, Johanna Topalis, Tobias Weber, Philipp Wesp, Bastian Sabel, Jens Ricke, et al. 2022. ChatGPT Makes Medicine Easy to Swallow: An Exploratory Case Study on Simplified Radiology Reports. *arXiv preprint arXiv:2212.14882* (2022).

[59] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *Comput. Surveys* 55, 12 (2023), 1–38.

[60] Chen Jia and Yue Zhang. 2022. Prompt-based Distribution Alignment for Domain Generalization in Text Classification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 10147–10157.

[61] Albert Qiaochu Jiang, Sean Welleck, Jin Peng Zhou, Timothee Lacroix, Jiacheng Liu, Wenda Li, Mateja Jamnik, Guillaume Lample, and Yuhuai Wu. 2023. Draft, Sketch, and Prove: Guiding Formal Theorem Provers with Informal Proofs. In *The Eleventh International Conference on Learning Representations*.

[62] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2019. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *arXiv preprint arXiv:1911.03437* (2019).

[63] Feihu Jin, Jinliang Lu, Jiajun Zhang, and Chengqing Zong. 2022. Instance-aware prompt learning for language understanding and generation. *arXiv preprint arXiv:2201.07126* (2022).

[64] Qiao Jin, Yifan Yang, Qingyu Chen, and Zhiyong Lu. [n. d.]. GeneGPT: Augmenting Large Language Models with Domain Tools for Improved Access to Biomedical Information. *ArXiv* ([n. d.]).

[65] 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* (2020).

[66] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems* 34 (2021), 1022–1035.

[67] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406* (2022).

[68] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large Language Models are Zero-Shot Reasoners. In *ICML 2022 Workshop on Knowledge Retrieval and Language Models*.

[69] Mojtaba Komeili, Kurt Shuster, and Jason Weston. 2022. Internet-Augmented Dialogue Generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 8460–8478.

[70] Eric Lehman, Evan Hernandez, Diwakar Mahajan, Jonas Wulff, Micah J Smith, Zachary Ziegler, Daniel Nadler, Peter Szolovits, Alistair Johnson, and Emily Alsentzer. 2023. Do We Still Need Clinical Language Models? *arXiv preprint arXiv:2302.08091* (2023).

[71] Markus Leippold. 2023. Sentiment Spin: Attacking Financial Sentiment with GPT-3. *Available at SSRN* (2023).

[72] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 3045–3059.

[73] Yoav Levine, Itay Dalmedigos, Ori Ram, Yoel Zeldes, Daniel Jannai, Dor Muhlgay, Yoni Osin, Opher Lieber, Barak Lenz, Shai Shalev-Shwartz, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2022. Standing on the Shoulders of Giant Frozen Language Models. *ArXiv* abs/2204.10019 (2022).

[74] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.

[75] Daliang Li, Ankit Singh Rawat, Manzil Zaheer, Xin Wang, Michal Lukasik, Andreas Veit, Felix Yu, and Sanjiv Kumar. 2022. Large Language Models with Controllable Working Memory. *arXiv preprint arXiv:2211.05110* (2022).

[76] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. CAMEL: Communicative Agents for" Mind" Exploration of Large Scale Language Model Society. *arXiv preprint arXiv:2303.17760* (2023).

[77] Haochen Li, Tong Mo, Hongcheng Fan, Jingkun Wang, Jiaxi Wang, Fuhao Zhang, and Weiping Li. 2022. KiPT: Knowledge-injected Prompt Tuning for Event Detection. In *International Conference on Computational Linguistics*.

[78] Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, et al. 2023. Can LLM Already Serve as A Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs. *arXiv preprint arXiv:2305.03111* (2023).

[79] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. *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)* abs/2101.00190 (2021).

[80] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Pete Florence, Andy Zeng, et al. 2022. Code as Policies: Language Model Programs for Embodied Control. In *Workshop on Language and Robotics at CoRL 2022*.

[81] Yaobo Liang, Chenfei Wu, Ting Song, Wenshan Wu, Yan Xia, Yu Liu, Yang Ou, Shuai Lu, Lei Ji, Shaoguang Mao, et al. 2023. TaskMatrix. AI: Completing Tasks by Connecting Foundation Models with Millions of APIs. *arXiv preprint arXiv:2303.16434* (2023).

[82] Hongzhan Lin, Pengyao Yi, Jing Ma, Haiyun Jiang, Ziyang Luo, Shuming Shi, and Ruifang Liu. 2022. Zero-Shot Rumor Detection with Propagation Structure via Prompt Learning. *arXiv preprint arXiv:2212.01117* (2022).

[83] Qi Liu, Dani Yogatama, and Phil Blunsom. 2022. Relational Memory-Augmented Language Models. *Transactions of the Association for Computational Linguistics* 10 (2022), 555–572.

[84] Ruibo Liu, Jason Wei, Shixiang Shane Gu, Te-Yen Wu, Soroush Vosoughi, Claire Cui, Denny Zhou, and Andrew M Dai. 2022. Mind's Eye: Grounded Language Model Reasoning through Simulation. *arXiv preprint arXiv:2210.05359* (2022).

[85] Xiangyang Liu, Tianxiang Sun, Xuanjing Huang, and Xipeng Qiu. 2022. Late Prompt Tuning: A Late Prompt Could Be Better Than Many Prompts. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates.

[86] Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, et al. 2023. Summary of chatgpt/gpt-4 research and perspective towards the future of large language models. *arXiv preprint arXiv:2304.01852* (2023).

[87] Alejandro Lopez-Lira and Yuehua Tang. 2023. Can ChatGPT Forecast Stock Price Movements? Return Predictability and Large Language Models. *arXiv preprint arXiv:2304.07619* (2023).

[88] Jiaying Lu, Jiaming Shen, Bo Xiong, Wengjing Ma, Staab Steffen, and Carl Yang. 2023. HiPrompt: Few-Shot Biomedical Knowledge Fusion via Hierarchy-Oriented Prompting. In *46th International ACM SIGIR Conference on Research and Development in Information Retrieval - Short Paper*.

[89] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 8086–8098.

[90] Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. 2022. Language models of code are few-shot commonsense learners. *arXiv preprint arXiv:2210.07128* (2022).

[91] Babak Mahjour, Jillian Hoffstadt, and Tim Cernak. 2023. Designing Chemical Reaction Arrays using phactor and ChatGPT. (2023).

[92] Bhavitvya Malik, Abhinav Ramesh Kashyap, Min-Yen Kan, and Soujanya Poria. 2023. UDApter–Efficient Domain Adaptation Using Adapters. *arXiv preprint arXiv:2302.03194* (2023).

[93] Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Wen-tau Yih, and Madian Khabsa. 2021. Unipelt: A unified framework for parameter-efficient language model tuning. *arXiv preprint arXiv:2110.07577* (2021).

[94] Raja Marjieh, Ilia Sucholutsky, Pol van Rijn, Nori Jacoby, and Thomas L Griffiths. 2023. What Language Reveals about Perception: Distilling Psychophysical Knowledge from Large Language Models. *arXiv preprint arXiv:2302.01308* (2023).

[95] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems* 35 (2022), 17359–17372.

[96] Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229* (2022).

[97] Jacob Menick, Maja Trebacz, Vladimir Mikulik, John Aslanides, Francis Song, Martin Chadwick, Mia Glaese, Susannah Young, Lucy Campbell-Gillingham, Geoffrey Irving, et al. 2022. Teaching language models to support answers with verified quotes. *arXiv preprint arXiv:2203.11147* (2022).

[98] Jacob Menick, Maja Trebacz, Vladimir Mikulik, John Aslanides, Francis Song, Martin Chadwick, Mia Glaese, Susannah Young, Lucy Campbell-Gillingham, Geoffrey Irving, et al. 2022. Teaching language models to support answers with verified quotes. *arXiv preprint arXiv:2203.11147* (2022).

[99] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

[100] Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, et al. 2023. Augmented language models: a survey. *arXiv preprint arXiv:2302.07842* (2023).

[101] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heinz, and Dan Roth. 2021. Recent advances in natural language processing via large pre-trained language models: A survey. *arXiv* (2021).

[102] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? *arXiv preprint arXiv:2202.12837* (2022).

[103] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309* (2021).

[104] Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022. Memory-based model editing at scale. In *International Conference on Machine Learning*. 15817–15831.

[105] Philip Moons and Liesbet Van Bulck. 2023. ChatGPT: Can artificial intelligence language models be of value for cardiovascular nurses and allied health professionals. *European Journal of Cardiovascular Nursing* (2023), zvad022–zvad022.

[106] Aashiq Muhamed, Iman Keivanloo, Sujan Perera, James Mracek, Yi Xu, Qingjun Cui, Santosh Rajagopalan, Belinda Zeng, and Trishul Chilimbi. 2021. CTR-BERT: Cost-effective knowledge distillation for billion-parameter teacher models. In *NeurIPS Efficient Natural Language and Speech Processing Workshop*.

[107] Prasanth Murali, Ian Steenstra, Hye Sun Yun, Ameneh Shamekhi, and Timothy Bickmore. 2023. Improving Multiparty Interactions with a Robot Using Large Language Models. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–8.

[108] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332* (2021).

[109] OpenAI. [n. d.]. ChatGPT plugins. https://openai.com/blog/chatgpt-plugins. Accessed: 2023-04-05.

[110] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[111] 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.

[112] Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative Agents: Interactive Simulacra of Human Behavior. *arXiv preprint arXiv:2304.03442* (2023).

[113] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277* (2023).

[114] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2020. AdapterFusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247* (2020).

[115] Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. Adapterhub: A framework for adapting transformers. *arXiv preprint arXiv:2007.07779* (2020).

[116] Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. Mad-x: An adapter-based framework for multi-task cross-lingual transfer. *arXiv preprint arXiv:2005.00052* (2020).

[117] Edoardo M Ponti, Alessandro Sordoni, and Siva Reddy. 2022. Combining modular skills in multitask learning. *arXiv preprint arXiv:2202.13914* (2022).

[118] Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. *arXiv preprint arXiv:2304.11015* (2023).

[119] Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. 2022. GrIPS: Gradient-free, Edit-based Instruction Search for Prompting Large Language Models. In *Conference of the European Chapter of the Association for Computational Linguistics*.

[120] Nishchal Prasad, Mohand Boughanem, and Taoufiq Dkaki. 2022. Effect of Hierarchical Domain-specific Language Models and Attention in the Classification of Decisions for Legal Cases. In *Proceedings of the CIRCLE (Joint Conference of the Information Retrieval Communities in Europe), Samatan, Gers, France.* 4–7.

[121] Chengwei Qin and Shafiq Joty. 2021. LFPT5: A unified framework for lifelong few-shot language learning based on prompt tuning of t5. *arXiv preprint arXiv:2110.07298* (2021).

[122] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, et al. 2023. Tool Learning with Foundation Models. *arXiv preprint arXiv:2304.08354* (2023).

[123] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences* 63, 10 (2020), 1872–1897.

[124] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[125] 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. *The Journal of Machine Learning Research* 21, 1 (2020), 5485–5551.

[126] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-Context Retrieval-Augmented Language Models. *arXiv preprint arXiv:2302.00083* (2023).

[127] Arya Rao, John Kim, Meghana Kamineni, Michael Pang, Winston Lie, and Marc D Succi. 2023. Evaluating ChatGPT as an adjunct for radiologic decision-making. *medRxiv* (2023), 2023–02.

[128] Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2023. Progressive Prompts: Continual Learning for Language Models. In *The Eleventh International Conference on Learning Representations*.

[129] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems* 30 (2017).

[130] Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910* (2020).

[131] Joshua Robinson, Christopher Michael Rytting, and David Wingate. 2022. Leveraging Large Language Models for Multiple Choice Question Answering. *arXiv* (2022).

[132] Jerret Ross, Brian Belgodere, Vijil Chenthamarakshan, Inkit Padhi, Youssef Mroueh, and Payel Das. 2022. Large-scale chemical language representations capture molecular structure and properties. *Nature Machine Intelligence* 4, 12 (2022), 1256–1264.

[133] Malik Sallam. 2023. The utility of ChatGPT as an example of large language models in healthcare education, research and practice: Systematic review on the future perspectives and potential limitations. *medRxiv* (2023), 2023–02.

[134] Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. 2022. Multitask Prompted Training Enables Zero-Shot Task Generalization. In *ICLR*.

[135] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761* (2023).

[136] Timo Schick and Hinrich Schütze. 2020. Exploiting cloze questions for few shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676* (2020).

[137] Dale Schuurmans. 2023. Memory Augmented Large Language Models are Computationally Universal. *arXiv preprint arXiv:2301.04589* (2023).

[138] Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. 2022. Fine-tuned Language Models are Continual Learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 6107–6122.

[139] Shohreh Shaghaghian, Luna Yue Feng, Borna Jafarpour, and Nicolai Pogrebnyakov. 2020. Customizing contextualized language models for legal document reviews. In *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2139–2148.

[140] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in HuggingFace. arXiv:2303.17580

[141] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. *Advances in neural information processing systems* 30 (2017).

[142] Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2022. Distilling multi-step reasoning capabilities of large language models into smaller models via semantic decompositions. *arXiv preprint arXiv:2212.00193* (2022).

[143] Devendra Singh, Siva Reddy, Will Hamilton, Chris Dyer, and Dani Yogatama. 2021. End-to-end training of multi-document reader and retriever for open-domain question answering. *Advances in Neural Information Processing Systems* 34 (2021), 25968–25981.

[144] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. [n. d.]. ProgPrompt: Generating Situated Robot Task Plans using Large Language Models. In *Workshop on Language and Robotics at CoRL 2022*.

[145] Yusheng Su, Xiaozhi Wang, Yujia Qin, Chi-Min Chan, Yankai Lin, Huadong Wang, Kaiyue Wen, Zhiyuan Liu, Peng Li, Juanzi Li, et al. 2022. On transferability of prompt tuning for natural language processing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 3949–3969.

[146] Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. 2022. Black-Box Tuning for Language-Model-as-a-Service. In *International Conference on Machine Learning*.

[147] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. *arXiv preprint arXiv:2206.06522* (2022).

[148] Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. Vipergpt: Visual inference via python execution for reasoning. *arXiv preprint arXiv:2303.08128* (2023).

[149] 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).

[150] Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2022. DyLoRA: Parameter Efficient Tuning of Pre-trained Models using Dynamic Search-Free Low-Rank Adaptation. *arXiv preprint arXiv:2210.07558* (2022).

[151] Josef Valvoda, Ryan Cotterell, and Simone Teufel. 2023. On the Role of Negative Precedent in Legal Outcome Prediction. *Transactions of the Association for Computational Linguistics* 11 (2023), 34–48.

[152] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[153] Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Matthew Cer. 2021. SPoT: Better Frozen Model Adaptation through Soft Prompt Transfer. *ArXiv* abs/2110.07904 (2021).

[154] Danilo Vucetic, Mohammadreza Tayaranian, Maryam Ziaeefard, James J Clark, Brett H Meyer, and Warren J Gross. 2022. Efficient Fine-Tuning of Compressed Language Models with Learners. *arXiv preprint arXiv:2208.02070* (2022).

[155] Zhongwei Wan, Yichun Yin, Wei Zhang, Jiaxin Shi, Lifeng Shang, Guangyong Chen, Xin Jiang, and Qun Liu. 2022. G-MAP: General Memory-Augmented Pre-trained Language Model for Domain Tasks. In *2022 Conference on Empirical Methods in Natural Language Processing*.

[156] Longyue Wang, Chenyang Lyu, Tianbo Ji, Zhirui Zhang, Dian Yu, Shuming Shi, and Zhaopeng Tu. 2023. Document-level machine translation with large language models. *arXiv preprint arXiv:2304.02210* (2023).

[157] Ruijie Wang, Zheng Li, Dachun Sun, Shengzhong Liu, Jinning Li, Bing Yin, and Tarek Abdelzaher. 2022. Learning to sample and aggregate: Few-shot reasoning over temporal knowledge graphs. In *Advances in Neural Information Processing Systems*.

[158] Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Guihong Cao, Daxin Jiang, Ming Zhou, et al. 2020. K-adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint arXiv:2002.01808* (2020).

[159] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Rationale-augmented ensembles in language models. *arXiv preprint arXiv:2207.00747* (2022).

[160] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations*.

[161] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-Instruct: Aligning Language Model with Self Generated Instructions.

[162] Yaqing Wang, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. Adamix: Mixture-of-adapter for parameter-efficient tuning of large language models. *arXiv preprint arXiv:2205.12410* (2022).

[163] Yihan Wang, Si Si, Daliang Li, Michal Lukasik, Felix Yu, Cho-Jui Hsieh, Inderjit S Dhillon, and Sanjiv Kumar. 2022. Preserving In-Context Learning ability in Large Language Model Fine-tuning. *arXiv preprint arXiv:2211.00635* (2022).

[164] Zihao Wang, Shaofei Cai, Anji Liu, Xiaojian Ma, and Yitao Liang. 2023. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560* (2023).

[165] 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*.

[166] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682* (2022).

[167] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed H Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*.

[168] Stephen Wolfram. [n. d.]. ChatGPT Gets Its "Wolfram Superpowers"! https://writings.stephenwolfram.com/2023/03/chatgpt-gets-its-wolfram-superpowers/. Accessed: 2023-03-27.

[169] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564* (2023).

[170] Zhuofeng Wu, Sinong Wang, Jiatao Gu, Rui Hou, Yuxiao Dong, V. G. Vinod Vydiswaran, and Hao Ma. 2022. IDPG: An Instance-Dependent Prompt Generation Method. In *North American Chapter of the Association for Computational Linguistics*.

[171] Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. 2021. Raise a child in large language model: Towards effective and generalizable fine-tuning. *arXiv preprint arXiv:2109.05687* (2021).

[172] Jingfeng Yang, Haoming Jiang, Qingyu Yin, Danqing Zhang, Bing Yin, and Diyi Yang. 2022. SEQZERO: Few-shot Compositional Semantic Parsing with Sequential Prompts and Zero-shot Models. In *Findings of the Association for Computational Linguistics: NAACL 2022*.

[173] Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Bing Yin, and Xia Hu. 2023. Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond. *arXiv* (2023).

[174] Kai-Cheng Yang and Filippo Menczer. 2023. Large language models can rate news outlet credibility. *arXiv preprint arXiv:2304.00228* (2023).

[175] Xianjun Yang, Wei Cheng, Xujiang Zhao, Linda Petzold, and Haifeng Chen. 2023. Dynamic Prompting: A Unified Framework for Prompt Tuning. *ArXiv* abs/2303.02909 (2023).

[176] Zonghan Yang, Xiaoyuan Yi, Peng Li, Yang Liu, and Xing Xie. 2022. Unified Detoxifying and Debiasing in Language Generation via Inference-time Adaptive Optimization. *arXiv preprint arXiv:2210.04492* (2022).

[177] Hongbin Ye, Ningyu Zhang, Shumin Deng, Xiang Chen, Hui Chen, Feiyu Xiong, Xi Chen, and Huajun Chen. 2022. Ontology-enhanced Prompt-tuning for Few-shot Learning. *Proceedings of the ACM Web Conference 2022* (2022).

[178] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199* (2021).

[179] Aston Zhang, Yi Tay, Shuai Zhang, Alvin Chan, Anh Tuan Luu, Siu Cheung Hui, and Jie Fu. 2021. Beyond fully-connected layers with quaternions: Parameterization of hypercomplex multiplications with $1/n$ parameters. *arXiv preprint arXiv:2102.08597* (2021).

[180] Chaoning Zhang, Chenshuang Zhang, Sheng Zheng, Yu Qiao, Chenghao Li, Mengchun Zhang, Sumit Kumar Dam, Chu Myaet Thwal, Ye Lin Tun, Le Luang Huy, et al. 2023. A Complete Survey on Generative AI (AIGC): Is ChatGPT from GPT-4 to GPT-5 All You Need? *arXiv preprint arXiv:2303.11717* (2023).

[181] Haojie Zhang, Ge Li, Jia Li, Zhongjin Zhang, Yuqi Zhu, and Zhi Jin. 2022. Fine-Tuning Pre-Trained Language Models Effectively by Optimizing Subnetworks Adaptively. *arXiv preprint arXiv:2211.01642* (2022).

[182] Renrui Zhang, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Yu Qiao. 2023. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199* (2023).

[183] Rongsheng Zhang, Yinhe Zheng, Xiaoxi Mao, and Minlie Huang. 2021. Unsupervised domain adaptation with adapter. *arXiv preprint arXiv:2111.00667* (2021).

[184] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493* (2022).

[185] Hongyu Zhao, Hao Tan, and Hongyuan Mei. 2022. Tiny-Attention Adapter: Contexts Are More Important Than the Number of Parameters. *arXiv preprint arXiv:2211.01979* (2022).

[186] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A Survey of Large Language Models. *arXiv preprint arXiv:2303.18223* (2023).

[187] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. In *The Eleventh International Conference on Learning Representations*.

[188] Jun-Jie Zhu, Jinyue Jiang, Meiqi Yang, and Zhiyong Jason Ren. 2023. ChatGPT and environmental research. *Environmental Science & Technology* (2023).

[189] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593* (2019).