Published: 06 October 2025
Online AM: 03 September 2025
Accepted: 18 May 2025
Revised: 14 January 2025
Received: 12 July 2023

SURVEY

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

CHEN LING, Emory University, Atlanta, GA, United States

XUJIANG ZHAO, NEC Laboratories America, Inc., Princeton, NJ, United States

JIAYING LU, Emory University, Atlanta, GA, United States

CHENGYUAN DENG, Rutgers University–New Brunswick, New Brunswick, NJ, United States

CAN ZHENG, University of Pittsburgh, Pittsburgh, PA, United States

JUNXIANG WANG, NEC Laboratories America, Inc., Princeton, NJ, United States

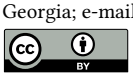
View all

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

CHEN LING, Computer Science, Emory University, Atlanta, United States
XUJIANG ZHAO, NEC Laboratories America Inc, Princeton, United States
JIAYING LU, Computer Science, Emory University, Atlanta, United States
CHENGYUAN DENG, Rutgers The State University of New Jersey, New Brunswick, United States
CAN ZHENG, University of Pittsburgh, Pittsburgh, United States
JUNXIANG WANG, NEC Laboratories America Inc, Princeton, United States
TANMOY CHOWDHURY, George Mason University, Fairfax, United States
YUN LI, George Mason University, Fairfax, United States
HEJIE CUI, Computer Science, Emory University, Atlanta, United States
XUCHAO ZHANG, Microsoft Research, Redmond, United States
TIANJIAO ZHAO, BlackRock Inc, Atlanta, United States
AMIT PANALKAR, BlackRock Inc, Atlanta, United States
DHAGASH MEHTA, Black Rock Inc, Chiyoda, United States
STEFANO PASQUALI, BlackRock Inc, New York, United States
WEI CHENG, NEC Laboratories America Inc, Princeton, United States

---

Xujiang Zhao, Jiaying Lu, Chengyuan Deng, Can Zheng, Junxiang Wang and Tanmoy Chowdhury contributed equally to this research.
Authors' Contact Information: Chen Ling, Computer Science, Emory University, Atlanta, Georgia, United States; e-mail: chen.ling@emory.edu; Xujiang Zhao (corresponding author), NEC Laboratories America Inc, Princeton, New Jersey, United States; e-mail: xuzhao@neclabs.com; Jiaying Lu, Computer Science, Emory University, Atlanta, Georgia, United States; e-mail: jiaying.lu@emory.edu; Chengyuan Deng, Rutgers The State University of New Jersey, New Brunswick, New Jersey, United States; e-mail: cd751@rutgers.edu; Can Zheng, University of Pittsburgh, Pittsburgh, Pennsylvania, United States; e-mail: caz51@pitt.edu; Junxiang Wang, NEC Laboratories America Inc, Princeton, New Jersey, United States; e-mail: junwang@nec-labs.com; Tanmoy Chowdhury, George Mason University, Fairfax, Virginia, United States; e-mail: tchowdh6@gmu.edu; Yun Li, George Mason University, Fairfax, Virginia, United States; e-mail: yli38@gmu.edu; Hejie Cui, Computer Science, Emory University, Atlanta, Georgia, United States; e-mail: hejie.cui@emory.edu; Xuchao Zhang, Microsoft Research, Redmond, Washington, United States; e-mail: xuchaozhang@microsoft.com; Tianjiao Zhao, BlackRock Inc, Atlanta, Georgia, United States; e-mail: tina.zhao@blackrock.com; Amit Panalkar, BlackRock Inc, Atlanta, Georgia, United States; e-mail: Amit.panalkar@blackrock.com; Dhagash Mehta, Black Rock Inc, Chiyoda, United States; e-mail: dhagash.mehta@blackrock.com; Stefano Pasquali, BlackRock Inc, New York, New York, United States; e-mail: pacca.pasquali@gmail.com; Wei Cheng, NEC Laboratories America Inc, Princeton, New Jersey, United States; e-mail: weicheng@nec-labs.com; Haoyu Wang, NEC Laboratories America Inc, Princeton, New Jersey, United States; e-mail: haoyu@nec-labs.com; Yanchi Liu, NEC Laboratories America Inc, Princeton, New Jersey, United States; e-mail: yanchi@neclabs.com; Zhengzhang Chen, NEC Laboratories America Inc, Princeton, New Jersey, United States; e-mail: zchen@nec-labs.com; Haifeng Chen, NEC Laboratories America Inc, Princeton, New Jersey, United States; e-mail: Haifeng@nec-labs.com; Chris White, NEC Laboratories America Inc, Princeton, New Jersey, United States; e-mail: chris.white@nec-labs.com; Quanquan Gu, University of California Los Angeles, Los Angeles, California, United States; e-mail: qgu@ucla.edu; Jian Pei, Duke University, Durham, North Carolina, United States; e-mail: j.pei@duke.edu; Carl Yang, Computer Science, Emory University, Atlanta, Georgia, United States; e-mail: j.carlyang@emory.edu; Liang Zhao (corresponding author), Computer Science, Emory University, Atlanta, Georgia, Georgia; e-mail: liang.zhao@emory.edu.



ACM 0360-0300/2025/10-ART79
https://doi.org/10.1145/3764579

HAOYU WANG, NEC Laboratories America Inc, Princeton, United States
YANCHI LIU, NEC Laboratories America Inc, Princeton, United States
ZHENGZHANG CHEN, NEC Laboratories America Inc, Princeton, United States
HAIFENG CHEN, NEC Laboratories America Inc, Princeton, United States
CHRIS WHITE, NEC Laboratories America Inc, Princeton, United States
QUANQUAN GU, University of California Los Angeles, Los Angeles, United States
JIAN PEI, Duke University, Durham, United States
CARL YANG, Computer Science, Emory University, Atlanta, United States
LIANG ZHAO, Computer Science, Emory University, Atlanta, United States

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 making large language models disruptive in many 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 summarize better 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.

CCS Concepts: • **Computing methodologies** → **Natural language processing**; *Planning and scheduling;*

Additional Key Words and Phrases: Large language models, natural language processing, domain specialization

ACM Reference Format:
Chen Ling, Xujiang Zhao, Jiaying Lu, Chengyuan Deng, Can Zheng, Junxiang Wang, Tanmoy Chowdhury, Yun Li, Hejie Cui, Xuchao Zhang, Tianjiao Zhao, Amit Panalkar, Dhagash Mehta, Stefano Pasquali, Wei Cheng, Haoyu Wang, Yanchi Liu, Zhengzhang Chen, Haifeng Chen, Chris White, Quanquan Gu, Jian Pei, Carl Yang, and Liang Zhao. 2025. Domain Specialization as the Key to Make Large Language Models Disruptive: A Comprehensive Survey. *ACM Comput. Surv.* 58, 3, Article 79 (October 2025), 39 pages. https://doi.org/10.1145/3764579

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 [167], **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 [192], multiple-choice question answering [141], and story generation [14], 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)** [184], such as GPT-3 [12] (175B parameters), PaLM [20] (540B parameters), and LLaMA [163] (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 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, and so on. The acquisition of such capabilities and experience even requires human beings many years of training, a lot of which is 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 LLM solver with no customization. More importantly, the requirement for domain knowledge in professional-level usage also needs to be very in-depth, 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 least, languages are constrained by social norms [126], cultural conformity [160], religious beliefs [127], legal and ethical requirements [97], all of which are changing parameters in different locations, countries, populations, races, communities, and so on, 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. In particular, 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 [172]. 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 [7]. 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 [184]. However, crafting effective demonstrations remains challenging due to ambiguous or incomplete user instructions. While newer LLMs have incorporated techniques like sparse attention to increasing the context window (up to 60K tokens) [189], these advancements primarily optimize for perplexity, not practical usability. Challenges still remain in efficiently utilizing extended context for complex tasks, such as context forgetting and difficulty in retrieving relevant information.

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 overfit to target domains. In addition to the data requirement and complex model architecture, LLMs typically consist of billions of parameters, e.g., LLaMA models [163] and Gemini [161] contain 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, adaptability 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, open problems, and potential future directions. To surmount these obstacles and harness artificial intelligence for more effectively accomplishing tasks across various domains, this survey article comprehensively reviews the current state-of-the-art LLM domain specialization. The major contributions of this article 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. This article 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 [108, 133] of PLMs and their use in diverse NLP tasks exist, they don't necessarily apply to LLMs. Given the recent growth in popularity and effectiveness of LLMs, several review articles have emerged, addressing various LLM aspects. Some focus on fundamental LLM components [93, 192, 206], others on the history and potential applications of generative AI [14, 200], and a few [107] on enhancing LLMs with reasoning capabilities. However, a comprehensive review and technical taxonomy of LLM domain specialization are yet to be provided.

Specializing language models for specific domains. Recent review articles 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, 143], including lack of originality and inaccuracies. Practical considerations for legal domain-specific language models have also been suggested in [149]. In the finance sector, initial steps towards a finance-specialized LLM have shown improved performance on financial tasks without compromising general benchmarks [187]. 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

LLMs are advanced PLMs characterized by their massive scale, typically containing billions of parameters, enabling them to perform a wide range of tasks and generate coherent, context-aware text [108, 133]. Both LLMs and PLMs undergo pretraining on large corpora to learn language patterns, but LLMs differ in their greater capacity, generalization ability, and effectiveness in handling complex, multi-step 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 LLMs for specific domains.

2.1 Background

In PLMs, a type of neural network pre-trained on large text corpora, the *input* is a text sequence with context, often accompanied by a task-specific *prompt* that clarifies the objective. For example, in text summarization, a prompt like "Summarize the key points in the following passage:" would precede the input passage. The *output* is the generated text sequence or prediction, which may require post-processing like token decoding or label extraction depending on the specific NLP task. As LLMs are typically scaled-up versions of PLMs, they follow a similar architecture design to PLMs, which

come in three main flavors: *encoder-only*, *encoder-decoder*, and *decoder-only* architectures. This brief introduction will provide an overview of these architectures and summarize their characteristics.

— *Encoder-only Language Models* transform input text into vector representations, capturing patterns and semantics without explicit decoding. Models like BERT [30] excel at tasks such as classification and sentiment analysis through masked language modeling.
— *Encoder-Decoder Language Models* combine encoding of input text with decoding for output generation, using cross-entropy loss to optimize sequence-to-sequence tasks. T5 [135] exemplifies this architecture for tasks like translation and summarization.
— *Decoder-only Language Models*, such as GPT [134], generate text autoregressively by predicting each subsequent token based on previous context, making them ideal for text generation tasks.

2.2 Taxonomy of Domain Specialization Techniques

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*. As shown in Figure 1, 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 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), ranging from the parameter setting to the model architecture.

Beyond LLM accessibility-based taxonomy, domain specialization methods can be categorized by training strategies (fine-tuning with domain-specific data, training from scratch, or mixed approaches), or intervention levels (pre-training, fine-tuning, or inference-time interventions). Each approach offers unique methods for enhancing LLMs' domain-specific capabilities. 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 and 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)). The relation between approaches in different categories is summarized as follows:

Different levels of specialization. Each approach operates at a different level of specialization (i.e., black, grey, and white boxes). 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 (Section 3). On the other hand, fine-tuning modifies the LLM's internal parameters, leading to more profound changes in the model's behavior (Section 5).

Tradeoffs. The approaches exhibit different tradeoffs regarding computational cost, ease of implementation, and generalization. Augmenting with external information (Section 3) and crafting

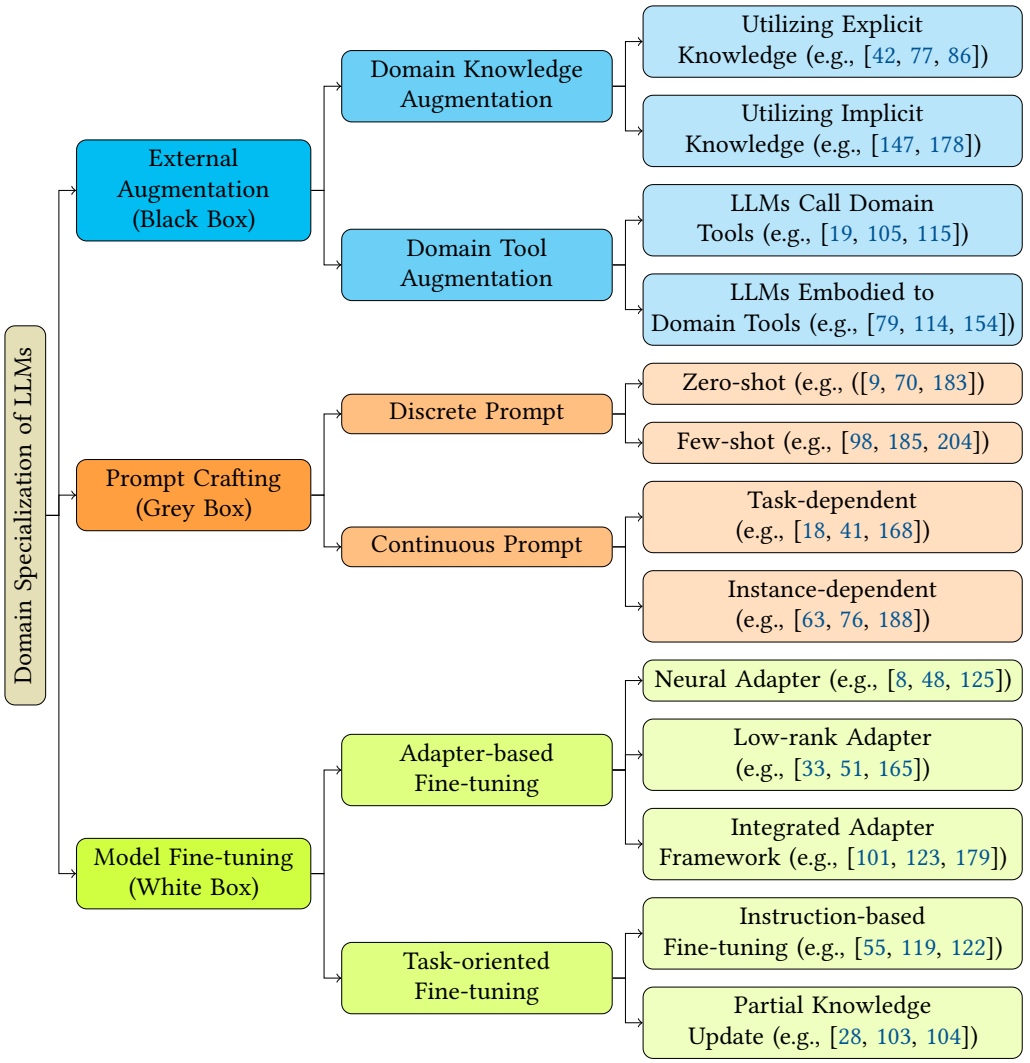


Fig. 1. The taxonomy for current techniques on LLM domain specialization.

task-specific instructions (Section 4) are often less computationally expensive than knowledge updates but may not yield the same level of performance improvement [107]. Fine-tuning provides more performance gains but can be more challenging to implement and may suffer from reduced generalization capabilities if overfitting occurs [206] (Section 5). While white-box approaches are often regarded as superior for domains requiring high precision or profound domain adaptation, there is evidence that black or grey-box methods can achieve comparable performance in certain cases. For example, tasks that require lightweight, modular adaptations—such as integrating real-time external knowledge for rapidly changing domains like news summarization—may benefit more from black-box methods due to their flexibility and lower computational demands [93]. Similarly, grey-box approaches can be particularly effective in structured domains, such as legal document analysis, where prompt crafting can exploit domain-specific patterns without retraining the entire model [164].

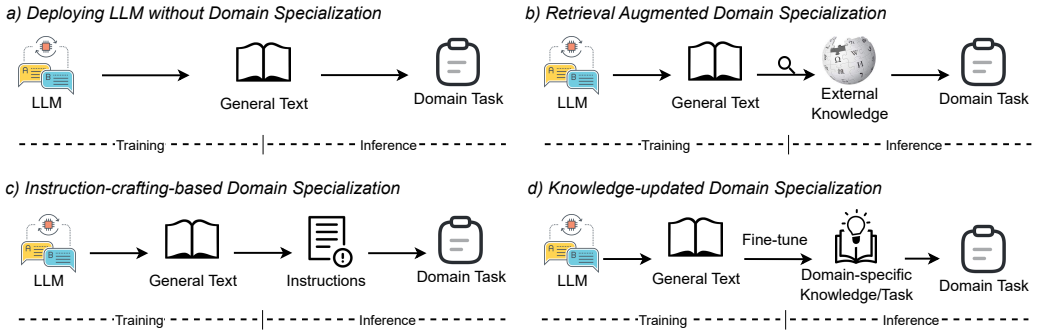


Fig. 2. Different approaches for tailoring LLMs to domain-specific tasks: (a) using a general LLM without modifications, (b) enhancing the LLM's performance through retrieving relevant external knowledge, (c) utilizing domain-specific and task-relevant instructions to improve LLM's capabilities, and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

Complementary nature: 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 [53]. Similarly, designed prompts can be used alongside neural adapters to guide the model's output while taking advantage of the newly learned domain-specific knowledge [75, 91].

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 article, 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**: The initial step involves clearly defining the domain, objectives, and constraints. Whether fine-tuning a model (white box, Section 5), crafting prompts (grey box, Section 4), or augmenting inputs/outputs (black box, Section 3), a clear understanding of the domain is crucial for identifying relevant data, knowledge, and resources.
(2) **Augmentation**: This stage incorporates domain-specific knowledge into the model or its inputs/outputs through various approaches: fine-tuning with domain-specific data (white box), using gradients to craft prompts (grey box), or modifying inputs/outputs with external tools (black box) [107].
(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 fulfill the domain objectives best. This can be done through methods like gradient descent for white box approaches [87], prompt engineering for grey box approaches [88], or post-processing and filtering of outputs for black box approaches [208].
(4) **Evaluation**: The final stage tests the specialized model's performance against predefined benchmarks, gathering feedback from domain experts or domain-specific test sets to refine the model.

3 External Augmentation for Domain Specialization

Retrieval augmentation aims to enhance LLMs by retrieving relevant information from external sources to augment the response. 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. Knowledge Augmentation supplements the model's responses with external information, while Tool Augmentation expands the model's capabilities for tasks it couldn't perform otherwise. Domain knowledge enhances depth and accuracy within a specific field, while domain tools enable the model to perform tasks beyond its inherent capabilities. Both approaches can augment LLMs on domain-specific tasks with up-to-date knowledge, solving Challenge 1 without frequent model retraining.

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 an external context, which is also known as **Retrieval-augmented Generation** (**RAG**). When presented with an explicit knowledge source containing domain-specific information, LLMs 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 without frequent fine-tuning.

Current techniques often employ a neural retriever to acquire task-relevant information from either a large corpus (e.g., Wikipedia [26, 77, 89, 153]) or a knowledge base (e.g., Wikidata) [11, 42, 78, 95]. Specifically, given a task-specific query, early works [11, 77, 89, 153] 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, typically by appending the retrieved content directly to the query' input representation before feeding it into a model. This concatenation can occur at the token level, where the retrieved text is treated as additional context [26], or in a latent space, where vectorized representations of the query and retrieved information are combined [77]. The enriched input allows the model to leverage external knowledge more effectively during task execution. Common downstream tasks can benefit from this approach. For instance, the model can reference retrieved factual data in question answering tasks to provide more accurate and context-aware answers [153]. Similarly, in fact verification tasks, incorporating external content ensures that responses are comprehensive and aligned with domain-specific details [121].

With the prevalence of LLMs, researchers have been using LLMs to replace the neural network-based retriever [57, 136], 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 PaLM [20] on question answering datasets. Furthermore, to enhance the transparency and explainability of the retrieval, He et al. [42] proposed to leverage LLMs to decompose the information retrieval process with detailed reasoning steps, and Lu et al. [95] explored utilizing

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 works [36, 39, 106, 170] leverage attention mechanisms to enable PLMs to retrieve task-related information from this implicit knowledge. These studies transform task-specific queries into embeddings, calculating attention between the query vector and each knowledge entry. A softmax function generates 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, allowing the model to access relevant information during inference.

While LLMs store substantial information in their parameters for generating high-quality responses, augmenting them with implicit knowledge isn't always necessary. Implicit knowledge requires additional steps, such as transforming domain-specific data into latent vectors, which can reduce practicality. However, researchers are exploring its potential to address complex scenarios. This process often involves transfer learning, where pre-trained models are adapted to perform specialized tasks by integrating domain-specific knowledge. For instance, implicit knowledge has been utilized to store instructional sequences in solving long-form mathematical problems or multi-step logical reasoning tasks [147, 178] that contain "large inputs".

In this context, "large inputs" refer to sequences or datasets that exceed standard LLM context lengths, such as detailed financial reports or lengthy legal documents. By employing an instruction cycle, implicit knowledge enables LLMs to retrieve relevant parts of such inputs dynamically [67]. For example, implicit knowledge can store case precedents as embeddings in legal analysis, allowing the LLM to retrieve and reference specific cases when analyzing new legal queries [198]. Similarly, implicit knowledge embeddings in financial forecasting can help the LLM process historical market data and extract relevant patterns to predict future trends [116]. This augmentation facilitates handling domain-specific problems requiring extensive or complex input processing beyond the typical token limit. Moreover, they also explored ways to integrate retrieved information beyond the input layer, incorporating it into intermediate and output layers of language models[35]. Specifically, this involves techniques such as fusing external knowledge into hidden states during forward propagation [139] or dynamically updating model representations through attention mechanisms [196]. These approaches aim to refine the model's understanding and improve its performance on knowledge-intensive tasks by leveraging external information at multiple stages of processing, rather than relying solely on static inputs.

3.1.3 Approach Efficiency and Open Challenges.

Computational efficiency and resource requirements. Broadly, the following factors influence the resource requirements: Domain knowledge augmentation involves varying computational demands depending on the type of knowledge integrated. For *explicit knowledge*, such as structured knowledge graphs or ontologies, the computational overhead arises primarily during preprocessing and integration, which may require moderate to high resources depending on the scale of the knowledge base and the complexity of the methods used. These approaches benefit from the structured nature of the data but often require large, curated datasets to be effective. *Implicit knowledge integration*, on the other hand, typically involves fine-tuning or domain-adaptive pretraining using unstructured datasets. This process demands substantial computational power, particularly for large-scale LLMs,

as it involves training or adapting models on extensive domain-specific corpora. While these techniques enable the model to internalize nuanced, domain-specific information, they are resource-intensive and time-consuming.

By incorporating external knowledge, LLMs function like librarians, retrieving relevant information rather than memorizing every resource. This approach enhances performance in specialized tasks without extensive retraining, enabling more adaptable AI systems that can update their knowledge dynamically. Unlike traditional memorization, this process involves the retrieval of domain-specific or up-to-date information from structured databases or knowledge sources [155], which can then be integrated at various stages beyond the input layer. For instance, retrieved knowledge can be explicitly embedded within intermediate layers of the model or used to refine output probabilities, enabling deeper interactions between the retrieved content and the model's internal reasoning process. However, this method raises significant challenges, such as ensuring the accuracy, relevance, and seamless integration of retrieved information—topics central to fields like digital libraries and information science [72]. Shared tasks in information retrieval, such as CLEF[1] and TREC,[2] provide valuable benchmarks for evaluating and improving these capabilities, offering insights into how retrieval-based approaches can achieve robust performance in real-world scenarios.

(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, the LLM needs to have the option of accepting or rejecting retrieved information [5], 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 (NCBI Web APIs for genomics QA [64], automated formal theorem prover for mathematical proofs [61], sandbox environment for social behavior simulation [120], etc.). These tools are designed to handle domain-specific tasks, data, or knowledge effectively, and they 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 also exhibit 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 [107, 145] include (1) unstable result formats depending on the random seeds, generation hyperparameters, and input contents [146]; (2) inability to access up-to-date information [115] 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 approach to overcome the limitations of solely using either domain tools or LLMs for domain-specific tasks. This approach combines the strengths of both, utilizing domain-specific knowledge, algorithms, and functionalities from the tools, while offering

[1] https://www.clef-initiative.eu/
[2] https://trec.nist.gov/

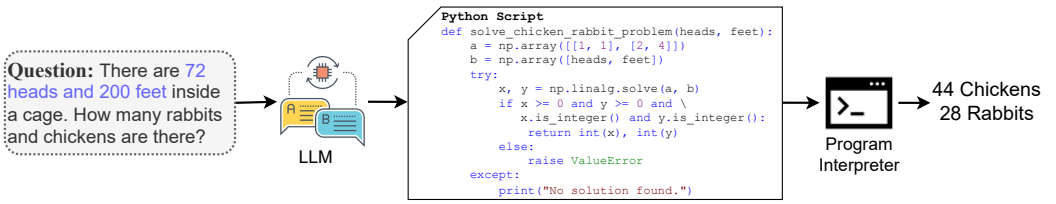


Fig. 3. A toy example for LLMs call domain tools.

a user-friendly interface through LLMs. This collaboration optimizes the use of domain 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 $c = f_\Theta(p)$. (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-LLM 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-shot or few-shot prompting for executable command generation (please refer to Section 4 for more details). According to Figure 3, where the task is 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 tools, LLMs can generate commands that adhere to their syntax and format requirements. Early explorations focused on eliciting search engine queries [71, 105, 115] and database queries [19, 81, 129]. Later work expanded to writing executable codes for interpreters like Python [17, 37, 159], Wolfram [186], and domain-specialized APIs like the chatGPT plugin system [117], typically using zero-shot or few-shot prompting techniques.

Some tasks involve multiple types of tools to accomplish. Researchers have started to generalize LLMs as **task planners** (also known as "API selectors" or "controllers") that call multiple tools. These approaches focus on decomposing complex tasks into concrete subtasks and coordinating between multiple tools. Examples include DSP [61]'s Draft, Sketch, and Prove framework for automated theorem proofs, TaskMatrix.AI [84]'s approach to deriving solution outlines and matching subtasks to domain models, HuggingGPT [150]'s LLM-controlled domain model management, and Qin et al. [132]'s tool-augmented framework for task decomposition and dynamic execution planning.

LLMs Embodied to Domain Tools. LLMs can also be called 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 [154] investigates LLMs' ability to assist robots, 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. [114] 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, 83, 181]. Furthermore, researchers are exploring how multiple LLMs can interact with their environment or collaborate to solve complex tasks. Mind's Eye [90] uses simulated physics engines to align LLM reasoning with physical tasks. CAMEL [79] enables LLM agents to collaborate through role-based instruction-following. Similarly, [120] employs 25 LLMs in a sandbox environment to simulate human-like behavior. These efforts point toward a community of specialized LLMs, where expert models collaborate, combining strengths to tackle real-world challenges beyond the scope of a single model.

3.2.1 Approach Efficiency and Open Challenges.

Computational efficiency and resource requirements. Generally, these approaches leverage pre-existing domain-specific tools, which may reduce the computational burden on the LLM itself by offloading specialized tasks to external systems. However, the integration process can introduce overhead, particularly if the LLM must format inputs and interpret outputs according to the strict requirements of the domain tool. For instance, the need to preprocess data or generate API-compatible queries can increase computational complexity. Additionally, domain tools often have their own resource demands, including memory, processing power, and sometimes specialized hardware or licenses. In terms of data requirements, these techniques typically rely on curated domain-specific datasets for fine-tuning or aligning the interaction between the LLM and the domain tool. While the amount of data needed can vary widely, the quality and specificity of the data are more critical than sheer volume. Thus, while domain tool augmentation is resource-intensive in its setup, it can achieve significant efficiency gains during task execution by leveraging specialized systems.

By harnessing the capabilities of LLMs, domain tools can facilitate a wide range of tasks across diverse fields, such as robotics, virtual agents, and real-world problem-solving. These tools enhance human-machine collaboration by enabling more intuitive and efficient interactions, ultimately improving adaptability and effectiveness in addressing complex challenges. However, augmenting LLMs with domain tools still presents several open obstacles:

(1) *Automated integration*: Integrating LLMs with domain tools requires significant effort to ensure compatibility and effectiveness. A promising direction is to use LLMs as a unified interface, leveraging standardized protocols to seamlessly connect diverse applications and services, thus enabling streamlined communication.
(2) *Eliminating dependency on domain tools*: An alternative future direction focuses on advancing LLMs toward **artificial general intelligence** (**AGI**) capable of handling tasks without relying on external tools or domain-specific knowledge. Such an AGI model could transform the utility of LLMs, enabling them to autonomously perform complex and sophisticated tasks with higher efficiency and minimal external intervention.

4 Prompt Crafting for Domain Specialization

While general LLMs are powerful, to address Challenge 2, further model tuning on prompts can enhance their ability to adhere to user intentions and generate accurate and less toxic responses [119, 177]. 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 [12, 118] allow LLMs to quickly adapt to unseen domains by discrete prompting, and GPT-3 [12] 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.

Given an LLM $f_\Theta(\cdot)$ where Θ denotes pre-trained model parameters, the task is to elicit desired output y from LLM with a discrete prompt p and a test query, denoted as $y = f_\Theta([p; c])$, when freezing Θ. It is worth noting that both y, p and 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.

Depending on the prompting types, discrete prompts can be divided into two categories: (1) zero-shot [9, 70, 144], where the prompt p consist of only the task description; and (2) few-shot [98, 109, 185], where the prompt p consists of the task description and few illustrative examples. The key difference lies in whether task demonstrations 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 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 [146], post-processing 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 [12, 144, 182], where different tasks can be represented in a unified sequence generation format. PADA [9] 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 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 [34], and rumor detection [85]. Later, Kojima et al. [70] extend the few-shot-Chain-of-Thoughts (Few-shot-CoT) [185] 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.

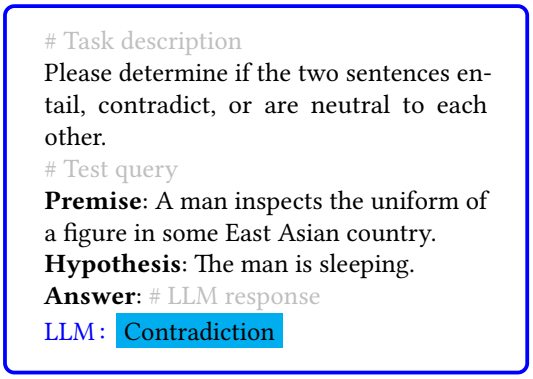


Fig. 4. An example (adapted from [73]) of zero-shot discrete prompts, where task description, and/or a test query are provided to LLMs. No illustrative examples are provided in zero-shot prompts.

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: A soccer game with multiple males playing.
Hypothesis: Some men are playing a sport.
Answer: Entailment
Example 2
Premise: Chest x-ray showed mild congestive heart failure.
Hypothesis: The patient complains of cough.
Answer: Neutral
Test query
Premise: A man inspects the uniform of a figure in some East Asian country.
Hypothesis: The man is sleeping.
Answer: # LLM response
LLM : Contradiction

Fig. 5. An example (adapted from [73]) 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 an example of how few-shot discrete prompts work. Different from zero-shot prompts, few-shot prompts consist of a few input and desired output pairs to convey the task intention and provide illustrations of the desired output formats. Few-shot prompts yield more stable output formats and more decent performance on downstream tasks [12, 118]. **Chain-of-Thought (CoT)** [185] enhances the domain specialization ability of LLMs by introducing a series of intermediate reasoning steps for complex reasoning tasks, such as multi-step mathematical problem solving, causal inference, or logical deduction. However, CoT also brings extra cost in manually designing reasoning chains for each test example. As a follow-up, Auto-CoT [204] 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 [98] studies the programming language format instructions to tackle structured reasoning tasks.

More advanced techniques are then proposed to further improve the discrete instruction of LLMs for domain specialization and customization. For instance, *ensemble-based instruction* [96, 175, 176] 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* [3, 31, 32, 68, 191, 207] to break down a complex, unseen task into a series of easier subtasks and then employs LLMs with specific instructions for each subtask.

4.1.3 Approach Efficiency and Open Challenges.

Computational efficiency and resource requirements. Discrete prompting techniques, particularly zero-shot and few-shot approaches, are computationally efficient compared to other domain specialization methods because they do not require updating the pre-trained LLM parameters. Instead, they rely on carefully crafted natural language prompts to guide the model's reasoning abilities. This eliminates the need for extensive fine-tuning or retraining. The computational cost is largely determined by the model's size and the context window required to accommodate the prompt and test query. Zero-shot prompting, involving only task descriptions, typically demands fewer tokens and is more lightweight than few-shot prompting, which includes illustrative examples and therefore requires more extensive use of the model's context window. Overall, discrete prompting is highly efficient in terms of both computation and data requirements, as it leverages pre-trained

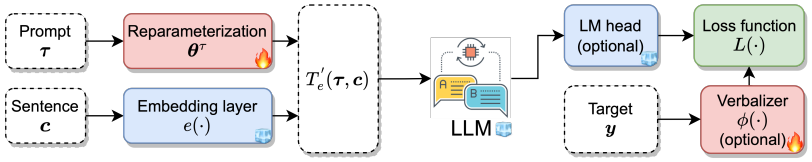


Fig. 6. An illustration of soft prompt tuning. The fire icon represents tunable modules and the ice icon represents that the parameters of those modules are frozen during tuning. Verbalizers are only used for classification tasks where a mapping from class label to label words is required, which can be one-one mapping, trainable tokens [41], or enhanced with extra knowledge [52].

knowledge without additional training datasets. However, performance may depend on the quality of the prompt design, requiring iterative experimentation rather than computational overhead.

However, crafting discrete prompts for LLMs in domain specialization poses several open challenges:

(1) *Effectiveness*: The discrete instructions are often curated by experts for domain-specific tasks, raising concerns about whether these instructions are the most effective or introduce unintended biases. Biases can emerge from the perspectives or preferences of the experts involved, potentially limiting the generalizability of the prompts. Therefore, there is a need for robust (automatic) evaluation methods to assess the effectiveness and neutrality of these instructions. This can be achieved through collaboration between domain experts and data scientists, who can iteratively analyze performance and refine instructions.
(2) *Scalability and adaptability*: Developing automated methods to generate, select, or combine discrete instructions without excessive human intervention is critical for improving scalability. However, such approaches must also address the challenge of data scarcity in certain domains, as limited access to high-quality domain-specific examples may hinder the adaptability of LLMs to niche or less-studied areas. Techniques for augmenting or synthesizing representative domain data could play a vital role in addressing this issue.

4.2 Continuous Prompt

The continuous prompt is a sequence of tokens proposed to attach with the input sentence and guide LLMs with extra and **learnable** knowledge from datasets by continuous *prompt tuning*. In this case, the continuous prompt serves as a parameterized vector/matrix instead of the natural language instruction. Prompt tuning aims 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.

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 ith 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 $\theta^{\mathcal{T}}$ using the downstream loss function $\mathcal{L}$ as follows:

$$\theta^{\mathcal{T}\star} = \underset{\theta^{\mathcal{T}}}{arg\,max}\ \mathcal{L}(f_{\Theta}(T_e^{'}(\tau, c)), y), \tag{1}$$

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:

$$\theta^{\mathcal{T}\star} = \underset{\theta^{\mathcal{T}}}{arg\,max} \sum_{c} \log P\Big([MASK] = \phi(y)|T_e^{'}(\tau, c)\Big). \tag{2}$$

The continuous prompt techniques fall 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 [75] 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 can only update the parameters in a limited vicinity, leading to the possibility of falling into a local minimum [2]. Therefore, a more effective initialization approach would involve using embeddings of concrete task-specific words.

One of the pioneering works, WARP [41] initializes the prompt by the embedding of a 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 models, while Prompt-tuning [75] 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 shown prompt transfer capabilities across domains, with methods like SPoT [168] and PPT [40] initializing prompts using source tasks or self-supervised learning. Su et al. [156] demonstrated prompt transferability across tasks and models, revealing that well-initialized prompts can accelerate training convergence. LFPT5 [131] leveraged this transferability for lifelong learning, using soft prompts to simultaneously solve current tasks and generate samples from previous tasks to mitigate *catastrophic forgetting*. Progressive prompts [137] further extended this approach by concatenating prompts optimized on previous tasks with 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 into the original input sentences without extra language phrases. The pioneering study, WARP [41] adopted all three intersections with a "[MASK]" token for classification tasks. In Prefix-tuning [82], 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 [75] prepend prompts at the input layer, achieving comparable results to fine-tuned models.

Templates are widely used to improve adaptation performance [146], such as reformulating NLP tasks like sentence classification into masked word prediction. For instance, KnowPrompt [18] designed a template appending a "[MASK]" between subject and object for relation extraction, incorporating trainable "virtual type words" to align output embeddings logically with target relations. Similarly, KiPT [80] developed a knowledge extractor for event detection that identifies trigger words based on semantic similarity, prepending these words and event labels to soft prompts and reformulating sequence tagging tasks into generative event record outputs.

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 focuses on learning a joint and adaptive representation of tasks as well as instance context. IDPG [188] proposed an additional two-layer perceptron as a prompt generator, down and up project the sentence embedding to the adaptive soft prompt. ATTEMPT [4] first trains multiple prompts on large-scale source tasks, and calculates an aggregated prompt based 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 to 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., [10] 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., [76] learn the joint representation of prompt and input by a frozen T5 encoder following cross- and self-attention layers. Liu et al., [92] 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 takes the embedding of [CLS] token or pooling of embeddings of sentence tokens.

Prompt Construction Enhancement. Similar to the construction enhancement, instance-dependent prompt tuning introduces instance-dependent knowledge as concrete words or learns adaptive prompts in terms of positioning and length. OntoPrompt [197] enriches the template with instance-related knowledge from external ontology as an additional text, and tunes 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 [194] 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 the post-fix prompt, given most prior works prepend the prompt to the input sentence.

4.2.3 *Approach Efficiency and Open Challenges.*

Computational efficiency and resource requirements. Continuous prompting is computationally efficient compared to full model fine-tuning because it updates only a small fraction (approximately 0.01%) of the LLM's parameters [87], specifically those associated with the prompt. This significantly reduces memory and computational overhead during the tuning process, as the LLM remains frozen. The primary computational cost arises from optimizing the trainable prompt embeddings using gradient-based methods during fine-tuning. This cost is relatively low compared to approaches requiring updates to the entire model's parameter set. The data requirements for effective prompt tuning depend on the task but are generally smaller than those for traditional fine-tuning, as the pre-trained LLM provides a strong foundation. However, achieving optimal performance may require iterative experiments with template construction and verbalizer mappings, which can increase the time spent in the development phase rather than computation.

Continuous prompt tuning presents an efficient way to handle domain-specific tasks. However, it has its limitations:

(1) *Interpretability* remains a key challenge, as continuous prompts often lack meaningful or human-interpretable content when mapped to token vectors [18, 41]. While methods like KnowPrompt [18] reveal domain-specific patterns, their coherence as sequences remain unclear.
(2) *Limited access* to LLMs, especially API-restricted models, constrains gradient-based optimization for continuous prompts. Derivative-free methods like **Black-box tuning (BBT)** [157] and RLPrompt [29] offer promising alternatives, but data scarcity and risks of introducing biases persist, particularly for discrete prompt search. Further advancements are needed to address these challenges effectively.

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 illustrated 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

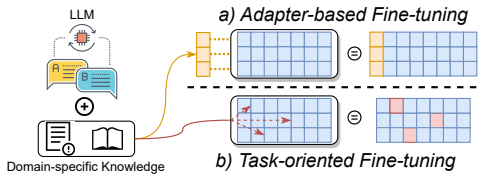


Fig. 7. The visualization of two approaches to fine-tune LLMs based on domain-specific knowledge, where the blue rectangle denotes the set of parameters in LLM. (a) the adapter-based LLM fine-tuning aims to fine-tune LLMs on specific domains with a small number of extra parameters (i.e., adapter); and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

subset of these parameters. To solve Challenge 3, both types of 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 [48, 138] under the umbrella of parameter-efficient fine-tuning.

5.1.1 Adapters. Adapters are trainable modules inserted between layers of a pre-trained model [48]. The key property of adapters highlights that the parameters of the original language model keep frozen, thus providing 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:

$$\phi(f_\Theta(X)) \approx \phi(f_\Theta \circ g_{\Delta\Theta}(X)), \qquad \phi_D(f_\Theta(D)) \leq \phi_D(f_\Theta \circ g_{\Delta\Theta}(D)). \tag{3}$$

The first equation ensures general task performance is preserved while the second improves domain-specific tasks. This balance is crucial for models requiring both versatility and specialization. In scenarios where task performance on general data is less critical, such as when focusing solely on domain-specific tasks, techniques like child models and knowledge distillation can be applied. These approaches further optimize performance by distilling knowledge from the general model while retaining essential domain-specific knowledge.

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 [203] 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 [100]. This was achieved using the architecture defined in AdapterFusion [123]. 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 PLMs like GPT-2 [22, 100, 203], 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 [202].

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 classified based on their architectures into neural and low-rank adapters. To facilitate user-friendly implementation, a growing body of work is dedicated to building comprehensive frameworks of different adapters [54, 124]. Certain studies have also shown that adapter integration can yield superior performance across various downstream tasks.

Neural adapters. We call adapters with neural network architectures neural adapters. In their original design, [48] uses a composition of down-projection, GeLU non-linearity [45] and up-projection with the feed-forward layers as the backbone. Later [8] 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 article when referring to [48]. Neural adapters draw inspiration from neural network architecture designs. Adapters in [123] feature residual connections, while [125]'s MAD-X framework introduces invertible adapters mimicking autoencoders. Zhao et al.[205] explored tiny-attention adapters, and most architectures use fully connected layers for projections. Compacters[66] innovatively use parameterized hypercomplex multiplication layers for parameter efficiency, while SparseAdapter [44] applies network pruning techniques. Adaptation approaches include inserting adapters *inside* language models (like standard adapters) and *outside* models, such as *K*-adapters [173] which train domain-specific adapters. Sung et al. [158] addressed training memory concerns by proposing ladder side-tuning, which adds small modules connected via shortcuts to the language model backbone.

Low-rank adapters. Low-rank adaptation (LoRA) [51] is inspired by the observation that LLMs reside on an intrinsic subspace [1], 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 [165], 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**) [33]. 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 [43, 54, 101, 179] as an important building block. More details on these frameworks follow below.

Integrated adapter framework. With the flourishing results of effective adapters, as introduced above, it is a natural extension to incorporate several adapters from various families to boost their performance. AdapterFusion [123] employs a straightforward idea: train multiple adapters on different tasks and combine the learned embeddings from each adapter with a fusion layer. UniPELT [101] 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 [48], LoRA [51], Prefix-tuning [82] and Bitfit [199]. Orthogonal to UniPELT, AdaMix [179] 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 concept of learning a routing function on adapter inventories inspired various works: Polytropon [128] jointly learns adapters and routing functions for multi-task learning, with variants studied by [13] including weight averaging and multi-head routing. AdapterHub [124] provides comprehensive adapter integration but lacks LLM support. LLM-adapters [54] addresses this by introducing a framework for open-access LLMs, incorporating four basic adapters (Serial [48], MAD-X [125], Parallel [195], and LoRA [51]) while remaining extensible.

5.1.2 *Approach Efficiency and Open Challenges.*

Computational efficiency and resource requirements. Adapter-based fine-tuning is a computationally efficient and resource-friendly approach to domain specialization. By adding a small number of trainable parameters through lightweight modules, such as adapters, this method avoids updating the original LLM's parameters, significantly reducing the computational overhead compared to full model fine-tuning. The added modules are designed to be compact, typically comprising a fraction of the total model size, making them resource-efficient in terms of both memory and storage. The training process for adapters is also resource-efficient, as it focuses only on the additional parameters rather than the entire model. This enables rapid adaptation to specific tasks or domains with minimal computational overhead. Moreover, since the base model remains frozen, parameter sharing is maintained across domains, allowing for scalable, sequential training of multiple tasks without requiring retraining from scratch.

Adapter-based methods have several drawbacks. Their performance can be sensitive to architectural design and size, risking insufficient representational power or overfitting. Additional modules enlarge the model size, increasing resource demands and potentially extending inference time. Moreover, as Sung et al. note, training memory remains substantial as backpropagation involves the entire model even when previous parameters are frozen [158]. These limitations are further compounded by broader challenges inherent to LLM domain specialization techniques. Potential biases introduced during the adaptation process—such as overemphasis on specific patterns in the fine-tuning data—can inadvertently propagate or exacerbate pre-existing model biases. Data scarcity in niche domains may also hinder the generalizability of adapter-based methods, as limited or imbalanced datasets can restrict the ability to capture diverse and nuanced domain-specific knowledge. Given these challenges, we outline two key ones:

(1) *Stability and universality:* The performance of adapters can vary with different architectures or hyper-parameters, raising questions about their stability and universality across LLMs and task settings. Addressing these issues requires ensuring robustness against potential biases introduced during domain adaptation and extending applicability to underrepresented domains with limited training data.
(2) *Computational resources:* While adapters have shown promise with large-scale LLMs, it remains unclear if they are sufficiently scalable. The potential need for more adapter modules could bring computational cost concerns, especially in resource-limited environments. This underscores the need for novel architectural designs, efficient training algorithms, and strategies to mitigate data scarcity, such as synthetic data generation or active learning.

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 reasons including but not limited to overfitting, catastrophic forgetting, and task-specific biases [180]. (2) Fine-tuning

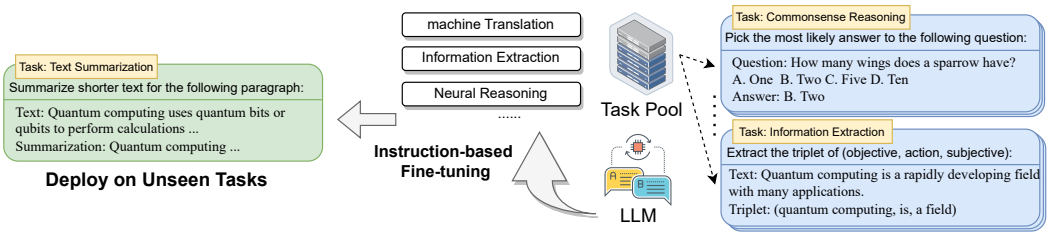


Fig. 8. The overview of fine-tuning an LLM with explicit instructions across various domains and datasets. Particularly, the LLM is fine-tuned on a collection of tasks (commonsense reasoning, information extraction, etc.) with detailed instructions, and the fine-tuned LLM is expected to obtain problem-solving skills.

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 it on a diverse set of tasks with explicit instructions or prompts, which aligns conceptually with Instruct Learning introduced in [119]. As illustrated in Figure 8, the LLM is fine-tuned on a wide range of NLP tasks (e.g., question answering, sentiment analysis, and others) and then evaluated on held-out and previously unseen tasks. Although the training set covers most major NLP tasks, some niche or highly domain-specific tasks (e.g., product question answering in online shopping scenarios) may not be included. Wei et al. [182] 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, 105], 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 CoT 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 [122, 171].

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 [148, 151].

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

$$\tilde{\Theta} = \Theta + \nabla f_\Theta(D) \odot T, \qquad T^{(i)} = \begin{cases} 1, & \text{if } \theta^{(i)} \in \Theta_T \\ 0, & \text{if } \theta^{(i)} \notin \Theta_T \end{cases}, \tag{4}$$

where T denotes the mask vector and $T^{(i)} \in T$ denotes the ith 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 PLMs [49, 140, 210], $|\Theta| = |\Theta_T|$. 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, 46, 103, 104]. 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 [110, 111]. 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, 103, 104]. 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 [46].

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 (4)). 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, 199] 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 [190] utilizes the downstream task data to detect the most task-related parameters as the child network and freezes the parameters in the non-child network to their pre-trained weights. Moreover, Zhang et al. [201] 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 [113]. Similarly, [6, 102, 169] 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, [50, 152] distill the step-by-step CoT reasoning abilities of larger models into smaller models.

5.2.3 *Approach Efficiency and Open Challenges.*

Computational efficiency and resource requirements. Task-oriented fine-tuning involves updating the internal parameters of LLMs to enhance their performance on specific domain tasks. This process is computationally intensive due to the size of modern LLMs and the complexity of their architectures. The resource requirements include substantial GPU or TPU memory, prolonged training time, and access to high-quality domain-specific datasets. Fine-tuning also demands significant energy consumption, making it less efficient compared to prompting-based methods. Additionally, the risk of overfitting, catastrophic forgetting, and task-specific biases can necessitate careful optimization and regularization strategies, further increasing the resource overhead. To mitigate these challenges, approaches like instruction-based fine-tuning and partial parameter updates aim to strike a balance by focusing on efficiency and preserving the model's general capabilities, reducing the overall computational burden compared to full fine-tuning of all parameters.

We have observed that different applications or users may have unique requirements or preferences. However, fine-tuning large-scale LLMs also poses several open challenges:

(1) *Compliance with regulations*: 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. However, ensuring compliance may introduce unintended biases, particularly if regulatory constraints conflict with the generality or fairness of the model.
(2) *Computational resources*: Fine-tuning LLMs necessitates access to high-performance or specialized hardware, which can be expensive to obtain, particularly for individual researchers or smaller organizations. Pursuing fine-tuning efficiency is still a practical yet essential

problem. Additionally, data scarcity in niche domains can hinder fine-tuning efforts, leading to suboptimal performance or overfitting to limited examples.

(3) *Bias and fairness*: The domain specialization process may inadvertently amplify or introduce biases present in the training data, resulting in outputs that are unfair or unreliable. Addressing these biases requires careful data curation and rigorous evaluation, which are resource-intensive and may not completely eliminate the issue.

6 Applications of LLM Domain Specialization

In this survey article, we explore the applications of LLMs across a range of domain-specific tasks in social sciences (e.g., education, finance, and law), natural sciences (e.g., biomedicine and earth science), and formal sciences (e.g., **human-computer interaction** (**HCI**), 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 updates. 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, e.g., predicting financial trends or suggesting personalized medical treatments.
— *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 [99, 142] to clinical healthcare support [58]. At the fundamental biomedicine science level, LLMs can be trained on vast domain-specializing data (e.g., genomic and proteomic) to analyze and predict biological functions, disease mechanisms, and drug discovery [162]. LLMs can also aid in predicting protein structures and interactions, which are critical for understanding cellular processes and designing new drugs [209]. At the clinical healthcare support level, pre-trained or medical corpus fine-tuned LLMs can be used for the NLP of medical records to identify patterns, make diagnoses, and provide personalized treatment recommendations [112]. 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 [174]. Overall, LLMs offer tremendous potential for advancing biology research and improving healthcare outcomes.

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 [74, 94, 187, 193], 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, 130, 166] demands understanding and generation of intricate legal language, comprehension of laws, legal codes, and court rulings, while maintaining absolute precision and an extra formal 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.

HCI and Software Engineering. HCI and software engineering also require 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 [69]. 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 [47]. 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.

7 Open Challenges and Future Works

7.1 Open Challenges

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*: Domain complexities 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 tradeoff 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 LLM specialization, we 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 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 from normal 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 human needs.

— *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.

8 Conclusion

In conclusion, the rapid advancement of LLMs has generated substantial interest in their application to domain-specific tasks across diverse fields. However, challenges such as limited domain expertise, knowledge elicitation, and model complexity often limit their effectiveness. This survey systematically categorizes existing domain specialization techniques based on access levels to LLMs, providing a comprehensive overview of application domains that can benefit from these techniques. By analyzing the advantages, limitations, and interrelationships of these methods, the survey aims to guide domain experts in selecting suitable approaches and inform data scientists about the practical significance and challenges within specific domains. While this work offers a detailed synthesis of the current landscape, future research should focus on the quantitative evaluation of these techniques, benchmarking their performance across metrics such as accuracy and computational efficiency. Incorporating such analyses will provide deeper insights and practical guidance for users. This article serves as a foundational resource for researchers and practitioners, fostering innovation and interdisciplinary collaboration as the field of LLM domain specialization continues to evolve.

References

[1] Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *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)*.

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

[3] 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 *Proceedings of the ICLR*.

[4] 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.

[5] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *Proceedings of the 12th International Conference on Learning Representations*.

[6] Zhangir Azerbayev, Ansong Ni, Hailey Schoelkopf, and Dragomir Radev. 2022. Explicit knowledge transfer for weakly-supervised code generation. arXiv:2211.16740. Retrieved from https://arxiv.org/abs/2211.16740

[7] Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. 2023. A multitask, multilingual, multimodal evaluation of ChatGPT on reasoning, hallucination, and interactivity. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*. Nusa Dua, Bali. Association for Computational Linguistics, 675–718.

[8] Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China. Association for Computational Linguistics, 1538–1548.

[9] 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.

[10] Rishabh Bhardwaj, Amrita Saha, and Steven C. H. Hoi. 2022. Vector-quantized input-contextualized soft prompts for natural language understanding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

[11] 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 *Proceedings of the International Conference on Machine Learning*. 2206–2240.

[12] 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.

[13] Lucas Page-Caccia, Edoardo Maria Ponti, Zhan Su, Matheus Pereira, Nicolas Le Roux, and Alessandro Sordoni. 2023. Multi-head adapter routing for cross-task generalization. *Advances in Neural Information Processing Systems* 36 (2023), 56916–56931.

[14] Yihan Cao, Siyu Li, Yixin Liu, Zhiling Yan, Yutong Dai, Philip Yu, and Lichao Sun. 2025. A survey of AI-generated content (aigc). *ACM Computing Surveys* 57, 5 (2025), 1–38.

[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. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 7870–7881.

[17] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research* (2023), 2835–8856.

[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. In *Proceedings of the ACM Web Conference 2022*.

[19] Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, et al. 2023. Binding language models in symbolic languages. In *Proceedings of the International Conference on Learning Representations*.

[20] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research* 24, 240 (2023), 1–113.

[21] Christiano et al. 2017. Deep reinforcement learning from human preferences. *NeurIPS 2017*. 4299–4307.

[22] Alexandra Chronopoulou, Matthew E. Peters, Alexander Fraser, and Jesse Dodge. 2023. AdapterSoup: Weight averaging to improve generalization of pretrained language models. In *Findings of the Association for Computational Linguistics: EACL*. 2054–2063.

[23] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research* 25, 70 (2024), 1–53.

[24] Hejie Cui, Jiaying Lu, Ran Xu, Shiyu Wang, Wenjing Ma, Yue Yu, Shaojun Yu, Xuan Kan, Chen Ling, Liang Zhao, Zhaohui S. Qin, Joyce C. Ho, Tianfan Fu, Jing Ma, Mengdi Huai, Fei Wang, and Carl Yang. 2025. A review on knowledge graphs for healthcare: Resources, applications, and promises. *Journal of Biomedical Informatics* 169 (2025), 104861.

[25] Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 8493–8502.

[26] Zhuyun Dai, Vincent Y. Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith Hall, and Ming-Wei Chang. 2022. Promptagator: Few-shot Dense Retrieval From 8 Examples. In *The Eleventh International Conference on Learning Representations*.

[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 *Proceedings of the 2nd 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 EMNLP*.

[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 *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

[30] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 4171–4186.
[31] 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 *Proceedings of the 11th International Conference on Learning Representations*.
[32] 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*.
[33] Ali Edalati, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J. Clark, and Mehdi Rezagholizadeh. 2025. KronA: Parameter-efficient tuning with kronecker adapter. In *Enhancing LLM Performance: Efficacy, Fine-Tuning, and Inference Techniques*. Cham: Springer Nature Switzerland, 49–65.
[34] Mohammad Fahes, Tuan-Hung Vu, Andrei Bursuc, Patrick Pérez, and Raoul De Charette. 2023. Poda: Prompt-driven zero-shot domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 18623–18633.
[35] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference*. 6491–6501.
[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. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*. PMLR, 10764–10799.
[38] C. Ge et al. 2022. Domain adaptation via prompt learning. In *IEEE Transactions on Neural Networks and Learning Systems* 36, 1 (2022), 1160–1170. DOI: 10.1109/TNNLS.2023.3327962
[39] Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017. Improving neural language models with a continuous cache. In *Proceedings of the 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)*. Dublin, Ireland, 8410–8423.
[41] Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: Word-level adversarial reprogramming. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.
[42] Hangfeng He, Hongming Zhang, and Dan Roth. 2022. Rethinking with retrieval: Faithful large language model inference. arXiv:2301.00303. Retrieved from https://arxiv.org/abs/2301.00303
[43] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.
[44] Shwai He, Liang Ding, Daize Dong, Jeremy Zhang, and Dacheng Tao. 2022. SparseAdapter: An easy approach for improving the parameter-efficiency of adapters. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. Association for Computational Linguistics, 2184–2190.
[45] Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear Units. In *International Conference on Learning Representations*.
[46] Raghav Garg, Mehul Manu, Neha Chauhan, Naveen Naval, and C. Tharini. 2024. Manipulation and measurement of knowledge representations of language models. In *2024 2nd International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS)*. IEEE, 1249–1254.
[47] Hou et al. 2024. Large language models for software engineering: A systematic literature review. *ACM TOSEM'24*. 1–79.
[48] 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 *Proceedings of the International Conference on Machine Learning*. PMLR, 2790–2799.
[49] Howard, Jeremy and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 328–339.
[50] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*. 8003–8017.
[51] Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*. 2022.
[52] 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*.

[53] Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2025. GRAG: Graph retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: NAACL 2025*. 4145–4157.

[54] Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. 2023. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 5254–5276.

[55] Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023. Large language models can self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 1051–1068.

[56] Shaohan Huang, Li Dong, Wenhui Wang, Yaru Hao, Saksham Singhal, Shuming Ma, Tengchao Lv, et al. 2023. Language is not all you need: Aligning perception with language models. *Advances in Neural Information Processing Systems* 36 (2023), 72096–72109.

[57] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.* 24, 1, Article 251 (January 2023).

[58] K. Jeblick, B. Schachtner, J. Dexl, A. Mittermeier, A. T. Stüber, J. Topalis, T. Weber, P. Wesp, B. O. Sabel, J. Ricke, and M. Ingrisch. 2022. ChatGPT makes medicine easy to swallow: an exploratory case study on simplified radiology reports. *Eur Radiol.* 34, 5 (2022), 2817–2825. DOI: 10.1007/s00330-023-10213-1

[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. *ACM Computing 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 *Proceedings of the 11th International Conference on Learning Representations*.

[62] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2177–2190.

[63] Feihu Jin, Jinliang Lu, Jiajun Zhang, and Chengqing Zong. 2023. Instance-aware prompt learning for language understanding and generation. *ACM Transactions on Asian and Low-Resource Language Information Processing* 22, 7 (2023), 1–18.

[64] Qiao Jin, Yifan Yang, Qingyu Chen, and Zhiyong Lu. 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] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*.

[68] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations*.

[69] Callie Y. Kim, Christine P. Lee, and Bilge Mutlu. 2024. Understanding large-language model (llm)-powered human-robot interaction. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*. 371–380.

[70] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Proceedings of the ICML 2022 Workshop on Knowledge Retrieval and Language Models*.

[71] 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.

[72] Harvey Lederman and Kyle Mahowald. 2024. Are language models more like libraries or like librarians? Bibliotechnism, the novel reference problem, and the attitudes of LLMs. *Transactions of the Association for Computational Linguistics* 12 (2024), 1087–1103.

[73] 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? In *Conference on Health, Inference, and Learning*. PMLR, 578–597.

[74] Markus Leippold. 2023. Sentiment spin: Attacking financial sentiment with GPT-3. *Finance Research Letters* 55, PB (2023).

[75] 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.

[76] Yoav Levine, Itay Dalmedigos, Ori Ram, Yoel Zeldes, Daniel Jannai, Dor Muhlgay, Yoni Osin, Opher Lieber, Barak Lenz, Shai Shalev-Shwartz, et al. 2022. Standing on the shoulders of giant frozen language models. arXiv:2204.10019. Retrieved from https://arxiv.org/abs/2204.10019

[77] 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.

[78] Daliang Li, Ankit Singh Rawat, Manzil Zaheer, Xin Wang, Michal Lukasik, Andreas Veit, Felix Yu, and Sanjiv Kumar. 2023. Large language models with controllable working memory. In *Findings of the Association for Computational Linguistics: ACL 2023*. 1774–1793.

[79] Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for "Mind" exploration of large language model society. *Advances in Neural Information Processing Systems* 36 (2023), 51991–52008.

[80] 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 *Proceedings of the International Conference on Computational Linguistics*.

[81] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, et al. 2023. Can llm already serve as a database interface? A big bench for large-scale database grounded text-to-SQLs. *Advances in Neural Information Processing Systems* 36 (2023), 42330–42357.

[82] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th ACL*.

[83] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Pete Florence, and Andy Zeng. 2022. Code as policies: Language model programs for embodied control. In *Proceedings of the Workshop on Language and Robotics at CoRL 2022*.

[84] Yaobo Liang, Chenfei Wu, Ting Song, Wenshan Wu, Yan Xia, Yu Liu, Yang Ou, et al. 2024. Taskmatrix. AI: Completing tasks by connecting foundation models with millions of apis. *Intelligent Computing* 3 (2024), 0063.

[85] Hongzhan Lin, Pengyao Yi, Jing Ma, Haiyun Jiang, Ziyang Luo, Shuming Shi, and Ruifang Liu. 2023. Zero-shot rumor detection with propagation structure via prompt learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 4 (2023), 5213–5221.

[86] Chen Ling, Xuchao Zhang, Xujiang Zhao, Yanchi Liu, Wei Cheng, Mika Oishi, Takao Osaki, Katsushi Matsuda, Haifeng Chen, and Liang Zhao. 2023. Open-ended commonsense reasoning with unrestricted answer candidates. In *Findings of EMNLP*. 8035–8047.

[87] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A. Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems* 35 (2022), 1950–1965.

[88] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *CSUR* 55, 9 (2023), 1–35.

[89] Qi Liu, Dani Yogatama, and Phil Blunsom. 2022. Relational memory-augmented language models. *Transactions of the Association for Computational Linguistics* 10 (2022), 555–572.

[90] Ruibo Liu, Jason Wei, Shixiang Shane Gu, Te-Yen Wu, Soroush Vosoughi, Claire Cui, Denny Zhou, and Andrew M. Dai. 2023. Mind's Eye: Grounded language model reasoning through simulation. In *The Eleventh International Conference on Learning Representations*.

[91] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 61–68.

[92] 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*. Abu Dhabi, UAE.

[93] Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, et al. 2023. Summary of chatgpt-related research and perspective towards the future of large language models. *Meta-Radiology* 1, 2 (2023), 100017.

[94] Alejandro Lopez-Lira and Yuehua Tang. 2023. Can ChatGPT forecast stock price movements? return predictability and large language models. arXiv:2304.07619. Retrieved from https://arxiv.org/abs/2304.07619

[95] 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 *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval - Short Paper*.

[96] 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.

[97] Rupert Macey-Dare. 2023. ChatGPT and generative AI systems as quasi-expert legal advice lawyers-case study considering potential appeal against conviction of tom hayes. *Available at SSRN 4342686* (2023), 35.

[98] Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. 2022. Language models of code are few-shot commonsense learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 1384–1403.

[99] B. Mahjour, J. Hoffstadt, and T. Cernak. 2023. Designing chemical reaction arrays using phactor and ChatGPT. *Organic Process Research & Development* 27, 8 (2023), 1510–1516. https://doi.org/10.1021/acs.oprd.3c00186

[100] Bhavitvya Malik, Abhinav Ramesh Kashyap, Min-Yen Kan, and Soujanya Poria. 2023. UDAPTER-efficient domain adaptation using adapters. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. 2249–2263.

[101] Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. 2022. UniPELT: A unified framework for parameter-efficient language model tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 6253–6264.

[102] Raja Marjieh, Ilia Sucholutsky, Pol van Rijn, Nori Jacoby, and Tom Griffiths. 2023. What language reveals about perception: Distilling psychophysical knowledge from large language models. In *Proceedings of the Annual Meeting of the Cognitive Science Society* 45, 45 (2023).

[103] 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.

[104] Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. 2022. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*.

[105] 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:2203.11147. Retrieved from https://arxiv.org/abs/2203.11147

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

[107] Grégoire Mialon, Roberto Dessi, Maria Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, Roberta Raileanu, Baptiste Roziere, et al. 2023. Augmented language models: A survey. *Transactions on Machine Learning Research*. 2835–8856.

[108] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. 2023. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys* 56, 2 (2023), 1–40.

[109] 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:2202.12837. Retrieved from https://arxiv.org/abs/2202.12837

[110] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022. Fast model editing at scale. In *International Conference on Learning Representations*.

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

[112] 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* 22, 7 (2023), zvad022–zvad022.

[113] 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 *Proceedings of the NeurIPS Efficient Natural Language and Speech Processing Workshop*.

[114] Prasanth Murali, Ian Steenstra, Hye Sun Yun, Ameneh Shamekhi, and Timothy Bickmore. 2023. Improving multiparty interactions with a robot using large language models. In *Proceedings of the Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–8.

[115] 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:2112.09332. Retrieved from https://arxiv.org/abs/2112.09332

[116] P. Niszczota and S. Abbas. 2023. GPT has become financially literate: Insights from financial literacy tests of GPT and a preliminary test of how people use it as a source of advice. SSRN Working Paper, Abstract ID 4384861. https://doi.org/10.2139/ssrn.4384861

[117] OpenAI. ChatGPT plugins. Retrieved from https://openai.com/blog/chatgpt-plugins Accessed: 2023-04-05.

[118] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774. Retrieved from https://arxiv.org/abs/2303.08774
[119] 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.
[120] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual Acm Symposium on User Interface Software and Technology*. 1–22.
[121] Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. arXiv:2302.12813. Retrieved from https://arxiv.org/abs/2302.12813
[122] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. arXiv:2304.03277. Retrieved from https://arxiv.org/abs/2304.03277
[123] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 487–503.
[124] Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 46–54.
[125] Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An adapter-based framework for multi-task cross-lingual transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 7654–7673.
[126] Viet Pham, Shilin Qu, Farhad Moghimifar, Suraj Sharma, Yuan-Fang Li, Weiqing Wang, and Reza Haf. 2024. Multi-cultural norm base: Frame-based norm discovery in multi-cultural settings. In *Proceedings of the 28th Conference on Computational Natural Language Learning*. 24–35.
[127] Flor Plaza-del Arco, Amanda Curry, Susanna Paoli, Alba Cercas Curry, and Dirk Hovy. 2024. Divine LLaMAs: Bias, stereotypes, stigmatization, and emotion representation of religion in large language models. In *Findings of EMNLP 2024*. 4346–4366.
[128] Edoardo Maria Ponti, Alessandro Sordoni, Yoshua Bengio, and Siva Reddy. 2023. Combining parameter-efficient modules for task-level generalisation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. 687–702.
[129] Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction. *Advances in Neural Information Processing Systems* 36 (2023), 36339–36348.
[130] 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.
[131] Chengwei Qin and Shafiq Joty. 2022. LFPT5: A unified framework for lifelong few-shot language learning based on prompt tuning of T5. In *International Conference on Learning Representations*.
[132] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, et al. 2024. Tool learning with foundation models. *ACM Computing Surveys* 57, 4 (2024), 1–40.
[133] 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.
[134] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. 2018. Improving language understanding by generative pre-training. OpenAI Technical Report. Available at: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf
[135] 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.
[136] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics* 11 (2023), 1316–1331.
[137] Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2023. Progressive prompts: Continual learning for language models. In *Proceedings of the 11th International Conference on Learning Representations*.
[138] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. *Advances in Neural Information Processing Systems* 30 (2017), 506–516.
[139] Benjamin Reichman and Larry Heck. 2024. Dense passage retrieval: Is it Retrieving? In *Findings of the Association for Computational Linguistics: EMNLP 2024*. 13540–13553.

[140] Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 5418–5426.

[141] Joshua Robinson, Christopher Rytting, and David Wingate. 2023. Leveraging large language models for multiple choice question answering. In *International Conference on Learning Representations*.

[142] 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.

[143] 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.

[144] Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, et al. 2022. Multitask prompted training enables zero-shot task generalization. In *Proceedings of the ICLR*.

[145] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems* 36 (2023), 68539–68551.

[146] Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 255–269.

[147] Dale Schuurmans. 2023. Memory augmented large language models are computationally universal. arXiv:2301.04589. Retrieved from https://arxiv.org/abs/2301.04589

[148] 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.

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

[150] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. HuggingGPT: Solving AI tasks with ChatGPT and its friends in hugging face. *Advances in Neural Information Processing Systems* 36 (2023), 38154–38180.

[151] 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), 2990–2999.

[152] Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2023. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics: ACL 2023*. 7059–7073.

[153] 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.

[154] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. ProgPrompt: Generating situated robot task plans using large language models. In *Proceedings of the Workshop on Language and Robotics at CoRL 2022*.

[155] Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. 2024. DRAGIN: Dynamic retrieval augmented generation based on the real-time information needs of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 12991–13013.

[156] 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.

[157] Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. 2022. Black-Box tuning for language-model-as-a-service. In *Proceedings of the International Conference on Machine Learning*.

[158] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. *Advances in Neural Information Processing Systems* 35 (2022), 12991–13005.

[159] Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. Vipergpt: Visual inference via Python execution for reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 11888–11898.

[160] Yan Tao, Olga Viberg, Ryan S. Baker, and René F. Kizilcec. 2024. Cultural bias and cultural alignment of large language models. *PNAS Nexus* 3, 9 (2024), page 346.

[161] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: A family of highly capable multimodal models. arXiv:2312.11805. Retrieved from https://arxiv.org/abs/2312.11805

[162] Shubo Tian, Qiao Jin, Lana Yeganova, Po-Ting Lai, Qingqing Zhu, Xiuying Chen, et al. 2024. Opportunities and challenges for ChatGPT and large language models in biomedicine and health. *Briefings in Bioinformatics* 25, 1 (2024), bbad493.

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

[164] Dietrich Trautmann, Alina Petrova, and Frank Schilder. 2022. Legal prompt engineering for multilingual legal judgement prediction. arXiv:2212.02199. Retrieved from https://arxiv.org/abs/2212.02199

[165] Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2023. DyLoRA: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. 3274–3287.

[166] 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.

[167] 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), 5998–6008.

[168] Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2022. SPoT: Better frozen model adaptation through soft prompt transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 5039–5059.

[169] 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. *ICML 2022 Workshop on Hardware-Aware Efficient Training (HAET 2022)*.

[170] 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 *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

[171] Longyue Wang, Chenyang Lyu, Tianbo Ji, Zhirui Zhang, Dian Yu, Shuming Shi, and Zhaopeng Tu. 2023. Document-level machine translation with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 16646–16661.

[172] 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 *Proceedings of the Advances in Neural Information Processing Systems*.

[173] Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuan-Jing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2021. K-adapter: Infusing knowledge into pre-trained models with adapters. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. 1405–1418.

[174] Sheng Wang, Zihao Zhao, Xi Ouyang, Tianming Liu, Qian Wang, and Dinggang Shen. 2024. Interactive computer-aided diagnosis on medical image using large language models. *Communications Engineering* 3, 1 (2024), 133.

[175] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Rationale-augmented ensembles in language models. arXiv:2207.00747. Retrieved from https://arxiv.org/abs/2207.00747

[176] 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 *Proceedings of the 11th International Conference on Learning Representations*.

[177] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi. 2023. Self-Instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL 2023)*, Long Papers, 13484–13508. https://doi.org/10.18653/v1/2023.acl-long.754

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

[179] 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:2205.12410. Retrieved from https://arxiv.org/abs/2205.12410

[180] 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:2211.00635. Retrieved from https://arxiv.org/abs/2211.00635

[181] Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, Yitao Liang, and Team CraftJarvis. 2023. Describe, explain, plan and select: interactive planning with large language models enables open-world multi-task agents. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*. 34153–34189.

[182] 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 *Proceedings of the International Conference on Learning Representations*.

[183] 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*.

[184] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, et al. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*. 2835–8856.

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

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

[187] 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:2303.17564. Retrieved from https://arxiv.org/abs/2303.17564

[188] 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 *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

[189] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient Streaming Language Models with Attention Sinks. In *Proceedings of the 12th International Conference on Learning Representations*.

[190] 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. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 9514–9528.

[191] 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 *Proceedings of the Findings of the Association for Computational Linguistics: NAACL 2022*.

[192] 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).

[193] Kai-Cheng Yang and Filippo Menczer. 2025. Accuracy and political bias of news source credibility ratings by large language models. In *Proceedings of the 17th ACM Web Science Conference 2025*. 127–137.

[194] Xianjun Yang, Wei Cheng, Xujiang Zhao, Linda Petzold, and Haifeng Chen. 2023. Dynamic Prompting: A Unified Framework for Prompt Tuning. arXiv:2303.02909. Retrieved from https://arxiv.org/abs/2303.02909

[195] Zonghan Yang, Xiaoyuan Yi, Peng Li, Yang Liu, and Xing Xie. 2023. Unified detoxifying and debiasing in language generation via inference-time adaptive optimization. In *The Eleventh International Conference on Learning Representations*.

[196] Fuda Ye, Shuangyin Li, Yongqi Zhang, and Lei Chen. 2024. R2AG: Incorporating retrieval information into retrieval augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. 11584–11596.

[197] 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. In *Proceedings of the ACM Web Conference 2022*.

[198] Fangyi Yu, Lee Quartey, and Frank Schilder. 2022. Legal Prompting: Teaching a Language Model to Think Like a Lawyer. arXiv:2212.01326. Retrieved from https://arxiv.org/abs/2212.01326

[199] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 1–9.

[200] 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:2303.11717. Retrieved from https://arxiv.org/abs/2303.11717

[201] 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. *Advances in Neural Information Processing Systems* 35 (2022), 21442–21454.

[202] Renrui Zhang, Jiaming Han, Chris Liu, Peng Gao, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, and Yu Qiao. 2024. LLaMA-Adapter: Efficient fine-tuning of language models with zero-init attention. In *The International Conference on Learning Representations*.

[203] Rongsheng Zhang, Yinhe Zheng, Xiaoxi Mao, and Minlie Huang. 2021. Unsupervised domain adaptation with adapter. In *Efficient Natural Language and Speech Processing (Models, Training, and Inference), 35th Conference on Neural Information Processing Systems (NeurIPS 2021)*.

[204] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

[205] Hongyu Zhao, Hao Tan, and Hongyuan Mei. 2022. Tiny-attention adapter: Contexts are more important than the number of parameters. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 6626–6638.

[206] 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:2303.18223. Retrieved from https://arxiv.org/abs/2303.18223

[207] 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 *Proceedings of the ICLR*.

[208] Han Zhou, Xingchen Wan, Lev Proleev, Diana Mincu, Jilin Chen, Katherine A. Heller, and Subhrajit Roy. 2024. Batch Calibration: Rethinking calibration for in-context learning and prompt engineering. In *The Twelfth International Conference on Learning Representations*.

[209] Le Zhuo, Zewen Chi, Minghao Xu, He-Yan Huang, Jianan Zhao, Heqi Zheng, Conghui He, Xian-Ling Mao, and Wentao Zhang. 2024. ProtLLM: An interleaved protein-language LLM with protein-as-word pre-training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 8950–8963.

[210] 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:1909.08593. Retrieved from https://arxiv.org/abs/1909.08593

Received 12 July 2023; revised 14 January 2025; accepted 18 May 2025