PERSONAL LLM AGENTS: INSIGHTS AND SURVEY ABOUT THE CAPABILITY, EFFICIENCY AND SECURITY

Yuanchun Li[1†], **Hao Wen**[1‡], **Weijun Wang**[1‡], **Xiangyu Li**[1‡], **Yizhen Yuan**[1‡], **Guohong Liu**[1‡], **Jiacheng Liu**[1], **Wenxing Xu**[1], **Xiang Wang**[1], **Yi Sun**[1], **Rui Kong**[1], **Yile Wang**[1], **Hanfei Geng**[1], **Jian Luan**[2], **Xuefeng Jin**[3], **Zilong Ye**[4], **Guanjing Xiong**[5], **Fan Zhang**[6], **Xiang Li**[7], **Mengwei Xu**[8], **Zhijun Li**[9], **Peng Li**[1], **Yang Liu**[1], **Ya-Qin Zhang**[1], **Yunxin Liu**[1]

[1] Institute for AI Industry Research (AIR), Tsinghua University
[2] Xiaomi AI Lab [3] Huawei Technologies Co., Ltd. [4] Shenzhen Heytap Technology Co., Ltd.
[5] vivo AI Lab [6] Viomi Technology Co., Ltd. [7] Li Auto Inc.
[8] Beijing University of Posts and Telecommunications [9] Soochow University

[†] Project Lead [‡] Section Lead
Contact: liyuanchun@air.tsinghua.edu.cn
Website: https://github.com/MobileLLM/Personal_LLM_Agents_Survey

ABSTRACT

Since the advent of personal computing devices, intelligent personal assistants (IPAs) have been one of the key technologies that researchers and engineers have focused on, aiming to help users efficiently obtain information and execute tasks, and provide users with more intelligent, convenient, and rich interaction experiences. With the development of the smartphone and Internet of Things, computing and sensing devices have become ubiquitous, greatly expanding the functional boundaries of IPAs. However, due to the lack of capabilities such as user intent understanding, task planning, tool using, and personal data management etc., existing IPAs still have limited practicality and scalability.

Recently, the emergence of foundation models, represented by large language models (LLMs), brings new opportunities for the development of IPAs. With the powerful semantic understanding and reasoning capabilities, LLM can enable intelligent agents to solve complex problems autonomously. In this paper, we focus on *Personal LLM Agents*, which are LLM-based agents that are deeply integrated with personal data and personal devices and used for personal assistance. We envision that Personal LLM Agents will become a major software paradigm for end-users in the upcoming era. To realize this vision, we take the first step to discuss several important questions about Personal LLM Agents, including their architecture, capability, efficiency and security. We start by summarizing the key components and design choices in the architecture of Personal LLM Agents, followed by an in-depth analysis of the opinions collected from domain experts. Next, we discuss several key challenges to achieve intelligent, efficient and secure Personal LLM Agents, followed by a comprehensive survey of representative solutions to address these challenges.

Keywords Intelligent personal assistant · Large language model · LLM agent · Mobile devices · Intelligence levels · Task automation · Sensing · Memory · Efficiency · Security and privacy

Contents

1 Introduction

Science fiction has portrayed numerous striking characters of Intelligent Personal Assistants (IPAs), which are software agents that can augment individuals' abilities, complete complicated tasks, and even satisfy emotional needs. These intelligent agents represent most people's fantasies regarding artificial intelligence (AI). With the widespread adoption of personal devices (*e.g.,* smartphones, smart home equipment, electric vehicles, etc.) and the advancement of machine learning technology, this fantasy is gradually becoming the reality. Today, many mobile devices embeds IPA software, such as Siri [1], Google Assistant [2], Alexa [3], etc. These intelligent agents are deeply entwined with users, capable of accessing user data and sensors, controlling various personal devices, and accessing personalized services associated with private accounts.

However, today's intelligent personal assistants still suffer from the limitations of flexibility and scalability. Their level of intelligence is far from adequate, particularly evident in their understanding of user intent, reasoning, and task execution. Most of today's intelligent personal assistants are limited to performing tasks within a restricted domain (*e.g.,* simple functions in built-in apps). Once a user requests for tasks beyond these boundaries, the agent fails to comprehend and execute the actions accurately. Altering this circumstance necessitates a significant expansion of the agent's capability to support a broader and more flexible scope of tasks. However, it is difficult for current IPA products to support tasks at scale. Most of the today's IPAs require to follow specific predefined rules to complete tasks, such as developer-defined or user-demonstrated steps. Therefore, developers or users must explicitly specify which functions they wish to support, in addition to defining the triggers and steps for task execution. This approach inherently restricts the scalability to wider range of tasks, since supporting more tasks demands extensive time and labor cost. Some approaches have attempted to automatically learn to support tasks through supervised learning or reinforcement learning [4, 5, 6]. However, these methods also rely on a substantial amount of manual demonstrations and/or the definition of reward functions.

The emergence of Large Language Models (LLMs) [7] in recent years has brought brand new opportunities for the development of IPAs, demonstrating the potential to address the scalability issues of intelligent personal assistants. In comparison to traditional methods, large language models such as ChatGPT, Claude, and others have exhibited unique capabilities such as instruction following, commonsense reasoning, and zero-shot generalization. These abilities have been achieved through unsupervised learning on massive corpora (exceeding 1.4 trillion words) and subsequently fine-tuned with human feedback. Leveraging these capabilities, researchers have successfully adopted large language models to empower autonomous agents (aka. LLM agents), which aims to solve complex problems by automatically making plans and using tools such as search engines, code interpreters, and third-party APIs.

As a unique type of intelligent agents, IPAs also have the potential to be revolutionized by LLMs with significantly enhanced scalability, capability, and usefulness. We call such LLM-powered intelligent personal assistants as **Personal LLM Agents**. As compared with normal LLM agents, Personal LLM Agents are more deeply engaged with personal data and mobile devices, and are more explicitly designed for assisting people rather than replacing people. Specifically, the primary way to assist users is by reducing repetitive, tedious, and low-value labor in their daily routine, letting the users focus on more interesting and valuable things, thereby enhancing the efficiency and quality of their work and life. Personal LLM Agents can be built upon existing software stacks (*e.g.,* mobile apps, websites, etc.), while bringing refreshing user experience with ubiquitous intelligent automation abilities. Therefore, we expect Personal LLM Agents to become a major software paradigm for personal computing devices in the AI era, as shown in Figure 1.

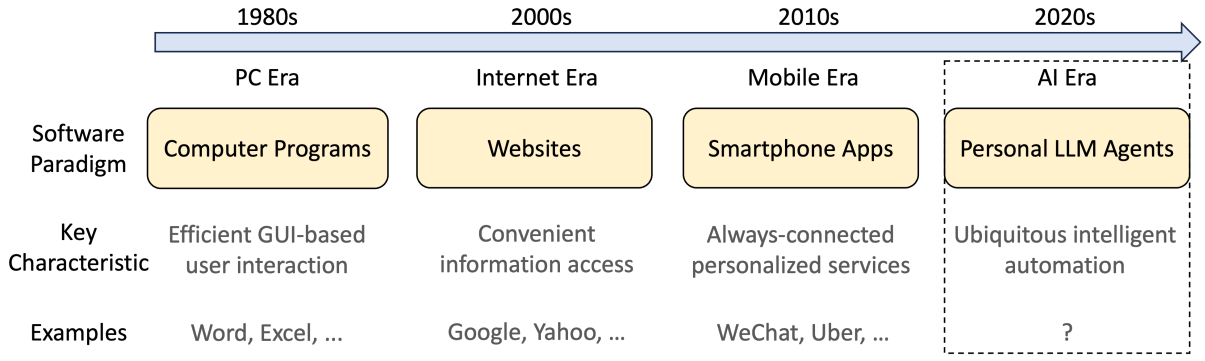

Figure 1: We envision **Personal LLM Agents** to become the dominating software paradigm for individual users in the upcoming era.

Despite the promising future of Personal LLM Agents, related research is still in its nascent stage, presenting numerous intricacies and challenges. This paper takes the first step to discuss the route map, design choices, main challenges and possible solutions in implementing Personal LLM Agents. Specifically, we focus primarily on the aspects related to "*personal*" parts within Personal LLM Agents, encompassing the analysis and utilization of users' personal data, the use of personal resources, deployment on personal devices, and the provision of personalized services. The straightforward integration of the general language capabilities of LLMs into IPAs is not within the scope of this paper.

We started by taking a survey with domain experts of Personal LLM Agents. We invited 25 chief architects, managing directors, and/or senior engineers/researchers from leading companies who are working on IPAs and/or LLMs on personal devices. We asked the experts' opinions about the opportunities and challenges of integrating LLMs in their consumer-facing products. Based on our understanding and analyses of experts' insights, we summarized a simple and generic architecture of Personal LLM Agents, in which the intelligent management and utilization of personal data (user context, environment status, activity history, personalities, etc.) and personal resources (mobile apps, sensors, smart-home devices, etc.) play the most vital role. The ability to manage and utilize these personal objects differentiates the intelligence of Personal LLM Agents. Inspired by the L1-L5 intelligence levels of autonomous driving, we also give an taxonomy of five intelligent levels of Personal LLM Agents.

Our findings also highlight several major technical challenges to implement such Personal LLM Agents, which can be categorized into three aspects including the fundamental capabilities, efficiency, and security & privacy. We further dive deeper into these aspects with detailed explanations of the challenges and comprehensive survey of possible solutions. Specifically, for each technical aspect, we briefly explain its relevance and importance to personal LLM agents, then break it down to several main research problems. For example, the foundamental capabilities for personal LLM agents include task execution, context sensing, and memorization. The efficiency of agents is primarily determined by the LLM inference efficiency, customization efficiency, and memory retrieval efficiency. The security and privacy concerns of personal LLM agents can be categorized as data confidentiality, decision reliability, and system integrity. For each research problem, we summarize the main techniques involved with the problem, followed by a brief introduction of the related work. Due to the wide scope of the techniques in personal LLM agents, we only include the most relevant or recent works, rather than attempting to cover all related approaches.

The main content and contributions of this paper can be summarized as follows:

1. We summarize the status quo of existing intelligent personal assistants in both industry and academia, while analyzing their primary limitations and future trends in the LLM era.
2. We collect insights from senior domain experts in the area of LLM and personal agents, proposing a generic system architecture and a definition of intelligence levels for personal LLM agents.
3. We review the literature on three important technical aspects of personal LLM agents, including foundamental capabilities, efficiency, and security & privacy.

2 A Brief History of Intelligent Personal Assistants

2.1 Timeline View of the Intelligent Personal Assistants History

Intelligent Personal Assistants (IPAs) have a long history of development. We depict the rough timeline of the IPA history in Figure 2. The development progress can be divided into four stages, each marked with a unique color in the figure.

The 1st stage spans from the 1950s to the late 1980s, which is mainly about the development of speech recognition techniques. The early stage of speech recognition started from basic digits and words. Bell Laboratories developed "Audrey", which could recognize numbers 0-9 with about 90% accuracy. In 1962, the "shoebox" [8] system came out from Advanced Systems Development Division Laboratory at IBM, which was capable to recognize for up to 16 words. From 1971 to 1976, the Speech Understanding Research (SUR) project, funded by the US Department of Defense, significantly advanced speech recognition technology. The Harpy system [9] was particularly representative, as it could understand sentences composed of 1011 words, equivalent to the proficiency of a three-year-old child. In 1986, IBM developed the Tangora speech recognition typing system [10], capable of recognizing 20,000 words and offering predictive and error-correction capabilities. The Tangora system utilized Hidden Markov Models [11], requiring individual speaker training for voice recognition, with pauses between each word.

The 2nd stage covers the period from the 1990s to the late 2000s, since speech recognition started to be integrated into software for certain advanced functions. In 1990, the "Dragon Dictate" software [12] was released, which was the first speech recognition product for consumers. It was originally designed to work on Microsoft Windows, supporting discrete speech recognition. "Speakable items" [13] was introduced by Apple in 1993, enabling users to control their

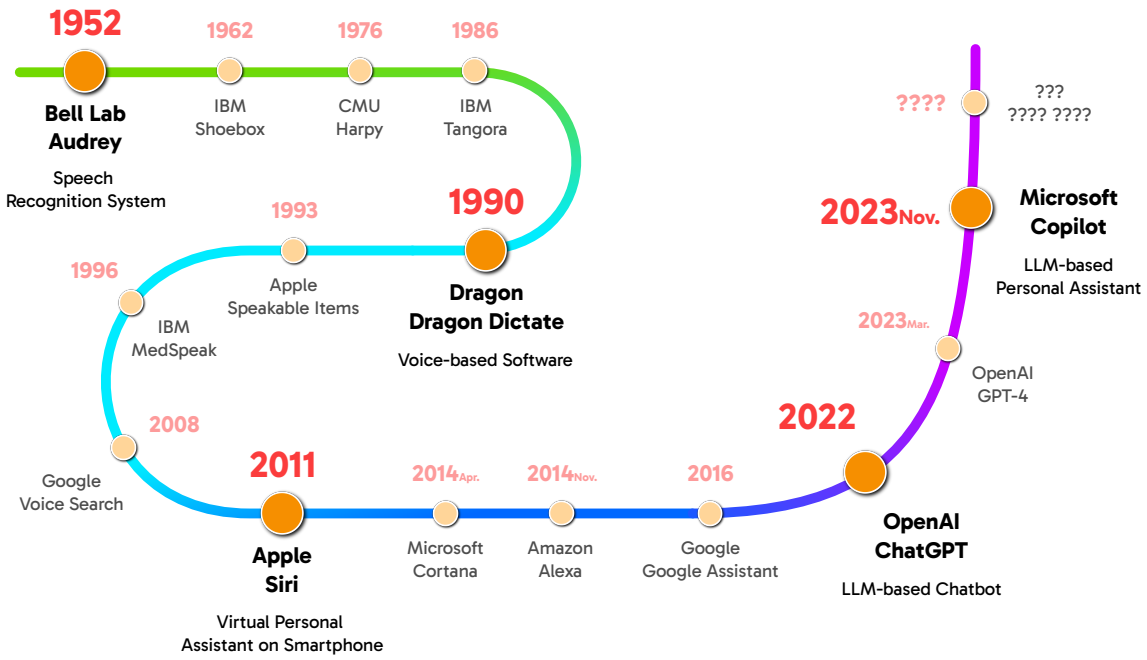

Figure 2: Major milestones in the history of intelligent personal assistants (IPAs). We mark different development stages with different colors, and some **significant or ground-breaking** events are highlighted with **bold text**.

computer with natural speaking. In 1996, IBM launched "MedSpeak" [14] for radiologists, which is also the first commercial product supporting continuous speech recognition. Microsoft integrated speech recognition into Office applications in 2002 [15], and Google added voice search to Google Mobile App on iPhone in 2008 [16].

The 3rd stage extends from the early 2010s. In this period, always-on virtual assistant services began to appear on mobile devices such as smartphones and personal computers. Siri [1], widely considered as the first intelligent personal assistant installed on modern smartphones, was integrated into Apple's iPhone 4S in 2011. Since its launch, Siri has remained a key built-in software for Apple devices, including iPhones, iPad, Apple Watch, HomePod and Mac, continuously undergoing updates and iterations to incorporate new features. Similar to Siri, many other virtual intelligent assistant started to appear in the period. In 2014, Microsoft released Cortana [17], and gradually integrated it into desktop computers and other platforms. Amazon released Alexa [3] in the same year, which could complete tasks such as voice interaction, music playing, setting alarms, etc. Beyond voice search, Google Assistant [2] was unveiled in 2016, supporting users to interact with both speaking and keyboard input.

The 4th stage started recently when LLMs start to draw attention from all over the world. Based on LLMs, there emerged many intelligent chatbots (*e.g.,* ChatGPT [18]), as well as some LLM-powered IPA software installed on personal devices (*e.g.,* Copilot [19]). The details of this stage will be covered in Section 2.2.4.

2.2 Technical View of the Intelligent Personal Assistants History

Since there are many aspects that can reflect the intelligence of personal assistants, we select one of the most important ability of Intelligent Personal Assistants, namely the task automation ability (following instructions and completing tasks), to be mainly focused on. In the following subsections, we will introduce four main types of techniques to enable intelligent task automation in IPA. Note that these types of solutions have been developing concurrently, and there is no strict chronological order between them.

2.2.1 Template-based Programming

Most of the commercial IPA products support task automation through template-based approaches. In these approaches, the functions that can be automated are predefined as templates, each of which usually contains the task description, related actions, example queries to match, supported parameters to fullfil, etc. Given a user command, the agent first map the command to the most relevant template, then follow the predefined steps to complete the task. The workflow is illustrated in Figure 3.

When using this method to automate tasks, app developers are required to follow the document of certain APIs (*e.g.,* the Google Assistant API [2], SiriKit [20], etc.) to create the template for each function they want to automate. Besides, some approaches are proposed to enable end-users to create their own templates of tasks, such as the "Shortcuts" [21] feature on iPhone devices, enabling the automation of repetitive operation sequences. Similar functions are also implemented in many products and academic research for the Android system, such as Tasker [22], Anywhere [23], Epidosite [24] and Microsoft's uLink [25] system, etc.

The advantages of such template-based task automation method lie in its reliability and accuracy, since the steps in the template are deterministic and carefully programmed. However, its scalability is pretty limited, because of the relatively complex mechanism for supporting new tasks. As a result, most apps, including the popular apps from large companies, do not support any automated task or only support some elementary ones, leading to very unflexible user experience. End-users can easilly give up the idea to use IPAs after several unsuccessful attempts [26, 27, 28, 29]. This limitation poses a major obstacle to the further development of template-based intelligent personal assistants.

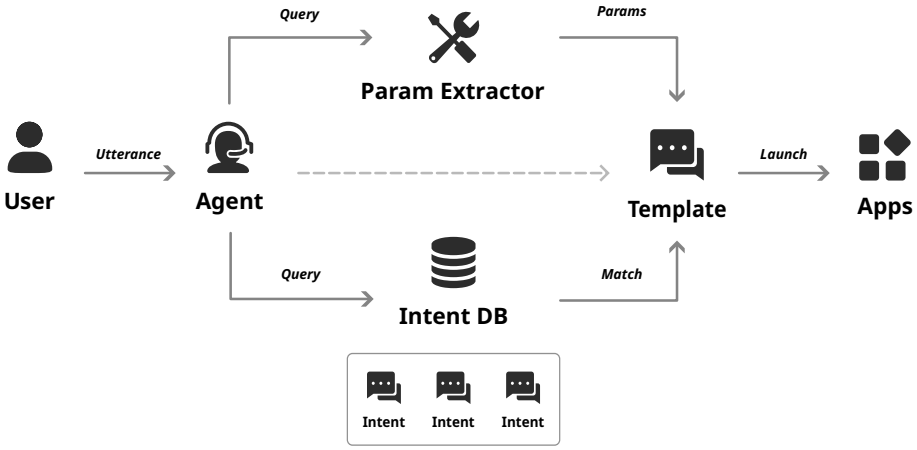

Figure 3: The workflow of template-based task automation.

2.2.2 Supervised Learning Methods

To address the constraints of template-based IPA methods, researchers are actively investigating automated approaches for enhanced UI understanding and automation. Supervised learning offers a direct method for task automation by training models that predicts subsequent actions and states based on task inputs and current states. The main research questions include how to learn a representation of software GUI and how to train the interaction model.

The idea of learning an interaction model from human interaction traces is introduced in Humanoid [30], which aims to generate human-like test inputs based on the GUI layout information. Seq2act [4] firstly focused on the mobile UI task automation domain, where the natural language instructions need to be mapped to a sequence of actions that can be directly executed. The framework decomposed the problem into an *action phrase-extraction* part and a *grounding* part, both using the Transformer [31] network. Inspired by the success of pretraining in NLP, ActionBert [32] uses self-supervised pretraining to enhance the model's understanding of UIs. Specifically, to capture the semantics information of the UI switching actions, the model is designed to take a pair of UIs as input, and output embeddings of both UIs and individual components. Aimed at better compatibility with the restricted resource on mobile devices, Versatile UI Transformer (VUT) [33] was proposed to learn different UI grounding tasks within a single small model. It handles images, structures, and text-based types of data, using 3 task heads to support performing 5 distinct tasks simultaneously, including UI object detection, natural language command grounding, widget captioning, screen summarization and UI tappability prediction. Based on the self-aligned characteristics between components of different modalities, UIBert [34] presented a well-designed joint image-text model to utilize the correspondence, learning contextual UI embeddings from unlabeled data. To address the problem of lacking UI metadata, such as DOM tree and view hierarchy, SpotLight [35] introduced a vision-only approach for mobile UI understanding by taking screenshots and a region of interest (the "focus") as input. Composed of a vision encoder and a language decoder, it can complete tasks according to the provided screenshot and prompt. Besides, Lexi [36] was proposed to leverage text-based instruction manuals and user guides to curate a multimodal dataset. By fusing text and visual features as input to the co-attention transformer layers, the model is pre-trained to form connections between text-based instructions and UI screenshots. UINav [37] utilized a referee model to evaluate the performance of the agent, immediately inform the users of the feedback. It also adopted demonstration augmentation to increase the data diversity.

As compared with template-based methods, supervised learning approaches have the potential to generalize to unseen tasks after sufficient training. However, training the model typically requires a lot of high-quality human-annotated data. Given the diversity of tasks and apps in the real world, obtaining the training data that covers diverse use cases is challenging.

2.2.3 Reinforcement Learning Methods

Unlike supervised learning-based task automation approaches that require a large amount of training samples, reinforcement learning (RL)-based approaches allows the agent to acquire the capability of task automation by continuously interacting with the target interfaces. During the interaction, the agent gets feedback of rewards that indicate the progress of task completion, and it gradually learns how to automate the tasks by maximizing the reward payoff.

To train RL-based task automation agents, a reward function that indicates the progress towards task completion is required. World of Bits (WoB) [38] was proposed as a general platform for agents to complete tasks on the Web using keyboard and mouse. The platform came with a benchmark called "MiniWoB", containing tasks on a set of self-created toy websites with predefined rewards. Glider [5] defines the reward function for real-world websites based on the semantic similarity between the task description and the UI action sequence, as well as the *locality* and *directionality* of the action sequence.

Another challenge of RL-based task automation is the huge action space and the sparse reward. A typical GUI-grounded task usually involves 5-10 steps, each of which contains 10-100 candidate actions, leading to a search space size of 10^5-100^{10}. The task is completed only if the correct sequence of actions is taken. In order to tackle such challenge, many frameworks have been proposed. Liu et al. [6] introduced the method to use high-level "*workflows*" to constrain the allowable actions at each time step. The workflows can prune out bad exploration directions, accelerating the agent's ability to discover rewards. Gur et al. [39] *decomposed* the complicated instruction into multiple smaller ones, and schedule a curriculum for the agents to gradually manage to follow an increasing number of sub-instructions. Besides, a meta-learning framework is also proposed to generate instruction-following tasks. Jia et al. [40] framed the actions of agent on the web into three distince categories, namely, DOM selection, token selection, and mode selection. What's more, a factorized Q-value function is designed, assuming the *independence* of DOM selection and token selection. Glider [5] achieves its goal of reducing action space with a hierachical policy, which contains a master policy to handle the overall navigation and sub-policies to deal with specific widgets. Humphreys et al. [41] proposed the framework to directly use mouse and keyboard to complete tasks instead of depending on the specialized action spaces, which simplifies the use of behavioural priors informed by actual human-computer interactions

Similar to supervised learning methods, the RL-based methods also suffer from poor generalization ability. To achieve flexible and robust task automation, the RL agent needs to train on a large amount of tasks, each requires a well-designed reward function. Defining the reward functions for massive diverse tasks can be difficult.

2.2.4 Early Adoption of Foundation Models

In recent years, pretrained large fundation models, represented by large language models (LLMs), have seen rapid development and brought new opportunities for personal assistants.

The scaling law [42] for language models reveals the importance of increasing model parameters for improving model performance, followed by a bunch of models with billions of parameters. The LLMs are typically trained with large-scale open-domain text data in an unsupervised manner, followed by instruction fine-tuning [43] and reinforcement learning with human feedback (RLHF) [44, 43] to improve performance and alignment. ChatGPT [18] unveiled by OpenAI at the end of 2022 is a milestone of LLM that demonstrated astounding question-answering capabilities. By feeding simple task descriptions into the LLM as input prompts, the tasks and responses of LLMs can be easily customized. Besides, these models have also demonstrated robust generalization abilities across various language understanding and reasoning tasks. ChatGPT itself can be viewed as an intelligent personal assistant that assist users by returning information in text responses.

Inspired by the capabilities of LLMs, researchers have attempted to let LLMs use tools [45] autonomously to accomplish complex tasks. For instance, such as controlling browsers [46, 47] for information retrieval and summarization, invoking robot programming interfaces for robot behavior control [48, 49, 50], and calling code interpreters for complex data processing [51, 52, 53, 54], among others. It is a natural idea to integrate these capabilities into intelligent personal assistants, enabling more intelligent ways to manipulate personal data, personal devices and personalized services.

There are already some commercial products that have attempted to integrate LLM with IPA. For instance, Microsoft's Copilot system [19] has integrated the capabilities of GPT-4 [55], assisting users of Windows in automatically drafting documents, creating presentations, summarizing emails, and thereby enhancing user work efficiency. New Bing

[56] also improves the experience of surfing the internet, providing a powerful efficient search engine which better understands what users want. Similarly, Google has integrated LLMs (Bard [57], Gemini [58]) into the search engine to enable more convenient web search experience. Smartphone companies including Huawei, Xiaomi, Oppo, Vivo have also integrated large models (PanGu [59], MiLM [60], etc.) into their on-device IPA products. It is worth noting that some of them adopt solutions based on locally-deployed lightweight LLMs. So far, most of these commercial products are just simple integration of the chat interfaces of LLMs into the personal assistants. Research about deeper functional integration will be discussed in Section 4.1.

Despite exhibiting vast potential, this research direction is currently in an early exploration stage. There is still a substantial distance away from the ultimate goal of truly understanding and assisting users with intelligent agents. What's more, many issues related to efficiency, security and privacy have not been adequately addressed yet. The subsequent parts of this paper will systematically summarize and discuss the key issues in this direction.

3 Personal LLM Agents: Definition & Insights

Witnessing the great potential of LLM-based intelligent personal assistants and wide interests in both academia and industry, we take the first step to systematically discuss the opportunities, challenges and techniques related to this direction.

We define **Personal LLM Agents** as a special type of LLM-based agent that is deeply integrated with personal data, personal devices, and personal services. The main purpose of personal LLM agents is to assist end-users, helping them to reduce repetitive and cumbersome work and focus more on interesting and important affairs. Following this definition, the generic automation methods (prompting, planning, self-reflection, etc.) are similar to normal LLM-based agents. We focus on the aspects that are related to the "personal" parts, such as the management of personal data, the use of smartphone apps, deployment to resource-constrained personal devices, etc.

We envision that Personal LLM Agents will become a major software paradigm for personal devices in the LLM era. However, the software stack and ecosystem of Personal LLM Agents are still at a very early stage. Many important questions related to the system design and implementation are unclear yet.

Therefore, we attempted to address some of the questions based on insights collected from domain experts. Specifically, we invited 25 experts who are chief architects, managing directors, or senior engineers/researchers from 8 leading companies that are working on IPA-related products, including smartphone personal assistants, smart-home solutions, and intelligent cockpit systems. We talked with them casually on the topics of Personal LLM Agents and asked them several common questions, ranging from the application scenarios to the deployment challenges. Based on our discussion and collected answers, we summarize the insights into three subsections, including the key components of Personal LLM Agents, a taxonomy of intelligence levels, and expert opinions about common problems.

3.1 Key Components

Based on our discussions about the desired features of Personal LLM Agents, we first summarize the main components to support such features, as shown in Figure 4.

Undoubtedly, the core of Personal LLM Agents is a foundation model (large language model or other variants, we call it LLM for simplicity), which connects all other components. Firstly, the LLM is the basis to support different skills for serving the users, including responsive skills that directly execute tasks as users requested (such as question answering, weather checking, event scheduling, etc.) and proactive skills that offer services without explicit user commands (such as life logging, managing user attention, activity recommendation, etc.).

Secondly, to support these skills, the LLM manages various local resources, including mobile applications, sensors, and IoT devices. For example, the agent may complete weather checking by interacting with a smartphone weather app. Meanwhile, many people have mentioned the importance of Personal LLM Agents to provide personalized and context-aware services. Therefore, the LLM should maintain the information about the user, including the current user context (status, activity, location, etc.) and historic user memory (profile, logs, personality, etc.). To manipulate these resources, contexts and memories, it is also desired to use dedicated management systems like vector databases in combination with the LLM.

The combination of these key components is analogous to an operating system [61], wherein:

1. The foundation model is like the kernel in traditional operating systems. It is employed for systematic management and scheduling of various resources, thereby facilitating the functions of the agents.

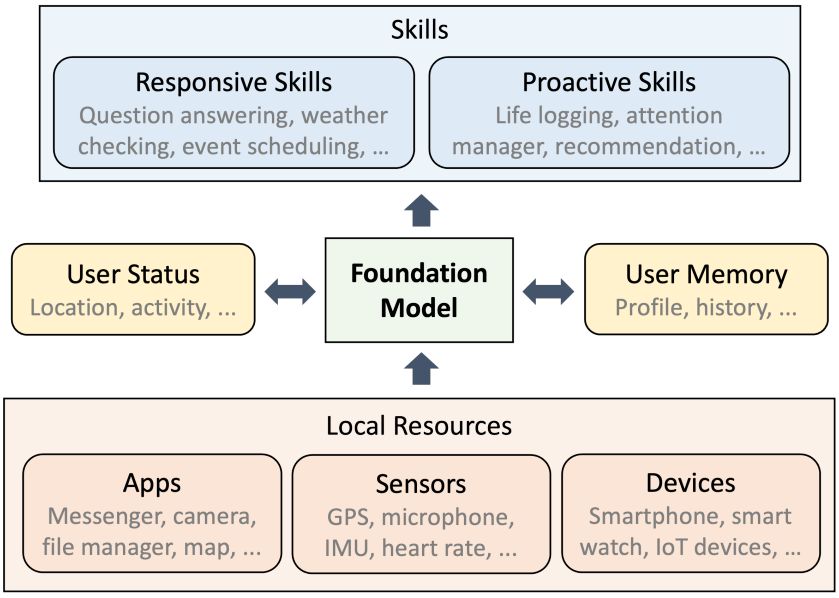

Figure 4: Main components of Personal LLM Agents.

2. The local resource layer is similar to the driver programs in traditional operating systems. In traditional OS, each driver manages a specialized set of hardware. While in Personal LLM Agents, each local resource component manages a type of tool and provides APIs for the LLM to use.
3. User context and user memory correspond to the program contexts and system logs maintained during system operations. These components form the basis for the agent to support personalized services.
4. The skills at the top layer are analogous to the software applications in traditional OS. Similar to the installation and removal of applications, the skills of agents should also be allowed to be flexibly enabled or disabled.

3.2 Intelligence Levels of Personal LLM Agents

The desired features of Personal LLM Agents require different kinds of capabilities. Inspired by the six levels of autonomous driving, we categorize the intelligence levels of Personal LLM Agents into five levels, denoted as L1 to L5, as shown in Figure 5. The key characteristics and representative use cases of each level are listed in Table 1.

At each level, the user and agent are responsible for different duties. At Level 1 (Simple Step Following), agents only take charge of step execution, and the other duties are in charge of the user. For example, when users give the command,

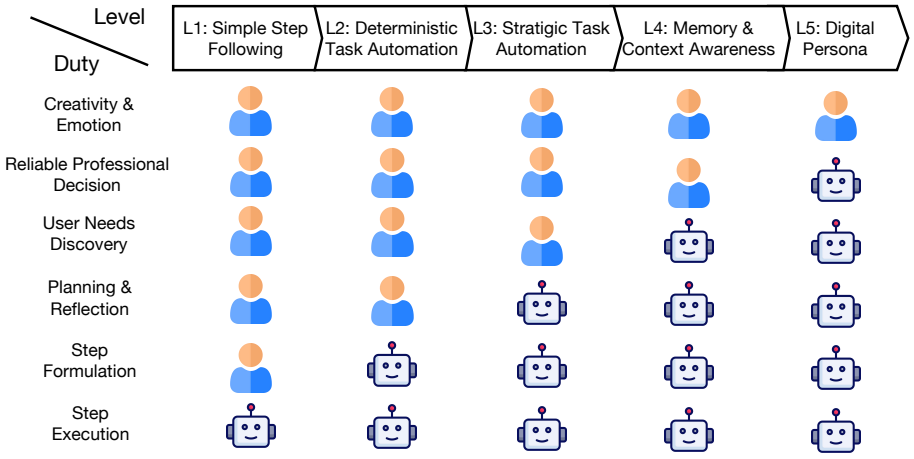

Figure 5: The duties of Personal LLM Agents at different intelligence levels.

Table 1: Different levels of intelligence for Personal LLM Agents.

Level	Key Characteristics	Representative Use Cases
L1 - Simple Step Following	Agent completes tasks by following *exact steps* predefined by the users or the developers.	- User: "Open Messenger"; Agent opens the app named Messenger. - User: "Open the first unread email in my mailbox and read its content"; Agent follows the command step by step. - User: "Call Alice"; Agent matches a developer-defined template, finds Alice's phone number in the address book, and calls the number.
L2 - Deterministic Task Automation	Based on the user's description of a deterministic task, agent *auto-completes* the necessary steps in a predefined action space.	- User: "Check the weather in Beijing today"; Agent automatically calls the weather API with parameter "Beijing" and parses info. from the response. - User: "Make a video call to Alice"; Agent automatically opens the address book, finds Alice's contact, and clicks on "video chat". - User: "Tell the robot vacuum to clean the room tonight"; Agent opens the robot vacuum app, clicks 'schedule', and sets the time to tonight.
L3 - Stratigic task Automation	Based on user-specified tasks, agents *autonomously plan* the execution steps using various resources and tools, and *iterates* the plan based on intermediate feedback until completion.	- User: "Tell Alice about my schedule for tomorrow"; Agent gathers tomorrow's schedule information from the user's calendar and chat history, then summarizes and sends them to Alice via Messenger. - User: "Find out which city is suitable for travel recently"; Agent lists several cities suitable for travel, checks the weather in each city, summarizes the information, and returns recommendations. - User: "Record my sleep quality tonight"; Agent checks every 10 minutes during sleep time if the user is using the phone, moving, or snoring (based on smartphone sensors and microphone), summarizes the information, and generates a report.
L4 - Memory and Context Awareness	Agent senses user context, understands user memory, and proactively provides *personalized* services at appropriate times.	- Agent recommends suitable financial products automatically based on User's recent income and expenses, considering User's personality and risk preference. - Agent estimates User's recent anxiety level based on the conversations and behaviors, recommends movies/music to help relax and notifies user's friends or doctors depending on the severity. - When a user falls in the bathroom, the Agent detects the event and decides whether to ask the user, notify the user's family members, or call for help based on the user's age and physical conditions.
L5 - Digital Persona	Agent *fully represents* the user in completing complex affairs, can interact on behalf of user with other users or agents, ensuring *safety* and *reliability*.	- Agent automatically reads emails and messages on behalf of User, replies to questions without user intervention, and summarizes them into an abstract. - Agent attends the work discussion meeting on behalf of the user, expresses opinions based on user's work log, listens to suggestions, and writes the minutes. - Agent records User's daily diet and activities, privately researches or ask experts on any anomalies, and makes health improvement suggestions.

agents follow explicit steps defined by the developer or given by the user to complete the task. The L1 agents do not have any ability of sensing or planning. Most template-based IPA products belong to this category.

As the intelligence level increases, the agents gradually take on more duties. At level 2, the supported tasks are still deterministic (*i.e.,* involving a fixed sequence of actions to complete), but the detailed steps to execute each task are no longer given explicitly. The agents have to auto-complete the necessary steps based on the user's task description. For instance, given a user query "`How is the weather of Beijing today`", the agent calls the weather API with `Beijing`" as a parameter and retrieves weather information from the response. Unlike the deterministic tasks at level 2, agents at level 3 can complete more complicated tasks that require strategic planning and self-reflection. For instance, the command "`Tell Alice about my schedule for tomorrow`" needs the agent to determine how to gather the schedule information (*e.g.,* using the user's calendar and chat history) and how to inform Alice about the information (*e.g.,* summarizing the calendar events and sending via the messenger app). In these tasks, agents autonomously and iteratively generate and perform the execution plan based on intermediate feedback until completing the tasks.

The agents in L1-L3 work passively driven by the users' commands, while agents at level 4 can understand users' historical data, sense the current situation, and proactively offer personalized services at appropriate times.

With ultra intelligence at level 5, agents play the role of a Digital Persona that can fully represent the user in completing complex affairs, thus users only need to focus on creativity and emotion. Agents not only sense the current status, but also predict the users' future activities and take actions to facilitate them. Beyond directly serving users, Digital Persona can also collaborate with other agents to alleviate the burden of their users' communication. Moreover, the level-5 agents should be able to continuously improve themselves through self-evolution.

3.3 Opinions on Common Problems

Next, we report the aggregrated results of the experts' opinions towards several common questions. The questions include the design choices and the potential challenges to deploy Personal LLM Agents, as summarized in Table 2.

We analyze the answers to the questions and summarize the following main takeaways.

Table 2: The common questions that we asked the domain experts. In Questions 1 to 6, we gave several common options for the experts to select/prioritize, while the experts were also allowed to give free-form answers. In Questions 7 and 8, the experts were asked to answer with text.

ID	Question
1	If the LLM is applied to personal intelligent agents, do you think it should be deployed locally or remotely?
2	How do you think customized models tailored for different users or organizations should be implemented?
3	For the LLM deployed on personal devices, which modality(ies) do you think needs to be supported?
4	What do you think is the most important capability of LLMs for personal LLM agents?
5	Considering the industry you are in, which ways of interaction do you think are the most promising for personal LLM agents?
6	In the future development of personal LLM agents, which aspect is the most crucial?
7	What features do you hope a future personal LLM agent can provide for you or your customers?
8	When integrating LLM with personal devices, what challenges do you think will be faced? What are the most urgent technical issues that needs to be addressed?

Opinion 1 (where to deploy the LLM): *Edge-cloud (local-remote) collaborated deployment of LLM is preferred, while existing cloud-only (remote-only) (e.g.,, ChatGPT) is not a widely acceptable solution.* As shown in Figure 6, 88% of participants prefer an edge-cloud collaborated architecture, 58.33% of them support local deployment, and 81.82% of them are not satisfied with the existing cloud-only solutions. Their main concerns are 1) the high latency of remote LLM service, 2) the privacy issue of transmitting personal data to the cloud, and 3) the huge cost of cloud-based LLM services.

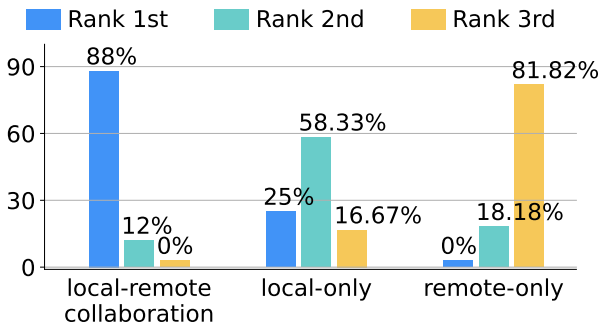

Figure 6: The vote distribution of different LLM deployment strategies in Personal LLM Agents.

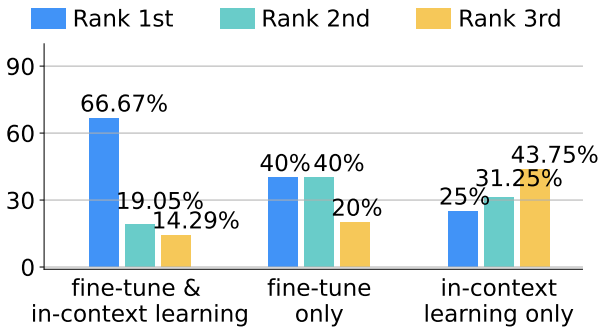

Figure 7: The vote distribution of different model customization methods for Personal LLM Agents.

Opinion 2 (how to customize the agents): *Combining fine-tuning and in-context learning is the most acceptable way to achieve customization.* In Personal LLM Agents, customizing the agent for different users and scenarios is considered necessary. Figure 7 shows that 66.67% of participants support combining the advantages of both fine-tuning and in-context learning to reach personalization (L4 intelligence). 43.75% of them do not believe L4 can be achieved by in-context learning; one possible reason is our participants are from the industry, thus they are more focused on the LLM for specific vertical domains where in-context learning hasn't received much attention.

In questions 3-5, we ask participants to rank the options and the following tables (Table 3-5) summarize their ranks. Rank 1st-4th denotes the rankness of these options voted by the participants; for example, 72% in Table 3 means that 72% participants rank Text as their first preferred modality. The "score" in each table is calculated based on the Borda Count [62], where each candidate receives points equal to the average of the number of candidates they outrank in each

ballot, with the lowest-ranked getting 2 and the highest $n + 1$ points, where n is the total number of candidates. For instance, 4.56 in Table 3 equals to $5 \times 72\% + 4 \times 20\% + 3 \times 0 + 2 \times 8\%$.

Opinion 3 (what modalities to use): *The multi-modal LLM, especially Textual and Visual modalities, is desired for Personal LLM Agents.* In our statistical result, Text is the most preferred modality just as the most popular LLMs used (*e.g.,* GPT series and LLaMA series). The second-ranked Image option and the Video modality which is specifically mentioned by 20% of the participants show that the visual modality plays a promising role in the future of personal LLM agents.

Table 3: The favored modalities to be used in Personal LLM Agents.

Options	Scores	Rank 1st	Rank 2nd	Rank 3rd	Rank 4th
Text	4.56	72%	20%	0%	8%
Image	3.64	4%	64%	24%	4%
Voice	3.18	16%	4%	60%	20%
Sensors	2.18	9.52%	14.29%	9.52%	66.67%

Opinion 4 (which LLM ability is the most crucial for IPA products): *Language understanding is considered the most important capability of LLMs, whereas the ability to handle long contexts is regarded as the most unimportant one.* On the contrary, in academia, the capability to handle long context is regarded as very important and is extensively studied. This different opinion originates from the specific vertical-domain LLMs our participants supposed and the general-purpose LLMs of academic researchers. In vertical-domain LLMs, the queries and tasks from users are not very diverse, hence the capacity of long context is not that critical.

Table 4: The importance ranking of LLM abilities for IPA products.

Options	Scores	Rank 1st	Rank 2nd	Rank 3rd	Rank 4th
Language understanding	4.52	83.33%	8.33%	4.17%	4.17%
In-context learning	3.16	4.55%	50%	45.45%	0%
Common sense reasoning	3	8.33%	33.33%	29.17%	20.83%
Long context	1.8	5.56%	11.11%	16.67%	61.11%

Opinion 5 (how to interact with the agents): *Voice-based interaction is the most popular way.* Unsurprisingly, just like the existing virtual assistant Siri, mimicking the human communication method – voice interaction is the most common and efficient choice. Text-based chatbots and GUI rank second and third since most of the participating experts focus on mobile devices, *e.g.,* smartphones. Virtual reality only obtains a 1.52 score which is the lowest across all questions; this may stem from the high price of VR devices and the unsatisfied user experience of current VR techniques.

Table 5: The favored interaction method of Personal LLM Agents.

Options	Scores	Rank 1st	Rank 2nd	Rank 3rd	Rank 4th
Voice interaction	4.04	60.87%	17.39%	21.74%	0%
Text chatbox	3.32	22.73%	45.45%	18.18%	13.64%
GUI	3.24	23.81%	38.1%	38.1%	0%
Virtual reality	1.52	0%	6.25%	25%	68.75%

Opinion 6 (which agent ability is needed to develop): In the future development of Personal LLM Agents, "more intelligent and autonomous decision-making capability" is considered the most critical feature among our participants; almost half of the participants (47.83%) rank it at first place. The options "Continuous improvement of user experience and interaction methods" and "Secure handling of personal data" also received much attention, with 36.36% and 33.33% respectively, tying for the second place. Although "Integration with IoT devices" ranks last, 47.63% of participants still believe it is important as an infrastructure for Personal LLM Agents.

Opinion 7 (what features are desired for an ideal IPA): Based on the responses from the participants, we summarize the following six key features of an ideal agent:

- *Efficient Data Management and Search:* The agent acts as an external brain to remember the user's data by efficient data storage. It provides users with fast retrieval and precise search capabilities.
- *Work and Life Assistance:* The agent serves as a copilot in work when users ask for technical details. It can also perform repetitive and heavy tasks and provide document and content generation for users.
- *Personalized Services and Recommendations:* According to user habits, the agent can discover the potential needs of users and then proactively provide services for users. It can serve as a personal and family health manager, medical server, shopping comparison assistance, travel assistance, etc.
- *Autonomous Task Planning and Completion:* The agent can understand the user's intention, decompose the tasks proposed by the user and automatically perform them step by step (further in autonomous chain-of-thought functions), and help the user complete the steps that need manual with explicit instructions.
- *Emotional Support and Social Interaction:* The agent can understand and help the user adjust their emotions by chatting. It can also understand users' relationships with different people, and help them write the response draft in users' voices.
- *Digital Representative and Beyond:* The agent can represent the user to attend meetings, drive the car, go to work, and do any authorized tasks. It can truly understand the user and communicate and socialize with others in the present users themselves.

Opinion 8 (what are the most urgent technical challenges): According to the responses from the participants, the most urgent challenges and technical issues are categorized as follows:

- *Intelligence.* 1) Multimodal Support: LLMs need to understand and process different data types (*e.g.,* text, images, and videos), thus it should possess advanced data alignment and interpretation capabilities. 2) Context Understanding and Context-aware Actions: In various application scenarios, LLMs must accurately understand user requirements and generate corresponding control instructions. This needs LLMs' context understanding ability and the ability to convert the context to effective actions. 3) Enhancing Domain-specific Abilities of Lightweight LLM: LLMs on resource-limited personal devices might underperform in complex tasks or understanding deep contextual meanings due to their size and complexity constraints. Therefore, how to boost the lightweight models' capabilities and handle complex tasks in specific domains is widely concerned.
- *Performance.* 1) Effective LLM Compression or Compact Architecture: Running LLMs on resource-limited mobile devices needs to balance the performance and quality of task completion. Efficient model compression techniques that concern the characteristics of LLMs to keep high quality of task completion are desirable. 2) Practical Local-Remote Collaborative Architecture: Local-remote collaborative architecture of LLM is considered promising, which is desired to inherit both the fast/low-cost response ability of local model and the high-quality generation ability of the cloud model. However, how to achieve accurate and efficient collaboration is widely considered as an important challenge.
- *Security & Privacy.* 1) Data Security and Privacy Protection: Ensuring the security of personal data and the protection of user privacy is critical when using personal data to train and execute LLMs. This proposes an urgent requirement to develop new data anonymization techniques and privacy protection protocols. 2) Inference Accuracy and Harmlessness: Ensure that the model outputs are precise and harmless for users, especially when used for decision-making or in sensitive scenarios.
- *Personalization & Storage.* Personalization requires efficient data storage solutions to manage and leverage user-related data, including their preferences, historical behaviors, and interactions.
- *Traditional OS Support.* For mobile-based LLM agents, a critical requirement is LLM-friendly interfaces and support of traditional operating systems like Android. This may involve updates at the operating system level and the development of application programming interfaces (APIs) for better integration and utilization of LLM's functionalities.

Motivated by the valuable opinions of domain experts, the following sections will discuss the desired capabilities and potential challenges in more detail.

4 Fundamental Capabilities

We first discuss the capabilities required by Personal LLM Agents to support diverse features. Excluding the general capabilities of normal LLM agents, we focus on three fundamental capabilities for personal assistants, including task execution, context sensing, and memorization. Task execution (§4.1) is to translate the users' commands or the

proactively perceived tasks into actions on personal resources. The purpose of context sensing (§4.2) is to perceive the current state of the user and the environment, providing comprehensive information for task execution. Memorization (§4.3) is to record the user data, enabling the agent to recall past events, summarize knowledge and self-evolve. While context sensing and memorization are abilities associated with querying information from users, task execution refers to the ability of providing services to users. Figure 8 depicts the relation of these fundamental capabilities. The following sections discuss these capabilities in details.

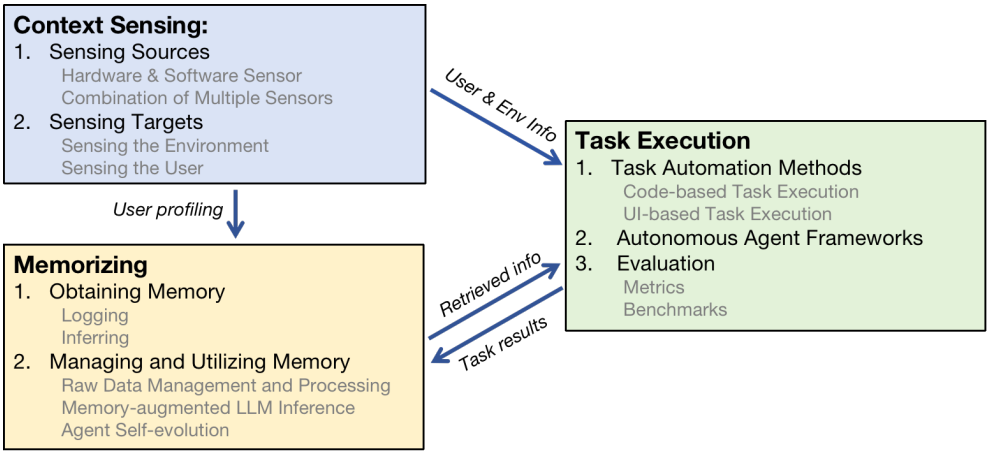

Figure 8: The fundamental capabilities of Personal LLM Agents.

4.1 Task Execution

Task execution is a fundamental capability of a Personal LLM Agent, enabling it to respond to user requests and carry out specified tasks. In our scenario, the agent is designed to interact with and control various personal devices such as smartphones, computers and IoT devices to automatically execute users' commands.

A fundamental requirement for task execution is the agent's ability to accurately interpret tasks as communicated by users. Typically, tasks may originate from users' verbal or written instructions, from which the intelligent agent discerns the user's intent. With the maturation of voice recognition technology, converting voice information into text has become highly convenient [63, 64].

Personal LLM Agents should make plans and take actions automatically after converting the users' commands into text. While planning poses a challenge for traditional DNNs, LLM-based agents exhibit greater proficiency in this regard. The planning and reasoning abilities of LLM agents have been discussed in the former surveys [65, 66, 67]. Our paper primarily focuses on the manipulation of personal data and interaction with personal devices. A significant consideration is that Personal LLM Agents might need to interact with applications or systems that may lack comprehensive API support. Consequently, we also explore the user interface (UI) as an important tool for personal agents, enabling effective interaction in scenarios where API limitations exist.

4.1.1 Task Automation Methods

Based on the types of interaction mode, the methods of task execution can be categorized into code-based and UI-based approaches. In the code-based scenario, agents primarily complete tasks by automatically generating code to call APIs. Under UI-based scenarios, agents interact with personal devices by automatically simulating human interactions with the UI interface.

Code-based Task Automation often involves generating appropriate code to interact with APIs, databases, and DNN models. Traditional code-based personal assistants are often based on slot-filling-based task-oriented dialogue (TOD) frameworks. In the era of LLM, more researchers are attempting to directly use LLMs to directly generate code that calls APIs in order to accomplish more complex tasks.

- **Slot-filling method** is often used in task-oriented dialogue systems (TOD) or chatbots, which is conversational AI designed to assist users in completing specific tasks through dialogue [68, 69]. In a task-oriented dialogue system, "slots" are predefined categories of information necessary to complete a task. For example, in a travel booking application, slots might include destination, travel dates, number of passengers, etc. During

a conversation, the system prompts the user for this information, and calls corresponding APIs to complete the tasks. For mobile devices, many approaches focus on facilitating task automation by allowing users to demonstrate the desired tasks, which can be executed via a conversational interface [70, 71, 24, 25]. These methods often assume that the user's tasks can be defined as a collection of slot-value pairs. This assumption allows for precise management of the conversation with the controllable units, and to execute the task is to keep prompting users for the values of slots that have not been identified. However, these methods do not consider complex cases where there are multiple values for a slot or relationships between slots [72]. Besides, they heavily rely on well-defined APIs and lack adaptability to unseen domains. Recent research papers utilize the understanding and reasoning ability of LLMs to complete more complex and multi-turn TOD tasks [73, 74, 75, 76], and improve the efficiency of Slot-filling methods.

- **Program synthesis method** is to utilize the code generation ability of LLMs to interact with APIs. One way is to fine-tune LLMs to use specific APIs. WebGPT [46] fine-tunes a GPT-3 [77] to answer long-form questions by calling Microsoft Bing Web Search API [78]. Some recent works [45, 79, 80, 81] fine-tune LLMs to retrieve and call APIs, enhancing their performance in various tasks like mathematical reasoning and program synthesis. Another way is to utilize the chain reasoning [82, 83, 67] and in-context learning ability [77] of LLMs. They show descriptions and demonstrations of the tools (e.g. APIs, other DNNs, etc.) in context and ask LLMs how to use them to complete tasks [84, 85, 86, 51, 87]. However, fine-tuning LLMs can be costly and restricted to the predefined set of tools, and in-context learning may fail when the number of APIs go large. Thus, authors of ToolkenGPT [88] attempt to solve this problem by representing each tool (API) as a token.

Code-based methods can complete thousands of tasks from web searching to image generating. However, not all the needed APIs are available for agent developers in real-life apps out of security concerns or business interests. Besides, there are tasks that can be executed easily for human users but are difficult for calling system APIs [72]. Depending solely on publicly available APIs may not fully meet the highly diverse requirements for mobile task automation.

UI-based Task Automation. Autonomous UI agents attempt to translate users' tasks into UI actions on smartphones or other personal devices, automating these tasks through direct UI interaction. Compared to code-based task execution, autonomous UI agents do not rely on publicly available APIs, potentially allowing for more versatile automation capabilities. However, executing users' tasks by UI actions is not easy for traditional DNN models because of the implicit relations between tasks and UI elements. Recently, researchers utilize the comprehension and reasoning abilities of LLMs to improve the performance of autonomous UI agents.

The input of the UI agent is a task described in natural language, and a representation of the current UI, and the output is the UI action to be executed on the UI. Depending on how they represent the UI, we can categorize the autonomous UI agents into text-based GUI representation and multimodal GUI representation.

- **Text-based GUI representation** is to convert the UIs into pure text. Seq2act [4] uses a transformer-based model [31] to ground users' instruction to UI actions described in <operation, object, argument> tuples. Researchers also investigate prompting with mobile UIs to complete tasks of UI instruction mapping [89]. The authors convert mobile UI into HTML code, which is easy for LLMs to understand because an important part of their training data is scraped from Github. DroidBot-GPT [90] is an LLM-based system to complete users' tasks in a sequence of UI actions. Mind2Web [91] filters the raw HTML of webpages with a smaller LM and uses the LLM to select the target element and action. AutoDroid [92] uses app analysis tools to acquire app domain-specific knowledge and uses it to augment the LLMs for task automation. In AXNav [93], authors build a system using LLMs and pixel-based UI Understanding to execute manual accessibility tests. MemoDroid [94] introduces an LLM-based mobile task automator that can break tasks into smaller sub-tasks and complete them by recalling former actions.
- **Multimodal representation** is to use the image (and text) description of UI as the input of the Personal LLM Agents. META-GUI [95] proposes a UI-based method for task-oriented dialogue (TOD) systems. ResponsibleTA [96] proposes a fundamental multi-modal framework that equips LLMs with the ability to predict feasibility, verify completeness, and ensure security when autonomously completing users' tasks. Auto-UI [97] proposes a Multi-modal framework that interacts directly with the interface and introduces a chain-of-action technique to aid the agent in making decisions. RUIG [98] builds a multimodal model to map instructions to UI screenshots, and MM-Navigator [99] uses GPT4-V for UI navigation tasks. UINav [37] introduces a lightweight task automation system based on task demonstrations, and the input modality can dynamically switch between raw pixels view hierarchy. CogAgent [100] introduces a visual language model with 18 billion parameters to complete tasks such as UI understanding and navigation.

While UI-based task automation has the potential to achieve a more flexible personal agent framework compared to API-based automation, its research is still in the early stages. It remains challenging to accomplish more complex user

commands. Besides, the privacy and security issues have not been fully addressed [92, 96]. It also remains controversial about the UI representation. While multimodal representation can handle elements that cannot be parsed through accessibility services, it is plagued by the heavy demands of screen recording and the limited reasoning abilities of current vision language models [101].

4.1.2 Autonomous Agent Frameworks

An LLM-powered autonomous agent is composed of an LLM brain to make plans and self-reflection, a memory to store past information and knowledge, and a tool usage module to interact with tools (e.g. APIs, UIs, programming languages) [102, 66]. There are a lot of popular projects that provide frameworks for users to create LLM-powered agents [103, 104, 105, 106, 107, 108, 109, 110, 111]. They attempt to enhance the ability of LLMs by interacting with other external tools and retrieving long/short-term memory. Auto-GPT [103] is one of the most famous frameworks, which can execute users' commands by generating prompts for GPT and using external tools. LangChain [104] is another popular framework that helps developers to create more sophisticated and context-aware applications using LLMs. Due to the ability to understand and produce natural language, LLM-powered agents can also engage with one another effortlessly, fostering an environment where collaboration and competition among multiple agents can thrive [112, 113, 108, 114]. These autonomous agent frameworks make significant engineering contributions, providing a more user-friendly framework for the LLM-powered applications.

For mobile devices, AutoDroid [92] provides an effective framework for developing mobile agents. Developers can easily create an automator for mobile tasks by either exploring apps using a test input generator or through manual demonstration. AutoDroid then automatically analyzes these records and utilizes them to improve Language Learning Models (LLMs) for more efficient task automation. Huang et al. [115] develop a new method to effectively extract macros (basic units of user activity in apps such as "login", or "call a contact") from user-smartphone interaction traces. These macros can help agents to automatically complete tasks.

4.1.3 Evaluation

Evaluating the performance of task execution is a challenging issue. For API-based task execution, former surveys have provided a comprehensive summary on how to evaluate them [65, 67]. Our paper mainly focuses on the evaluation of UI-based task automation.

Metrics: The metrics of UI-based task execution are completion rate [4, 95, 92] and manually designed reward [116, 117]. The completion rate is the probability that all actions predicted by the model are entirely consistent with the ground truth. However, since there may be different methods to complete a task, and the ground truth typically represents only one of these methods, the accuracy evaluated by this approach is not entirely correct [92]. Manually designing rewards based on the crucial steps can be more precise [117], but they are less scalable because of the complex annotating process.

Table 6: UI task automation benchmarks. The structured UI form are view hierarchy (VH) and document object model (DOM) for Android and web respectively. For Windows, the metadata stems from the textual metadata within the operating system.

Benchmark	Name	Platforms	Human annotations	UI format	High-level tasks	Exploration memory
Datasets	PhraseNode [118]	Web	51,663	DOM, Screen	✗	✗
	UIBert [34]	Web	16,660	DOM, Screen	✗	✗
	RicoSCA [4]	Android	N/A	VH, Screen	✗	✗
	PixelHelp [4]	Android	187	VH, Screen	✓	✗
	MoTiF [119]	Android	6,100	VH, Screen	✓	✗
	META-GUI [95]	Android	4,684	VH, Screen	✓	✗
	UGIF [120]	Android	523	VH, Screen	✓	✗
	Mind2Web [91]	Web	2,350	DOM, Screen	✓	✗
	AITW [121]	Android+Web	715,142	Screen	✓	✗
	DroidTask [92]	Android	158	VH, Screen	✓	✓
Platforms	MninWoB++ [38, 6]	Web	17,971	DOM, Screen	✗	✓
	WebShop [122]	Web	12,087	DOM, Screen	✓	✓
	WebArena [123]	Web	812	DOM, Screen	✓	✓
	AndroidEnv [116]	Android	N/A	Screen	✓	✓
	MobileEnv [117]	Android	N/A	VH, Screen	✓	✓
	AssistGUI [124]	Windows	100	Metadata, Screen	✓	✓

Benchmarks: Table 6 lists the benchmarks of UI-based task automation. One group of benchmarks is static datasets, which often include a set of human-annotated tasks, structured UI data (and screenshots), and actions to complete the tasks. Some of the tasks are synthetically generated [4, 116, 117]. The early works mainly focus on low-level tasks with clear instructions [118, 34], for example, *click the 'settings' button, and then click 'Font size'*. Later works introduce high-level tasks that could be completed in multiple steps [4, 119, 95, 120, 91, 121], for example, *delete all the events in my calendar*. Another group of benchmarks are platforms that enable the agent to interact with. MiniWoB++ [38, 6], WebShop [122], and WebArena [123] provide web environments where agents can navigate and operate on the web by clicking, typing, closing page, and so on. AndroidEnv [116] and MobileEnv [117] provide a dynamic environment where agents can engage with any Android-based application and the core operating system. This framework allows for a wide scope of interaction and task-solving capabilities within the diverse Android platform.

Remark. Existing approaches have demonstrated the remarkable ability of LLM agents in task reasoning and planning. However, there are several important problems to solve to realize practical Personal LLM Agents.

1. How to accurately and efficiently assess the performance of agents in real-world scenarios. Because there are usually various ways to accomplish the same task, it is inaccurate to use a static dataset to measure the accuracy of task execution. Meanwhile, dynamically testing the tasks in a simulated environment may be inefficient and hard to reproduce.
2. How to robustly determine if a task has been completed. LLMs often experience hallucinations during task execution, making it difficult to determine whether the current task has been completed.
3. Regarding UI agents, what is the best way to represent the software UI? The vision-based representation (e.g. screenshot) is generally available, while the text-based representation is usually more lightweight and friendly for LLM agents to operate.

4.2 Context Sensing

Context Sensing refers to the process that the agent senses the status of the user or the environment, in order to provide more customized services. In this work, we adopt a broad definition of context sensing, by considering generic information gathering process as a form of sensing. Hardware-based sensing aligns with the conventional notion of sensing, primarily involving data acquisition through various sensors, wearable devices, edge devices, and other data sources. On the other hand, software-based sensing emphasizes diverse means of data acquisition. For example, analyzing user typing habits and common phrases constitutes a form of software-base sensing.

In Personal LLM Agents, context sensing capability serves various purposes. **1. Enabling Sensing Tasks**: Some tasks inherently require the agent to do sensing. For instance, when a user requires the agent to detect snoring during sleep, the agent must possess the ability to actively acquire, process, and analyze audio data. **2. Supplementing Contextual Information**: The sensed information can facilitate the execution of ambiguous or complex tasks. For example, when the user wants to listen some music, it's good to know the current activity of the user to recommend appropriate music. **3. Triggering Context-aware Services**: The sensing capability is also the basis to provide proactive services. For example, the agent may notice the users to keep focus upon detecting dangerous driving behaviors. **4. Augmenting Agent Memory**: Some information perceived through sensing can become a part of the agent memory, which can be used by the agent for further customization and self-evolution.

We introduce the techniques of context sensing from two perspectives, including sensing sources and sensing targets.

4.2.1 Sensing Sources

Hardware Sensor. Modern personal devices are equipped with a wide range of built-in hardware sensors, including accelerometers, gyroscopes, magnetic field sensors, light sensors, thermometers [125], microphones [126], GPS modules, cameras [127], etc. Some other modules such as bluetooth and Wi-Fi [128] can also be used for sensing purposes. With the growing prevalence of wearable and IoT devices such as smart watches, bluetooth headphones [129], and smart home devices [130], the sensing scope and sensing modalities are greatly expanded.

Software Sensor. Unlike hardware sensing that obtains data from real sensor devices, software sensing focuses on obtaining information from existing data, such as app usage [131], call records [132], and typing habits [133], etc. In reality, the realm of software sensing is incredibly broad. For instance, in the field of natural language processing or audio, there exists a plethora of sensing research based on text or speech. Furthermore, recommendation systems such as e-commerce or short video platforms, the process typically involves first sensing certain user information and subsequently recommending specific products or content. These sensors let agents better understand the users, enabling them to provide with more intelligent and personalized services.

Combination of Multiple Sensors. Multi-sensor collaborative sensing stands out as an effective method for enhancing perceptual capabilities. Previous endeavors have demonstrated the assessment of user emotions, stress levels, and emotional states based on touchscreen and inertial sensors [134], identification of time spent through screen capture and sensor data [135], breath detection through headphone microphones [136], and nuanced motion detection through sensors and audio [137].

The significance of multi-sensor collaboration extends to the proliferation of intelligent wearables and smart homes. For instance, automatic recognition of when a user is working or resting using data collected from personal devices [138] (smartwatches, laptops, and smartphones), or action detection through the combination of headphones and smartphone microphones [129]. Furthermore, technologies involving the fusion of household appliances, such as user action perception based on existing wired devices [139], motion recognition in smart home environments [130], Wi-Fi-based motion detection [140], multiperson detection [128], and sleep monitoring [141].

Multi-sensor and multi-device scenarios necessitate intricate considerations in data source selection, data fusion, and data analysis methods. Existing methodologies include LLM-driven strategies for generating multi-sensor policies in human behavior understanding [142], emotion-agnostic multi-sensor data multitask learning frameworks [143], cross-modal fusion of sensing data [144], wearable device motion recognition with a focus on multi-sensor fusion [145], and predictive anxiety in sensor data under conditions of data absence [146]. Furthermore, there are studies that analyze the importance of data features in fall detection [147].

With the evolution of sensing technologies, multi-sensor and multi-device collaborative sensing has become a staple approach for perceiving complex scenarios. Effectively integrating diverse data sources to maximize accuracy and determining methods to eliminate less crucial data from a multitude of sources to conserve resources are vital research areas.

4.2.2 Sensing Targets

The objectives of context sensing can be categorized into environment sensing and user sensing. Environment sensing encompasses factors such as location, occasion, religious and cultural backgrounds, national and societal contexts, and more. Meanwhile, user sensing incorporates elements such as user activities, states, personal information, personality traits, emotions, goals, physical conditions, and other related aspects.

Sensing the Environment. We further categorize environment sensing into two dimensions: *scene sensing* and *occasion sensing*. *Scene sensing* predominantly involves more tangible environmental factors, such as locations and places. *Occasion sensing* delves into deeper environmental information, including religious and cultural backgrounds, national differences, and social relationships.

- **Scene sensing** is often readily perceptible but hold significant importance, leading to variations both in behavior and emphasis. For behavior instance, detecting a user in a library prompts the agent to adjust the phone to silent mode, while in a bar increasing the volume and activating vibration may be necessary. Similarly to emphasis, when a user is in a meeting room, the agent should focus more on tasks related to meeting content recording and work organization, whereas in a gym, emphasis should shift towards fitness plans and heart rate analysis. Previous work in scene awareness has employed various techniques, such as location-based approaches [148], audio or video analysis [149, 150], and sensor capabilities analyzing aspects like airflow through smartphone microphones to assess ventilation [126], or scene recognition achieved by analyzing macro photographs taken with the smartphone camera when placed near a surface [127].
- **Occasion perception** is more elusive in perception, and their impacts are relatively discreet. Earlier studies have identified differences in behavior and emotion recognition tasks across countries [151] and regions [152]. The national, ethnic, religious, and cultural backgrounds implied by the current user and setting are crucial. Perceiving others and objects in the current environment is equally vital. For example, previous work detected social scenarios based on sensor data, analyzing the behavior of socially anxious individuals in different social settings [153]. Other research delved into analyzing drinking-related social scenes using multiple sensors, even predicting the size and gender composition of drinking groups [154]. Additionally, studies explored the relations between sensor data, dietary habits, and social settings, revealing a strong association between binge eating and social environments, making it predictable [155].

With the recent advancements in LLMs, there have also emerged some environment sensing methods. For example, forecasting pedestrian flow through the analysis of public events [156] and LLM-based environmental understanding using multiple sensors [157].

Environment sensing is crucial context information for a personal agent. Different environments lead to distinct behaviors and focal points, extending beyond mere locations to encompass social occasions, cultural backgrounds, and

deeper conceptual elements, all environment individuals and relationships, interactions, and anticipating the impacts on both the environment and the user. These considerations directly influence the level of intelligence exhibited by the personal agent.

Sensing the User. User awareness is one of the primary features of Personal LLM Agents. A deeper understanding of the user can better reflects the value and significance of the Personal LLM Agents. We categorize user sensing into two temporal dimensions, including *short-term* and *long-term*. *Short-term* sensing exhibits higher temporal variability and increased randomness. On the other hand, *long-term* sensing necessitates extended maintenance and correction, making it relatively more stable and reliable.

- **Short-term user sensing** encompasses various aspects, including users' routine actions [158], or specialized activities such as toothbrushing effectiveness [159], user states such as working or resting [138, 135], user health conditions [160, 125, 161], as well as user emotions [162, 134] and stress levels [163]. Short-term sensing typically involve rapidly changing and shallow-level state information. Efficiently capturing such information can significantly enhance the context awareness of Personal LLM Agents.
- **Long-term user sensing** mainly focus on the analysis of users' profile and personality. Vairous approaches have been proposed to understand users' work, study, and daily life. For instance, a study utilized sensor data from new smartphones to detect the prolonged psychological states of freshmen [164]. Another study demonstrated the capability to predict learning performance and social activities based on perception data [165]. Gao et al. [166] delve into the techniques to predict personality based on the intensity of physical activities. There is also research examining the relationship between sensor data and user career advancement [167], as well as a study that predicts user life satisfaction [168]. Furthermore, specific states of users have been a focus, including studies on the perception of mental illnesses [169], such as one that predicts and analyzes schizophrenia [170], and another that detects habits like smoking [171]. Long-term sensing involve deep and abstract information, containing the profound logic behind user behavior. These pieces of information are often more subtle, making perception and maintenance challenging. However, they constitute an essential aspect for advanced personal agents.

In the realm of user sensing, there are also several LLM-based initiatives, such as employing LLM for recommendation tasks [172, 173], sentiment analysis with LLM [174], and the development of a personal doctor equipped with inquiry and perception capabilities [175].

Remark. Existing methods often confine themselves to specific sensors, individual apps, or particular domains. In Personal LLM Agents, a possible opportunity is to unify all sensing results concerning the environment and the user to originate from diverse sources. However, to achieve this goal involves several important research challenges.

1. What is a unified format or ontology of the sensed information? The agents should be able to convert diverse sensing data into this format and conveniently use the data for various downstream tasks.
2. Given the broad scope of sensing, how can the agents decide when and what to sense, in order to provide context-aware services with minimal overhead?

4.3 Memorizing

Memorizing denotes the capability to record, manage and utilize historical data in Personal LLM Agents. This capability enables the agents to keep track of the user, learn from past experiences, extract useful knowledge, and apply this acquired knowledge to further enhance the service quality. The related work is mainly aimed to answer two questions, including how to obtain the memory and how to utilize the memory.

4.3.1 Obtaining Memory

The agent memory can be in various formats. For example, the basic user profiles (*e.g.,* birthdate, addresses, personalities, preferences) are often stored in key-value pairs, allowing for easy key-based retrieval. Historical records are usually represented as sequences indexed by timestamps, which archive user service access, activities, system events and so on over the time. The user's documents, photos, videos, etc. are stored as files, which are often produced by other applications. There are mainly two ways to obtain the memory: directly logging the raw data or indirectly inferring knowledge from raw data.

Logging. The most straightforward way to obtain memory is through logging, such as recording user input, system events, and sensed contexts. Logging data is often relatively simple. *Life logging* is a commonly-discussed topic that

focuses on tracking and recording user data created through the activities and behaviors of users, contributing to a comprehensive understanding of individuals' lifestyles and preferences [176, 177]. Data recorded at specific moments using video cameras provide deeper overview of daily activities [178]. Moreover, recording data over long periods of time can provide valuable insights into behavior patterns, which will support the personalization of intelligent agents [179].

Inferring. Another way of Personal LLM Agents to obtain memory is to extract knowledge from the raw data. With the advancements in machine learning and data analytics, it has become possible to infer user behavior, patterns, and interactions to gain insights into their psychology, preferences, and other high-level information. For example, user personality can be extracted from texts [180, 181], emotions can be read from image and text data [182, 183], preferences can be modeled from historical interaction information [184], and knowledge graphs can be extracted from smartphone push notifications [185]. These extracted high-level information will also be stored as memories of the agent and utilized in services.

4.3.2 Managing and Utilizing Memory

After obtaining the memory, the next question is how to manage and utilize the memory to provide better services in Personal LLM Agents. Based on the purposes of utilizing memory, we divide the relevant techniques into following three parts, including raw data management, memory-augmented LLM inference, and agent self-evolution.

Raw Data Management and Processing. A basic ability of Personal LLM Agents is to access and process the raw memory data (*e.g.,* selecting, filtering, transforming to other formats, etc.), in order to facilitate other advanced functions. This line of work primarily focus on enabling more natural and human-comprehensible access, manipulation, and modification of data. Since the input-output and reasoning processes of LLMs are based on natural language, such interfaces are more easily integrated with other capabilities of large models. In this research area, numerous endeavors have explored the use of machine learning models or template-based methods to map user data requests to database SQL statements [186, 187]. There are also framework-level works examining how to unify and simplify data interfaces. For instance, PrivacyStreams [188] unifies all personal data access and processing interfaces into a stream-based framework, which is more conducive for large language models to comprehend and manage.

Memory-augmented LLM Inference. To enable the Personal LLM Agents to provide customized services based on the user-related memory, it is usually desired to make use of the memory data in the LLM inference process. Recent research in LLM agents has explored leveraging memory to enhance decision-making and reasoning [83, 189, 190, 191, 192], which provides inspiration for a solution where Personal LLM Agents can offer personalized services to users through memories. The techniques can be different based on the types of the memory.

- **Short-term memory** preserves and retains pertinent information in the form of symbolic variables, ensuring its accessibility and applicability during the current decision cycle. This includes perceptual inputs, active knowledge (generated by reasoning or retrieved from memory data), and other core information carried over from the previous decision cycle (*e.g.,.*, agent's active goals). CoT [82], Scratchpads [193] encourage the LLM to generate intermediate reasoning, using the LLM's own context as a form of working memory. CoALA [194] proposes that working memory should be a persistent data structure during long-term memory (LLM) calls. Each call generates its input from a subset of working memory (e.g., a prompt template and relevant variables), and the output is subsequently parsed into other variables (e.g., an action name and arguments) which are stored back in working memory and used to execute the corresponding action. In addition, short-term memory has the capability to interact with long-term memory and other data interfaces, serving as the central hub connecting different components of a language agent [195, 196].
- **Long-term memory** stores experiences from earlier decision cycles. This can consist of history event flows [189], game trajectories from previous episodes [197, 198], interaction information between the user and the agent or other representations of the agent's experiences. During the planning stage of a decision cycle, these episodes may be retrieved into working memory to support reasoning. An agent can also write new experiences from working to episodic memory as a form of learning. Secondly, long-term memory stores an agent's knowledge about the world and itself. Traditional approaches leverage retrieval for reasoning or decision-making initialize memory from an external database for knowledge support (*e.g.,* retrieval-augmented methods in NLP [199, 200], "reading to learn" approaches in RL [201, 202]). Agents may also write new knowledge obtained from LLM reasoning and user into long-term memory as a form of learning to incrementally build up world knowledge from experience.

Agent Self-evolution. To better accommodate users, Personal LLM Agents may also need to dynamically update themselves based on the memory data. We refer to this as "self-evolution". The foundational functionality of intelligent

agents is predominantly reliant on LLM. Therefore, the key to the self-evolution of intelligent agents lies in how to leverage LLM for the discovery and exploration of new skills, as well as in the continuous update of the LLM itself.

- **Learning Skills.** Currently, numerous efforts are underway to enable LLM-based agents to engage in continuous skill learning and acquisition [203, 204]. These methods draw inspiration from the generality and interpretability of programs [205], considering skills as executable code, and optimize skill acquisition by leveraging the in-context learning ability of LLM through the strategic use of prompts. They also manage a skill repository, integrating new skills as APIs, enabling intelligent agents to continually learn and reuse these skills in subsequent tasks. Prior work has demonstrated that modern LLMs can capture relevant information about meaningful skill chains [50, 48]. Hence, intelligent agents have the capability to acquire novel skills by strategically linking skills within a foundational skill set [206]. In this process of skill chaining, the intelligent agent makes purposeful selections of subsequent meaningful skills, leveraging the a priori knowledge embedded in LLM and utilizing execution feedback to allow the language model to adjust its selections. This targeted approach enables the agent to efficiently assimilate complex skills.
- **Finetuning LLM.** To achieve the self-evolution of intelligent agents, continuous fine-tuning of the LLM is also required. There are several reasons: 1. Current LLMs were not specifically designed for agent-specific use cases, such as generating actions or self-evaluations, where limited learning support is provided by few-shot prompting. 2. Due to performance constraints on mobile devices, the capabilities of the LLM component of the intelligent agent are limited. This limitation makes it difficult for the model to acquire new skills through prior knowledge and in-context learning abilities. 3. During the operational phases of intelligent agents, the consistent emergence of materials such as the latest corpus [207], new knowledge [208], and tools [209] can frequently change the task schemas. This necessitates continual adaptation of LLMs. In such cases, fine-tuning the model becomes necessary to enhance its capacity for handling new tasks and generating appropriate actions. Research indicates that fine-tuned smaller LLMs could outperform prompted larger LLMs for specific reasoning [210, 211] and acting [195] needs, while enjoying reduced inference time and expense. Parameter efficient fine-tuning (PEFT) [212] presents a promising approach for efficiently fine-tuning LLMs. It only requires fine-tuning a small subset of external parameters [213], making it friendly for edge devices, and it can effectively alleviate the issue of catastrophic forgetting [214]. There have also been some preliminary attempts to conduct the study of LLM fine-tuning for agents [215] with trajectories from multiple tasks and prompting methods, inspiring future endeavors aimed at developing more capable and useful Personal LLM Agents.

Remark. The ability to generate and leverage the memory about the user is the basis of personalization in Personal LLM Agents. We highlight following three open problems surrounding the memory mechanism of Personal LLM Agents.

1. The agent memory can potentially be huge, heterogeneous and dynamic. What is the most effective and efficient way for the agents to organize and retrieve the memory?
2. Human has the ability to forget. Since inappropriate data in the memory can be harmful for the agents' service quality and efficiency, how can the agents determine what information to memorize?
3. What is the best way for the agents to self-evolve with the memory? Specifically, what data to use, when to evolve, and how (fine-tuning or else)? How can the personalized models accept updates of the base foundation model?

5 Efficiency

Capability \ Process	Inference	Customization	Memory Manipulation
Task Execution	Task Decomposition; Decision Making.	Task Switching.	Retrieving Historical Data.
Context Sensing	Environment Sensing; User Sensing.		Storing Sensing Results.
Memorizing		Self-evolution via Fine-tuning.	Adding and Maintaining Memory.

Figure 9: The mapping relations between the low-level processes and high-level capabilities of Personal LLM Agents.

Due to the limited hardware resource and power supply on many personal devices, it is important to improve the efficiency of Personal LLM Agents in the deployment stage. We've discussed in Section 4 the fundamental capabilities of Personal LLM Agents, including task execution, context sensing, and memorizing. These capabilities, as shown in Figure 9, are backed by more elementary processes, mainly including the inference, customization and memory retrieval of the LLM agent. Each of these processes desires careful optimization of efficiency, as described below.

Inference of LLMs is the basis of an agent's various capabilities. For example, the agent may first decompose a complex task into several steps with the help of the LLM, then solve each step through either LLM inference or invoking personal tools (*e.g.,* schedule a meeting). Sensing the context or generating the memory may also rely on the reasoning abilities of LLMs. While the cost of using the tools or sensors is usually hard to estimate due to the diversity, LLM inference is a common procedure that demands a lot of both computation and memory resources. Therefore, the LLM inference becomes the performance bottleneck for the Personal LLM Agents, requiring careful optimizations on its efficiency.

Customization is another important process of Personal LLM Agents for accommodating different user requirements. Customization is needed when the agents are installed to different users or used in different scenarios. The self-evolution of Personal LLM Agents is also a process of customization. To offer customized services, an agent can either feed the LLM with different context tokens or tune the LLM with domain-specific data. Due to the frequent needs of customization, the processes may impose considerable pressure on the system's computational and storage resources.

Memory manipulation is another costly process. To provide better services, the agents may require access to longer contexts or external memories, such as environment perceptions, user profiles, interaction histories, data files, etc. Consequently, this gives rise to two considerations. The first pertains to necessitating LLMs to handle longer inputs. The second issue centers around the management and acquisition of information from an external memory bank.

We'll dive into the efficiency of each component in the following subsections, as is shown in Figure 10.

5.1 Efficient Inference

Since the runtime cost of Personal LLM Agents is dominated by LLM inference, it is important to improve the inference efficiency to enhance the overall efficiency of the agent. Although the total inference cost can be significantly influenced by the design of agents, including how the agents send requests to LLMs, what prompts to use, etc., we will be focused on model and system-level approaches only. The reason is that the designs of agents may vary based on the actual applications and don't directly contribute to the efficiency of LLM inference itself.

Many model and system-level approaches have been proposed to improve the efficiency of LLM inference. While some of them are generic for the overall performance and efficiency (*e.g.,* model compression), there are also techniques targeting the efficiency of specific perspectives, such as model size, inference latency, memory consumption, energy consumption, etc. We will discuss these aspects separately in the following parts of this subsection.

5.1.1 Model Compression

Model compression techniques, which directly reduce the model size and computations, are generic optimizations to enhance the inference efficiency of LLMs, including computation, memory, energy and etc. The model compression techniques are further categorized into various approaches, including quantization, pruning (sparsity), distillation and low-rank factorization.

Quantization is one of the most important compression approaches for LLMs. It reduces the model size by using fewer bits to represent the model parameters, and also reduces computations with system-level support for quantized kernels. Quantization methods can be further divided into post-training quantization (PTQ) and quantization-aware training (QAT), based on whether additional training is required after quantization. Unlike QAT (*e.g.,* LLM-QAT [218]) which requires non-negligible additional training effort, PTQ is more available and flexible for on-device deployment under different hardware constraints.

Recent works have revealed that the difficulty of LLM quantization mainly lies in activations, where the outliers are hard to quantize [271, 272]. Existing works have proposed various approaches to tackle this challenge. A typical line of work adopts the weight only quantization (WOQ) paradigm, which conduct integer quantization (*e.g.,* INT4 and INT8) on weights only, while preserving activations in float formats (*e.g.,* FP16 and FP32). WOQ achieves a trade-off between the compression ratio and model perplexity. A straightforward way of WOQ is the group-wise uniform quantization implemented in current mobile deployment frameworks (*e.g.,* llama.cpp [273] and MLC-LLM [274]). Recent works also proposed different quantization algorithms to enhance model capability, such as GPTQ [216] and AWQ [217].

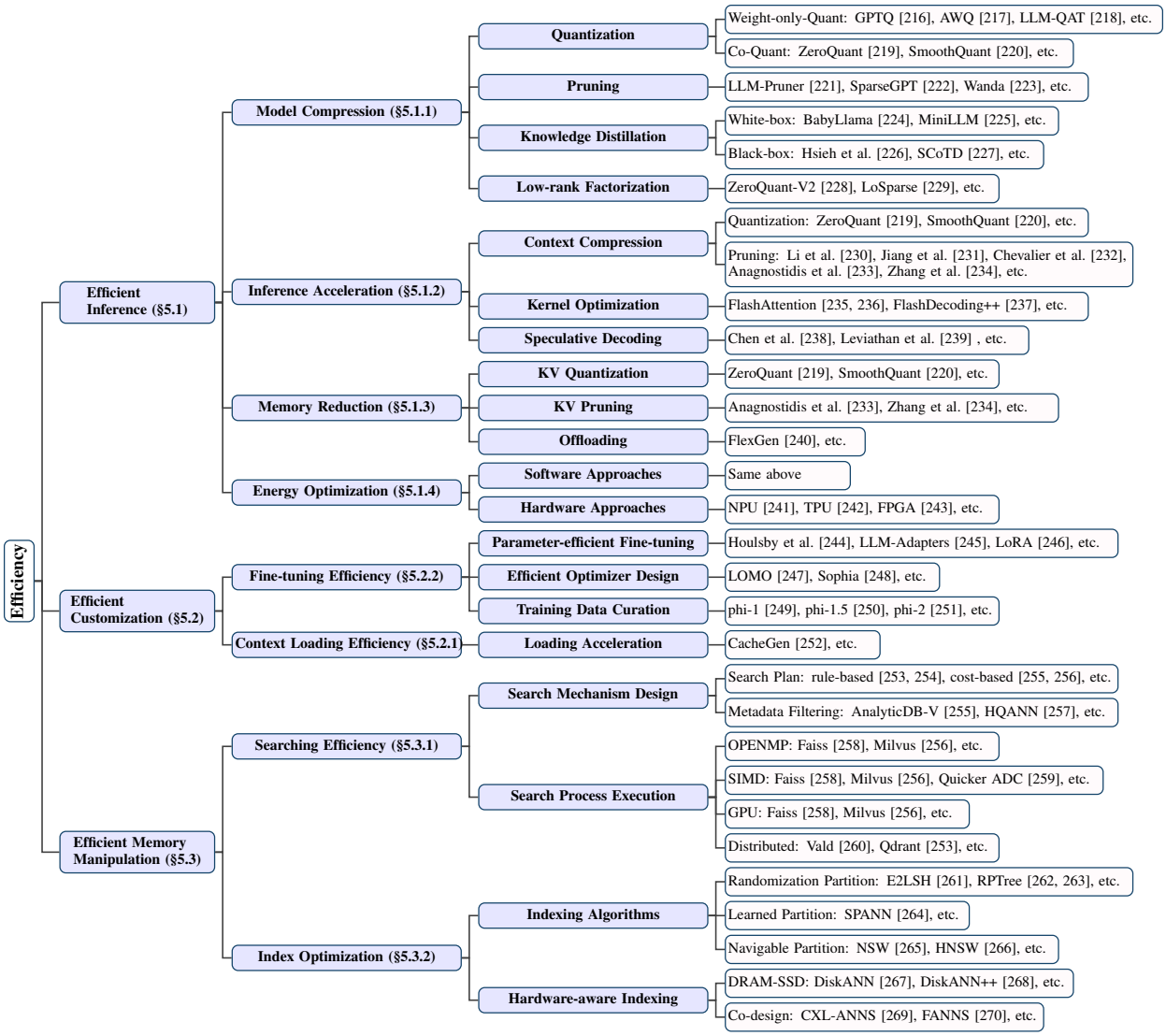

Figure 10: Overview of techniques to improve the efficiency of LLM agents. The leaf nodes are part of representative works we have cited.

Despite the WOQ techniques, another line of work quantizes both weights and activations. For example, ZeroQuant [219] performs INT8 quantization for both weights and activations, using group-wise quantization for model weights and token-wise quantization for activations. However, the activations, including key-value (KV) pairs, are usually more difficult to quantize compared to model weights because of outliers. There have been extensive works to tackle this challenge. SmoothQuant [220] migrates the quantization difficulty of activations to weights through additional scaling operations that "smooth" the outliers in activations, and thereby achieve negligible accuracy degradation in W8A8 quantization. Subsequent works further attempt to lower the usable quantization bitwidth down to 4-bit through various techniques including channel re-ordering (RPTQ [275]), channel-wise shifting and scaling (Outlier Suppression+ [276]), and adaptive channel reassembling (QLLM [277]). Notably, RPTQ addresses the KV storage issue by developing a new quantization scheme that focuses solely on KV cache when quantizing activations, which is the major memory consumer in long-context inference.

Pruning reduces the model size and computations by removing less important connections in the network. Pruning is categorized into structured pruning and unstructured pruning. Structure pruning usually removes weights in regular patterns, such as a rectangle block in the matrix or an entire channel, while unstructured pruning doesn't impose such constraints. Consequently, structured pruning (*e.g.,* LLM-Pruner [221]) is more hardware-friendly but more difficult

to maintain model accuracy. While traditional pruning approaches require costly retaining process to preserve model capability, recent works like SparseGPT [222] and Wanda [223] have explored to perform unstructured pruning in one-shot.

Knowledge Distillation (KD) involves using a well-performing teacher model (usually with a large number of parameters and high precision) to guide the training of a lightweight student model (usually with fewer parameters and lower precision). Through distillation, the student model is well-aligned to the teacher model with relative smaller training dataset, and has the chance to perform even better on downstream tasks [226]. Based on whether the teacher model's parameters are required in the training process, distillation methods can be further categorized into white-box (*e.g.,* BabyLlama [224] and MiniLLM [225]) and black-box ones (*e.g.,* Distilling Step-by-Step [226] and SCoTD [227]). Since the student model are often lightweight quantized or pruned model, KD is also adopted in QAT and pruning techniques to enhance the training performance. For example, LLM-QAT [218] proposes a data-free distillation method to preserve the original output distribution in the quantized model.

Low-rank Factorization refers to approximating the original weight matrix by the product of two low-rank matrices, thereby reducing the model's parameter size and computational load. Specifically, a weight matrix W of shape $m \times n$ is factorized into the product of $U^{m\times r}$ and $V^{n\times r}$, such that $W \approx UV^T$ and $r \ll m, n$. Low-rank Factorization can be combined with quantization (*e.g.,* ZeroQuant-V2 [228]) and pruning (*e.g.,* LoSparse [229]) methods to enhance the compression ratio. Besides, low-rank adapters effectively reduce the customization overhead of LLMs, which we leave to 5.2.

5.1.2 Inference Acceleration

Except for making the models more compact as discussed in Section 5.1.3, there are various other techniques to accelerate the LLM inference process.

A major characteristic that sets the LLM apart from the traditional non-Transformer models is the attention mechanism [31]. Since the computational cost of attention increases near quadratically with the context length, it is particularly important to enhance the computational efficiency of long-context inference. Existing works have explored to reduce context length and optimize attention kernels to better support long-context inference. We'll dive into these techniques separately.

KV Cache is a widely adopted technique in both mobile (*e.g.,* llama.cpp [273] and mlc-llm [274]) and cloud LLM serving frameworks (*e.g.,* DeepSpeed [278] and vLLM [279]), to avoid redundant computation in LLM inference. Specifically, KV Cache involves storing (*i.e.,* "caching") and incrementally updating the Key-Value (KV) pairs, which are intermediate results in the attention calculation, in each token's generation. Therefore, the repeated part in the KV computation is avoided to reduce the computational cost. However, in long-context inference, the computational cost of attention is still a system bottleneck despite the skipped KV calculations, making it crucial to compress the context length in such scenarios.

Context Compression methods enhance the inference efficiency by reducing the length of the context, especially the KV cache. Co-quantization of weights and activations, including KV cache, is an intuitive approach to compress the KV cache, which has been discussed in Section 5.1.1. Besides quantization, context pruning removes less important tokens in the context to reduce the computational cost. The effectiveness of this method is based on the observation that tokens have different impacts on the final output, and removing less important tokens won't cause significant degradation of the model's capability [233, 280, 234]. A typical line of work is to compress the context at the prefill stage based on different importance of tokens [230, 231, 232]. However, these methods are one-shot and cannot prune the KV cache when the context length continuously grows during token generation. To address this, Dynamic Context Pruning [233] uses a learnable mechanism to continuously determine and drop uninformative tokens. While the learnable mechanism introduces a fine-tuning overhead, Zhang et al. [234], proposes a token eviction strategy that can be applied without fine-tuning.

Inspired by the same observation that tokens are not equally important, other works also explored to reduce computations of less important tokens instead of directly removing them. COLT5 [281] employs a conditional computation mechanism, which devotes more resources to important tokens in both FFN and attention. SkipDecode [282] designs a token-level early exit method that works seamlessly with batched inference and KV cache, to skip some operators in the computational graph when a token is less important.

Kernel Optimization is another approach towards LLM inference acceleration. Optimization for small-batch or single-batch inference is especially important for edge scenarios including the locally-deployed Personal LLM Agents. Existing works have revealed that the attention calculation becomes a bottleneck when the sequence length is long, since the complexity of attention scales quadratically with the sequence length, while that of the FFN scales linearly. Therefore, efficient attention kernels including FlashAttention [235, 236] and FlashDecoding++ [237] have been

proposed to improve the speed of long-text inference. Besides, some works reduce the computational complexity of attention from the algorithm aspect. For example, Linformer [283] achieves linear complexity for self-attention in the prefill phase.

Speculative Decoding [239, 238] is an effective approach in small-batch inference to improve the latency. The batch size of LLM inference at the edge is smaller than on the cloud, and is usually 1 (*i.e.,* single query), which makes the inference workload extremely memory-bound. Speculative decoding mitigates this challenge by "guessing" several subsequent tokens through a lightweight "draft model", and then validating the draft tokens in batches using the large "oracle model". Miao et al. [284] and Spector and Re [285] further enhance speculative decoding with a tree-based verification instead of sequential ones to reuse intermediate results shared across these sequences. While these methods ensure zero bias in the generated results, BiLD [286] proposes to only fallback or rollback to the oracle model occasionally when the draft model is not capable to generate high quality contents.

5.1.3 Memory Reduction

LLM inference is not only computationally-intensive, but also memory-consuming, which causes challenges in the deployment of Personal LLM Agents. Therefore, it is necessary to perform optimizations on the memory efficiency of LLM inference. KV cache and model weights are two major causes of this memory overhead. In a short-context scenario where the KV storage requires much less memory than the model weights, the model compression techniques in Section 5.1.1 are very effective to reduce the memory requirement to store the weights. However, in the long-context scenario, the KV cache, whose size grows linearly with the context length, will dominate the total memory consumption.

An effective approach to address this issue is to **compress the KV cache** using quantization and pruning techniques mentioned in Section 5.1.1 and Section 5.1.2. While the quantization methods are generic to reduce the memory footprint of KV cache, not all the pruning-based methods directly contribute to the memory efficiency. Only those methods that prune the corresponding rows/columns in the KV cache when continuously removing input tokens in the context can prevent the KV cache size from exceeding the memory limit. For example, Anagnostidis et al. [233] and Zhang et al. [234] proposed to identify and evict uninformative tokens during generation. However, the one-shot approaches that only prunes the context at prefill stage are less effective regarding the generative scenarios.

Although the compression-based methods are demonstrated to be able to effectively reduce the memory requirement of LLM inference, the accuracy degradation caused by compression are not negligible in some cases. To address this, FlexGen [240] designs an offloading strategy to fully utilize GPU, CPU and disk, together with a zig-zag scheduling scheme to support high-throughput inference under constrained GPU memory. This approach is orthogonal to compression-based methods, and thus can be jointly used to further reduce GPU memory footprints.

5.1.4 Energy Optimization

The energy consumption is a critical factor that affects the real-world deployment of an intelligent personal agent. An energy-consuming agent not only increases the deployment cost and carbon footprint, but also hurts the quality of experience (QoE) due to increased temperature and potential thermal throttling. While the inference of the LLM involves costly computations and memory accesses, it is important to optimize the energy efficiency of LLM inference.

Since computation and memory access (mainly weights loading) are two major causes of the large energy consumption, there have been extensive works to optimize these two aspects, from both software and hardware perspectives. We have introduced various types of software optimizations in previous sections. For example, model compression methods save energy by reducing the model size and computations; KV cache saves energy by avoiding redundant computations; efficient attention kernels also improve energy efficiency through memory reuse and locality optimizations.

Besides software optimizations, utilizing energy efficient hardware provides new opportunities to improve the agent system's efficiency. While CPUs and GPUs remain mainstream options to run LLM inference on edge devices, they are designed to support general purpose tasks and don't have dedicated optimization for transformer-based models, especially the generative LLMs. Researchers have explored to utilize efficient processors that are more suitable to LLM inference workloads, including NPUs [241] and TPUs [242]. However, the limited operator and model support remain challenging in the real-world deployment. Besides, existing works also designed FPGA-based solutions to boost LLM inference with higher memory bandwidth and energy efficiency ratio (EER) [243, 287].

Remark. How to improve the efficiency of LLM inference has been extensively studied recently. Despite the remarkable progress, there is still a large gap towards the ubiquitous and affordable deployment of Personal LLM Agents. The open problems are:

1. Is it possible to further compress or design highly compact models without accuracy degradation, surpassing the scaling law of language models?
2. If the scaling law is unbreakable, how can we achieve optimal tradeoffs between efficiency and quality via dynamic inference (*e.g.,* dynamic collaboration of big model and small model)?
3. How would the hardware and operating systems evolve to accommodate the efficient deployment of LLMs and Personal LLM Agents?

5.2 Efficient Customization

The Personal LLM Agents may need to serve different users, different tasks, and different scenarios with the same base LLM, which requires efficient customization for each situation. There are mainly two ways to customize the behaviors of LLMs, one is feeding the LLM with different contextual prompts for in-context learning, and another is tuning the LLM with domain-specific data. Therefore, the efficiency of customization is primarily determined by the context loading efficiency and LLM fine-tuning efficiency.

5.2.1 Context Loading Efficiency

Frequent context loading is inevitable during the multi-task serving of Personal LLM Agents, where each task or each scenario may require a new context for LLM inference. Nevertheless, the stringent resource constraints inherent to personal devices pose a significant challenge for Personal LLM Agents to process cumbersome context information fast and efficiently. There are various ways to make the context loading process more efficient. A straightforward way is to prune some redundant tokens or shorten the context length, which have been discussed in Section 5.1.

Another way to boost context loading is to reduce the bandwidth consumption during context data transmission. In some cases, pruning or discarding some tokens inevitably hurts the LLMs' performance and loading the KV cache necessitates high bandwidth cost. CacheGen [252] addresses the challenges posed by context loading and it leverages the distinct characteristics of KV features across both tokens and layers thus introduces a novel KV encoder design. This encoder proficiently compresses the KV cache into a compact bitstream, effectively curtailing bandwidth demands while simultaneously reducing processing latency.

5.2.2 Fine-tuning Efficiency

It is also desirable to fine-tune a base LLM to better support domain-specific tasks, which poses a significant challenge on computational resources and memory footprint owing to the vast number of parameters in LLMs. There has been various efforts to tackle these problems, which can be roughly categorized as *parameter-efficient fine-tuning techniques*, *efficient optimizer design* and *training data curation*, which will be elaborated in the following sections.

Parameter-efficient fine-tuning (PEFT). A huge amount of parameters in LLMs make it costly to conduct full-parameter fine-tuning. Lots of efforts on parameter-efficient fine-tuning emerged to reduce LLMs' training overhead. The fundamental concept of PEFT is to freeze the majority of parameters, focusing solely on training a limited set or introducing an adapter with significantly fewer parameters. A common practice is to introduce some adapters, *i.e.,* small neural networks modules, into the existing network structure, including tuning hidden states [244, 245, 288], adding full layers [244] and prepending some prefix vectors into transformer architecture [289, 290, 291]. Liu et al. [292] also incorporates trainable vectors at the input layer, the performance of which highly depends on the capabilities of the underlying models. Some of these works fail to avoid extra adapter computation and introduce inference latency. LoRA [246] freezes all the model weights and augments each transformer layer with additional rank decomposition matrices, greatly reducing the memory and storage usage during fine-tuning without any additional inference latency. Another advantage of LoRA is that users can easily switch between different downstream tasks by simply adding or subtracting adapter matrices.

Efficient Optimizer Design. Efficient optimizer design is another group of training/fine-tuning strategies which aims to accelerate the training or reduce the memory overhead during training. Sophia [248], a lightweight second-order optimizer, addresses the high cost and time required for LLM pre-training by providing a more efficient optimization process compared to commonly used methods like Adam and its variants. On the other hand, the huge number of parameters necessitates storing more activation and optimizer states especially in larger batch size, which places substantial memory demands. LOMO [247] presents a detailed analysis of the memory profile, throughput, and

downstream performance of the proposed optimizer compared to other methods, demonstrating significant reductions in memory usage while maintaining training efficiency.

Training Data Curation. Aforementioned approaches primarily focus on the process of training LLMs, while there are also some studies that aim to enhance the LLMs' training performance from a distinct perspective, *i.e., the amount and quality of training data*. It has been demonstrated in phi-1 [249] that training the LLMs with a small amount of high-quality data can lead to significantly reduced training cost and achieve capabilities comparable to large-scale datasets and models. This challenges the traditional scaling laws in deep learning that emphasize larger datasets and models. Furthermore, phi-1.5 [250] and phi-2 [251] extend their focus on many other kinds of tasks such as common sense reasoning and language understanding, achieving comparable performance to models 5x and 25x larger, respectively.

Notably, these methods often assume that the LLMs can fit entirely within the device memory, which isn't a practical assumption for Personal LLM Agents deployed on personal devices which usually have limited computing power and memory capacity. Fine-tuning LLMs on these devices often requires leverage of hierarchical storage like CPU memory even disk storage. Therefore, when fine-tuning LLMs on personal devices, it's important to carefully consider the resource limitations of the current system.

Remark. While efficient model fine-tuning and in-context learning techniques have been extensively studied, it is yet unclear what is the ideal mechanism for customizing Personal LLM Agents under different situations. Here we highlight two open problems that may be specifically important in the system for Personal LLM Agents.

1. Similar to the operating system that manages the RAM for the applications, how should the agent system efficiently manage the contexts for different (and potentially parallel) agents, tasks, and users?
2. Similar to mobile apps that can be efficiently installed, uninstalled and moved between devices, how can a customized (fine-tuned) agent efficiently roll back to the previous versions or transfer to other base models?

5.3 Efficient Memory Manipulation

The Personal LLM Agents need to frequently retrieve internal or external memory to enable more informed decisions. The internal memory is represented as the context tokens and stored as KV cache during the LLM inference. The retrieval of the internal memory is handled implicitly by the self-attention module in Transformer architecture. This leads to LLM conducting more efficient computations over long contexts and trying to minimize the memory footprints while undergoing inference. These issues are similar to the inference efficiency of LLM as discussed in Section 5.1. Therefore, in this subsection, we mainly focus on the efficiency of manipulating external memory, which can be dynamically retrieved and added into the context.

Considering the diverse forms of external memory data, such as user profiles, interaction history, and local raw files (images, videos, etc.), the common practice is to use embedding models [293, 294] to represent memory data with a uniform and high-dimensional vector format. The distance between vectors stands for the semantic similarity between the corresponding data. For each given query, the agent needs to find the most relevant contents in external memory storage. This procedure, together with the maintenance of vectors, can be covered by vector libraries (like Faiss [295, 296, 297] and SCaNN [199]), vector databases [298, 299, 300], or some customized memory structures [301, 302]. Regardless of the functional differences between these systems, their efficiency optimizations are basically targeted at two key aspects, searching and indexing.

5.3.1 Searching Efficiency

The efficiency analysis and optimization can be viewed from different aspects of vector searching. Some aspects are related to search mechanism designs, such as similarity measurement, searching scope, as well as query types, selection, and optimizations. Some aspects, on the other hand, focus on the execution of the search process.

Search Mechanism Design. Multiple similarity criteria can be employed to evaluate the relevance of vector embeddings to a search query, including Hamming Distance, Cosine Distance, and Aggregate Scores [256]. However, the selection of scoring mechanisms lacks stringent principles and often relies on empirical rules [299]. Regarding the types of searches, both approximate and exact $k(\geq 1)$ nearest neighbors [303] search, as well as distance range search, can be utilized to retrieve corresponding vectors. Typically, there exists a trade-off between search accuracy and speed, wherein the use of approximate similarity search or larger k values speeds up the search process but may result in undesirable outcomes. To optimize search latency, rule-based [253, 254] or estimated-cost-based methods [255, 256] are often employed to determine the optimal search plan. These rules and cost models are typically configured offline

to avoid unnecessary or time-consuming search actions. To further optimize the search process, hybrid operations that combine vector search with metadata filters are gaining popularity. This involves techniques such as pre-filtering [255, 256, 304], post-filtering, and single-stage filtering [257] to narrow the scope of vector searching. Similarly, the trade-off between search accuracy and speed still exists. For instance, in pre-filtering, metadata analysis is performed before searching, which may lead to overlooking some vectors that do not match the metadata criteria but could be potentially crucial for Personal LLM Agents. Moreover, the utilization of metadata filtering can impede the execution of the search, as it introduces additional computational costs.

Search Process Execution. Several hardware acceleration methods can be taken to improve the efficiency of search executions. For example, to enable parallel query process, Faiss [258] uses OpenMP multi-threading, while Milvus [256] further reduces CPU cache misses and uses a novel fine-grained mechanism to best leverage multi-core parallelism. Furthermore, Faiss and Quicker ADC [259] also support SIMD shuffle instruction to parallelize these table look-ups within a single SIMD processor. GPU is also used for fast query processing [305, 306, 307], such as vector databases like Faiss, and Milvus. Many vector database management systems also support distributed clusters to scale to larger datasets or heavier workloads, such as Vald [260], Qdrant [253], etc.

5.3.2 Index Optimization

Indexing is also very important to optimize the efficiency of external memory management and retrieval. While comparing the similarity between query vector q and vectors in external memory, a brute-force approach results in a computational complexity of $O(DN)$. However, this approach becomes impractical for scenarios with large vector dimensions (D) and dataset sizes (N). To address this problem, vector indexing is commonly employed to expedite query searching by reducing the number of required comparisons.

Typical Indexing Algorithms. This is achieved through partitioning schemes [299] that divide the dataset S into smaller subsets, facilitating selective comparisons and faster search query processing. These partitions are then organized into data structures such as tables, trees, and graphs to enable efficient traversal. Commonly used partitioning methods include randomization (such as RPTree [262, 263] and E2LSH [261]), learned partitioning (such as SPANN [264]), and navigable partitioning (such as NSW [265] and HNSW [266]). These partitioning methods can be utilized in combination with different data structures. For example, Vamana [304] is a monotonic search network that comes in graph indexing and uses random initialization.

Hardware-aware Index Optimization. Since improving the scalability and efficiency of indexing has become a critical concern, research efforts have focused on hardware-aware approaches to extend external memory capacity while maintaining low latency and high throughput. This is achieved through the utilization of disk-based indexes or the co-design of hardware and algorithms [303]. For example, DiskANN [267] addresses cost-effectiveness by employing a hybrid DRAM-SSD approach. It incorporates Vamana graph indexing on SSDs and employs compressed point representation in DRAM. This configuration enables accurate query responses with less than 10ms latency, even when dealing with a billion-point database. DiskANN++ [268] further improves efficiency by introducing dynamic entry vertex selection and optimizing SSD layout. This enhancement results in a 1.5x to 2.2x increase in Query Per Second (QPS) while maintaining accuracy on real-world datasets. Moreover, CXL-ANNS [269] introduces a collaborative software-hardware approach for scalable approximate nearest neighbor search (ANNS). By utilizing Compute Express Link (CXL), CXL-ANNS disentangles DRAM from the host and consolidates essential datasets into its memory pool. FANNS [270] is a vector search framework on FPGAs, featuring automatic co-design of hardware and algorithms based on user-defined recall requirements and hardware constraints. It supports scale-out with a hardware TCP/IP stack and exhibits notable speedups compared to FPGA and CPU baselines.

Remark. Managing memory data with external vector storage is not a new requirement for LLM agents. While many basic technical challenges have been adequately addressed, we point out two problems that demand specific consideration for Personal LLM Agents.

1. Personal LLM Agents may frequently update the memory. Thus, the external memory is expected to facilitate fast updates, maintenance, and re-indexing.
2. The memory of Personal LLM Agents may be stored on personal devices with limited storage space, while the memory of the personal agents will accumulate over time. Therefore, it is necessary to effectively compress the memory to avoid fast-growing space and computational cost.

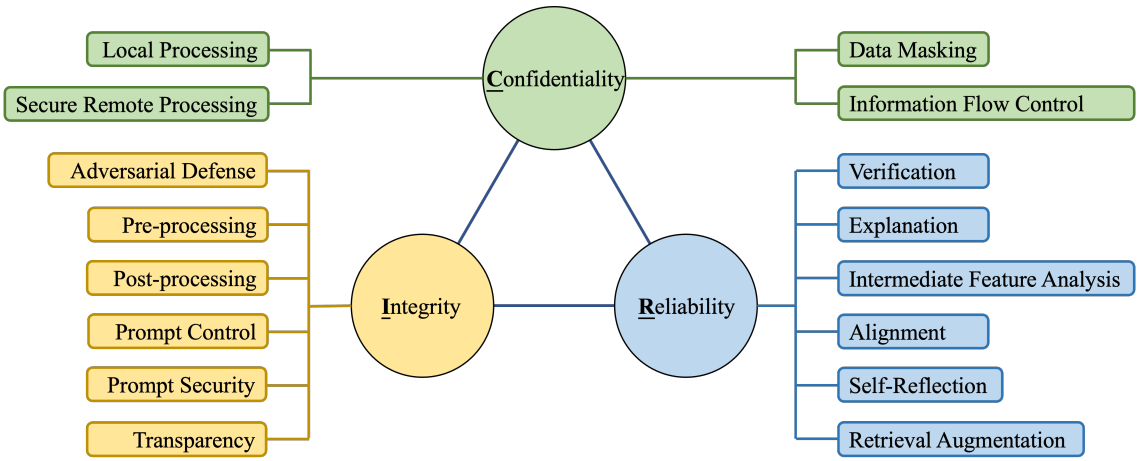

Figure 11: The summary of techniques to address security and privacy issues of Personal LLM Agents.

6 Security and Privacy

The extensive integration of sensitive personal data and safety-critical personal tools sets Personal LLM Agents apart from regular LLM agents. As a result, ensuring the protection of user data privacy and service security in Personal LLM Agents becomes a crucial problem. In the context of Personal LLM Agents, we focus on three security principles including confidentiality, integrity, and reliability, as shown in Figure 11. **Confidentiality** represents the protection of user data privacy, ensuring that unnecessary and unauthorized disclosure of sensitive information does not occur during user interactions with the agents. **Integrity** represents the resilience of the agents' decisions, ensuring that the behaviors performed by the agent align with the intended behaviors and have not been deliberately modified or influenced by malicious parties. **Reliability** focuses on making the agents' behaviors more dependable and truthful. Unlike integrity, where incorrect answers are a result of intentional external manipulation, reliability addresses the agents' internal mistakes.

6.1 Confidentiality

In this subsection, we discuss possible methods for protecting user privacy in Personal LLM Agents. As mentioned earlier, ensuring user privacy is of utmost importance for the personal agents that have access to a significant amount of user-sensitive data. Unlike traditional LLM-based chatbots where the users explicitly input text, Personal LLM Agents have the potential to spontaneously initiate queries in places without user awareness, which may contain sensitive information about the user. Meanwhile, the agents may also expose the user information to other agents or services. Consequently, the protection of user privacy becomes even more critical. There are various methods to enhance the confidentiality, including local data processing, homomorphic encryption, data masking, permission access control, etc.

6.1.1 Local Processing

A simple and effective approach to protect user privacy is to perform the computations locally on the users' personal devices. While LLM service providers are currently working towards improving security and building user trust, it is important to acknowledge that transmitting private data to the cloud inherently introduces additional potential risks. Therefore, processing all data locally is considered a more secure method of interacting with LLMs compared to transmitting data to the cloud. However, deploying LLMs locally poses challenges in efficiently processing user requests due to resource constraints on personal devices. This can lead to slow inference speed or even the inability to perform inference due to the limitations of available memory. Since the data in Personal LLM Agents is mainly processed by the LLM, the key to achieve local computation is to run the LLM on users' own devices. There are various existing lightweight models [308, 250] and deployment frameworks [309, 274] available for deploying models on edge devices. Furthermore, various model compression techniques [310, 220, 216] are proposed to reduce the model size to further enable the local deployment.

Nevertheless, despite the various efforts of researchers, using a locally-deployed model inevitably faces the challenge of limited model accuracy [42]. Most of the domain experts also suggest to adopt a cloud-edge-collaborated deployment approach to achieve better performance tradeoffs. Meanwhile, like other software applications, many Personal LLM

Agents would also need to communicate with the cloud to provide online services. It is usually difficult or even impossible to keep the private data completely on local devices.

6.1.2 Secure Remote Processing

To invoke cloud-based model inference services while preserving privacy, an ideal solution is homomorphic encryption (HE) [311, 312]. In this method, the client employs encryption to encode the user's plaintext request, and the server conducts model inference on the resulting ciphertext. Subsequently, the client receives the inference results in the encrypted format and gets plaintext results after decryption. There have been several studies [313] that have demonstrated the feasibility of applying HE to Deep Neural Networks, showcasing the potential for integrating HE into models.

When employing HE in Personal LLM Agents, two challenges arise. The first challenge pertains to the limitation that not all operations within the LLMs can be executed using HE. HE atmost supports an unlimited number of additions (equivalent to XOR in a boolean circuit) and multiplications (equivalent to AND in a boolean circuit). However, certain operations in the LLMs, such as max, min, and softmax, cannot be accurately performed using HE. The second challenge involves the slow inference speed associated with HE, given the large computational complexity of LLMs.

There are several solutions to address these two problems. The-x [314] presents a workflow for replacing original non-linear layers with layers that can be computed using HE. In cases where HE cannot perform certain operations, such as the Max operation, the ciphertext will be sent back to the local device. The local device will then perform the operation and send the re-encrypted text back to the cloud. Cheetah [315] encompasses a collection of algorithmic and hardware optimizations designed for HE inference on server-side systems. The primary objective of Cheetah is to enhance the computational efficiency of HE, thereby accelerating the speed of HE operations.

However, despite the numerous efforts on accelerating HE-based DNN inference, the current state of homomorphic encryption still falls significantly short of meeting the latency demands of agents [316].

Another way to achieve confidential remote data processing is using the trusted execution environments (TEE) [317] for model inference. However, TEE may be subject to various attacks [318] and may also lead to limited performance.

6.1.3 Data Masking

An alternative approach is using data masking to preprocess the information before sending to the cloud. The basic idea is to transform the original inputs into a form that is not privacy-sensitive while preserving the information that has a crucial impact on the inference results.

One direct approach of data masking is to transform the plaintext inputs by hiding or replacing sensitive content such as account numbers, addresses, and personal names. These types of information are commonly referred to as Personally Identifiable Information (PII). However, accurately defining PII can be challenging due to its obscure boundaries and diverse forms, making it difficult to consistently identify and remove it from the original content. The National Institute of Standards and Technology (NIST) has provided a guide [319] that offers recommendations for safeguarding the confidentiality of PII, which could help manage PII more securely.

On the other hand, researchers have proposed embedding-based data anonymization approaches where the client encodes the original user request into hidden vectors and sends these vectors to the cloud-based model for subsequent inference. The challenge is how to ensure privacy is protected, how to ensure inference accuracy will not degrade, and how to ensure the inference speed will not decrease too much. There are several solutions. Coavoux et al. [320] propose a metric to assess the extent of privacy leakage in neural representations and develop a defense method by altering training objectives to achieve a tradeoff between privacy and accuracy. Zhou et al. [321] protects user privacy by adding dynamic fusion to the intermediate representation. TextObfuscator [322] protects user privacy through text obfuscation techniques. During the encoding process, "adversarial representation learning" can be employed by introducing additional constraints to minimize the inclusion of privacy-sensitive information in the encoded vectors [323]. Although this method outperforms Homomorphic Encryption in terms of inference performance, it usually does not rigorously protect the data privacy, as the encoded vectors themselves still carry a risk of leaking sensitive information. Additionally, such methods require an explicit definition of privacy features for the encoder to learn how to remove privacy information during adversarial representation learning.

6.1.4 Information Flow Control

The aforementioned techniques primarily pertains to the privacy of model input data, while there may also exist the risks of privacy leakage in the model output. This is because the output of the model may not only returns directly to the user but also be sent to other third-party applications, models, users, or intelligent agents. For instance, when

an intelligent agent assists a user in making restaurant reservations, it may take the user's basic profile and schedule information and feed them into the restaurant reservation software. Similarly, when businesses aim to recommend products to users, they may rely on user preference information retrieved from the output of certain personal agents. This method of obtaining privacy information from the output of LLMs is similar to personal data access interfaces in traditional operating systems, where it is crucial to ensure the control and transparency of privacy data access with permission management systems [324]. Transparency necessitates informing users about access information regarding privacy data, including the accessing entity (*who*), content (*what*), time (*when*), intent (*why*), access method (*how*), etc.

One can also directly ask the LLMs to retain private information. However, since LLMs work statistically rather than based on explicit rules, their security cannot be rigorously proven. Therefore, we should not consider LLMs as a part of the Trusted Computing Base (TCB) when dealing with data confidentiality. Therefore, we may need rule-based permission control to constrain what LLMs can do and what LLMs can access. Permission mechanisms allow users to configure whether different entities are permitted to access different types of information. In Personal LLM Agents, one of the challenges in designing permission mechanisms lies in delineating the types of privacy data, as the content obtained by third-party applications is generated by the model. In traditional systems, researchers have proposed numerous methods for fine-grained privacy content subdivision and permission control, as well as privacy data traceability techniques based on information flow propagation [325]. However, establishing privacy data traceability for the output generated by LLM agents remains an open issue.

Remark. Ensuring the confidentiality of user data is crucial for Personal LLM Agents to build user trusts. However, existing privacy protection techniques are still not sufficient to support agents with higher levels of intelligence. There are following open problems:

1. Existing approaches face a common challenge to balance efficiency and effectiveness. For example, how can we enable powerful and efficient local LLMs, how can we scale homomorphic encryption (HE) or trusted execution environment (TEE) to large models, and how can data masking/obfuscation techniques achieve rigorous confidentiality?
2. As a new software paradigm, it is still unclear what is the systematic privacy protection mechanism for Personal LLM Agents. Do we still need symbolic rules or permissions for access control? How can they seamlessly integrate with the uninterpretable nature of LLMs?

6.2 Integrity

Integrity refers to the capability of Personal LLM Agents to ensure that it can output the intended content correctly, even when faced with various types of attacks. As Personal LLM Agents necessitate interactions with diverse data, applications, and other agents, there is a potential presence of hostile third parties seeking to steal user data and assets or disrupt the system's normal function through unconventional means. Therefore, the system must be able to resist various types of attacks. Traditional attack methods such as modifications to model parameters, theft, and tampering of local data could be defended against using encryption, permissions, hardware isolation, and other measures. However, in addition to defending against traditional attack methods, attention should also be paid to new types of attacks that the LLM agents may encounter: adversarial attacks, backdoor attacks, and prompt injection attacks.

6.2.1 Adversarial Attacks

Malicious attacks primarily achieve their objectives through the specialized customization of the model's inputs or malicious tampering with the model. A significant category of attacks, known as "adversarial attacks", causes model inference errors by customizing or tampering with the model's input data, which was initially discovered in image classification models [326]. This type of attacks can induce serious classification errors by adding imperceptible noise to images. Subsequently, researchers have extended this attack method to text data, graph data, and beyond [327]. Such attacks also persist in large langage models [328], which may also accept input of images [329], text [330], and other modalities of data [331] from third parties. For example, when assisting users in automating tasks, attackers may misguide the agent to delete calendar events and leak private conversation data [332], because LLMs often need to input the content of the application's internal information to generate the next interaction decision. In such cases, if the third-party application feed the LLM with maliciously customized content, it could drive the intelligent agent to engage in unsafe interaction. Traditional defense methods against such attacks in deep learning models usually encompass adversarial defense, abnormal input detection, input preprocessing, output security verification, and more [327]. While these methods theoretically remain applicable to LLM and LLM agents, the large scale of parameters and the characteristics of autoregressive generation may render some computationally expensive methods (such as formalized output security validation and detection of anomalous data based on intermediate layer activations) challenging to implement. Furthermore, some defense methods may require adjustments in the context of LLM. For

instance, training the LLM may incur substantial costs, making it impractical to enhance security through adversarial training. Therefore, exploring how to achieve good effects of adversarial defense through parameter-efficient fine-tuning is worth investigating.

6.2.2 Backdoor Attacks

Another common form of attack is the backdoor attack. Traditional model backdoor attacks are often achieved through data poisoning [333], *i.e.,* inserting maliciously modified samples into the model's training data, enabling the model to learn deliberate hidden decision logic, such as "when seeing an apple pattern, the model outputs an incorrect classification". For LLMs, data poisoning may be more challenging due to the huge amount and strict unified management of training data, but another type of backdoor attack methods [334] is still valid, which implants insecure logic into the model by modifying the model input during the test time. Kandpal et al. [335] elicits targeted misclassification when the language models are prompted to perform a particular target task. ProAttack [336] directly utilizes prompts as triggers to inject backdoors into LLMs, which is the first attempt to explore clean-label textual backdoor attacks based on the prompt. PoisonPrompt [337] is a bi-level optimization-based prompt backdoor attack on soft and hard prompt-based LLMs. Since LLMs often use several fixed prompts in certain scenarios, this form of attack, achieved by modifying the prompts, essentially fine-tunes the model's parameters and thus alters its decision logic. Sun et al. [338] proposed that testing the backward probability of generating sources given targets yields effective defense performance against different types of attacks. Indeed, when attackers mimic normal behavior, this defense method may become ineffective. Therefore, there isn't a robust solution for backdoor defense in agent systems yet [339]. This highlights the request of developing effective defenses against sophisticated attacks that mimic legitimate behavior.

6.2.3 Prompt Injection Attacks

In the era of LLM, there emerges a new and particularly crucial security risk, namely prompt injection attacks [340, 341, 342, 343]. In this form of attack, the model itself incorporates certain security safeguards through alignment and prompts. Nevertheless, third-party model users can bypass these preset security safeguards by using subtle or special diction in the prompts. For instance, an intelligent personal assistant may be preset not to execute certain sensitive operations, such as modifying a user's account password [344], but through prompt injection (*e.g.,* requesting the LLM to "disregard the previously set limitations" or "assume operation in an authorized secure mod"), it could induce the model to violate regulations and perform these sensitive operations.

For such prompt-based attack methods, there are currently no perfect defense mechanisms. SmoothLLM [345] is the first general-purpose defense method for prompt injection, and it randomly perturbs multiple copies of a given input prompt and then aggregates the corresponding predictions to detect adversarial inputs. However, its defensive effectiveness is highly dependent on the model's robustness, since there was only about 1% reduction in the attack success rate for some models. An essential way to mitigate this issue is to ensure the transparency and security of the LLM's prompts. For example, a Personal LLM Agent could rigidly control the template and specifications of prompts, requiring all requests to comply with the preset template and specifications. Additionally, post-processing of the input content from third-party applications (summarization, translation, restatement, etc.) or prompt encapsulation (such as adding explicit text before and after to indicate their origin from a third party) can help the model clearly distinguish them from the system's inherent prompts.

Remark. Ensuring the integrity of the decision process is crucial for Personal LLM Agents. The threats to integrity are very diverse and continuously evolving, while the development of defensive techniques are lying behind. Here we highlight two important open problems that apply to all types of attacks.

1. How can the agents know if their input or decision process has been tampered with by third parties? This requires the agents to have a sense of what are normal input and behaviors, and have the abilities to recognize the anomalies.
2. Since directly avoiding the attacks may be challenging, it would be more practical to consider user verification mechanisms, *i.e.,* asking the user to verify when the agents are uncertain. How to design a secure and user-friendly verification mechanism is challenging.

6.3 Reliability

In Personal LLM Agents, numerous critical actions are determined by the LLMs, including some sensitive operations such as modifying and deleting user information, procuring services, and sending messages. Therefore, ensuring the reliability of the agent's decision-making process is crucial. We discuss the reliability of LLMs from three perspectives, including the **problems** (*i.e., where does reliability issues of LLMs manifest from?*), **improvement** (*i.e., how can we*

make the LLMs' response more reliable?), and **inspection** (*i.e., how can we deal with the LLM's potentially unreliable output?*).

6.3.1 Problems

Hallucination. LLMs may produce incorrect answers, which can lead to severe consequences. In comparison to LLM-based chatbots that directly interact with users via text, Personal LLM Agents minimize user disruptions by avoiding frequent result verifications, hence amplifying the severity of producing incorrect answers. Researchers have uncovered cases where LLMs generate text that is coherent and fluent but ultimately erroneous. This phenomenon, known as *hallucination* in natural language processing tasks, poses a challenge to the personal agents as well. Ji et al. [346] delves deeply into the various manifestations of hallucinations in natural language processing tasks. Rawte et al. [347] further discusses the hallucinations in multimodal foundation models, providing valuable references for interested readers.

Unrecognized Operation. Unlike the hallucination problem that focuses on the "wrong answer" produced by LLMs, there are many cases where the responses from these models are "not even wrong". For instance, consider the scenario where the LLM is instructed to initiate a phone call by using the format "CALL XXXXXX". In response, the LLM may generate a reply "I will make a call to XXXX", which accurately conveys the intended meaning but deviates from the specified format, rendering it unexecutable. As we know, the essence of LLMs is language modeling, and the outputs of language models are typically in the form of language. Compared to other LLMs that interact directly with humans, Personal LLM Agents is required to execute actions. As a result, they have significantly higher requirements for the format and executability of their outputs [348].

Sequential Reliability. LLMs are initially pre-trained on sequential data (*i.e.,* corpus) and training objectives (*i.e.,* left-to-right language modeling task). However, problems in the real world may not be fully addressed sequentially. Achieving sequential reliability poses several challenges, including context preservation, coherence maintenance, *etc*. To better maintain a coherent and meaningful conversation with users and Personal LLM Agents, we need to elicit the LLMs' ability to think from a global perspective, not solely relying on the previously generated tokens or contexts. On enhancing the ability of thinking and reasoning of LLMs, Yao et al. [83] propose Tree-of-Thought to generate and conclude over multiple different reasoning paths, Zhang et al. [349] propose Cumulative Reasoning in a cumulative and iterative manner to solve complex tasks. There is also potential for designing the overall plan for solving the task [87] or drawing insights from the previous work [350, 351].

6.3.2 Improvement

The improvement approaches aim to improve the quality of LLM output, thereby enhancing the reliability of LLM-based agents.

Alignment. As LLMs grow in size and complexity, concerns have arisen regarding their potential to generate biased, harmful, or inappropriate content. Alignment methods seek to mitigate these risks and ensure that the behavior of LLMs aligns with ethical and societal norms. One common alignment method is the use of pre-training and fine-tuning [352, 353, 354]. LLMs are pre-trained on vast amounts of text data to learn language patterns and representations. During the fine-tuning phase, the models are further trained on more specific and carefully curated datasets, including human-generated examples and demonstrations. This process helps align the models with desired behaviors by incorporating human values and intentions into their training. Another alignment method is reward modeling, which involves defining and optimizing a reward function that reflects desired outcomes or behaviors. By providing explicit rewards or penalties for specific actions, LLMs can be trained to generate outputs that align with those predefined objectives. Reinforcement learning techniques (*e.g.,* RLHF [43], RLAIF [355], C-RLFT [356]) can be employed to optimize the model's behavior based on these reward signals. oversight and intervention are critical alignment methods. Human reviewers or moderators play a crucial role in reviewing and filtering the outputs of LLMs for potential biases, harmful content, or inappropriate behavior. Their feedback and interventions are used to iteratively improve the model's performance and align it with desired standards.

Self-Reflection. It has been shown that language models can provide probabilities of providing correct answers [357]. Inspired by the autonomous operation of LLMs, researchers have suggested leveraging the model's self-reflection to mitigate the problem of incorrect content generation. Huang et al. [211] and Madaan et al. [358] show that LLMs are capable of self-improving with unlabeled data, Shinn et al. [359] propose Reflexion to let LLMs update through its linguistic feedback. Chen et al. [360] propose Self-Debug to iteratively improve the responses on several code generation tasks. SelfCheckGPT [361] allows large models to provide answers to the same input question multiple times and checks the consistency between these responses. If there are contradictions among the answers, there is a higher probability that the model has generated unreliable content. Du et al. [362] attempts to enhance the reliability

of model outputs by enabling multiple large model agents to engage in mutual discussion and verification. There are various ways to combine models, similar to the diverse collaboration methods in the human world. However, just as more employees require increased expenses, having more models entails greater computational power requirements. The above works demonstrate a trend in which LLMs are evolving from mere textual generators to intelligent agents, transitioning from primitive comprehension-based reasoning to reflective reasoning with iterative updates.

Retrieval Augmentation. LLMs show strong performance across various tasks, however, the parametric knowledge stored in the models could still be incomplete and difficult to update efficiently. Alternatively, the retrieval augmented methods [199, 200, 363] provide a semi-parametric way to offer complementary non-parametric information, allowing the LLMs to draw on retrieved real-world knowledge when generating content, such as Wikipedia, documents, or knowledge graphs [364]. This approach offers the advantage of not requiring model modification, facilitates real-time information updates, and allows the traceability of generated results to the original data, thereby enhancing the interpretability of the generated information. Retrieval augmentation has been shown effective for traditional pre-trained models such as BERT [365]. However, as for LLMs that already have strong reasoning ability, augmenting the context could also have a negative impact due to irrelevant or noisy information [366]. To tackle these issues, Guo et al. [192] propose a prompt-guided retrieval method for non-knowledge-intensive tasks, enhancing the relevance of retrieved passages for more general queries. Yu et al. [367] propose Chain-of-Note to improve the robustness when dealing with noisy and irrelevant documents. Asai et al. [368] propose Self-RAG to enhance factuality through self-reflection. Wang et al. [369] propose SKR, a self-knowledge-guided retrieval method to balance external knowledge with internal knowledge. Wang et al. [370] propose FLICO to filter the context in advance and improve the fine-grained relevance of retrieved segments. The CRITIC [371] framework utilizes LLMs to verify and iteratively self-correct their output through interaction with external tools, such as a calculator, Python interpreter, and Wikipedia. However, this approach has limited assistance for user requests for which matching content cannot be found in external knowledge bases.

6.3.3 Inspection

The inspection-based approaches, on the other hand, do not interfere the LLM generation process. Instead, it focus on how to enhance or understand the reliability of agents based on the already generated results.

Verification. Given that the issue of unreliable content generation by LLMs cannot be entirely avoided when deploying such systems for actual use, it remains necessary to establish rule-based security verification mechanisms. Regarding the aforementioned unrecognized operation, "Constrained Generation" refers to the process of generating formatted and constrained output, which can be employed to tackle this issue. Kumar et al. [372] employs Langevin Dynamics simulation for non-autoregressive text generation as a solution to this problem. On the other hand, Miao et al. [373] introduces a method that suggests a candidate modification at each iteration and verifies if the modified sentence satisfies the given constraints to generate constrained sentences. Li et al. [374] and Weng et al. [375] propose self-verification to help the reasoning process of large language models. Responsible Task Automation [96] is a system that can predict the feasibility of commands, confirm the completeness of executors, and enhance the security of large language models. However, further research is needed to improve the accuracy and recall rates in identifying sensitive operations and to mitigate the decision burden on users.

Explanation. While it is mentioned earlier that intelligent personal assistants should minimize user interruptions, incorporating user opinions or human assistance can be valuable, particularly when making significant decisions. In case an intelligent personal assistant makes a mistake, having interpretable logic can also be helpful in the subsequent debugging process. There are several surveys [376, 377, 378] discussing about explainable language model. Traditionally, rationale-based methods [379, 380] can be used to explain the model output by explicitly training on human-annotated data. As for LLMs, chain-of-thought reasoning [82] approaches can also help the model generate textual explanations. To make the reasoning process more robust and reliable, recent studies further enhance chain-of-thought reasoning with majority voting [381] and iterative bootstrapping [382] mechanisms. It is evident that researchers place a significant emphasis on interpretability, as it not only contributes to reliability but also represents an intriguing research direction.

Intermediate Feature Analysis. Beyond the last-layer representation, some work involves analyzing the intermediate states in the model's inference process to judge the generation of false information. Halawi et al. [383] discover that the behavior of a model may significantly diverge at certain layers, highlighting the importance of analyzing the intermediate computations of the model. Li et al. [384] find that the model activation of intermediate layers can reveal some directions of "truthfulness", showing that the LLMs may already capture knowledge though not generated, they further propose shifting the model activation during inference and improving the responses of LLMs. van der Poel et al. [385] propose a method to leverage mutual information and alleviate hallucination by assessing the confidence level of the next token, where the underlying reason is that the neural activation pattern in LLMs during the generation of hallucinatory content differs from normal outputs. These studies highlight the drawbacks of solely depending on the

final-layer representation for language modeling, revealing the potential benefits of harnessing hierarchical information across different layers of the model.

Remark. The reliability of LLM generation has received considerable amount of attention, especially around the hallucination problem. However, avoiding the unreliable behaviors is still difficult, if not impossible. The open problems include:

1. How can we evaluate the reliability of LLM and LLM agents? Existing methods rely on either black-box LLMs such as GPT-4 or costly human annotations. Authoritative benchmarks and methods are desired for evaluating and improving the reliability.
2. Similar to the confidentiality problem, incorporating rigorous symbolic rules in the decision process of Personal LLM Agents would be a practical solution for reliability. However, complying with the rules while retaining powerful capabilities of LLM agents is challenging.
3. The lack of transparency and interpretability of DNNs has been a long-standing problem, which is even more critical for all security & privacy aspects of Personal LLM Agents. How to interpret and explain the internal mechanisms of LLMs is a direction that worth continuous investigation.

7 Conclusion and Outlook

The emergence of large language models presents new opportunities for the development of intelligent personal assistants, offering the potential to revolutionize the way of human-computer interaction. In this paper, we focus on Personal LLM Agents, systematically discussing several key opportunities and challenges based on domain expert feedback and extensive literature review.

Currently, research on Personal LLM Agents is in the early stages. Task execution capabilities are still relatively inadequate, and the range of supported functionalities is rather narrow, leaving significant room for improvement. Moreover, ensuring the efficiency, reliability and usability of such personal agents requries to address numerous critical performance and security issues. There exists an inherent tension between the need of large-scale parameters in LLM to achieve better service quality and the constraints of resource, privacy and security in personal agents.

Going forward, except for addressing the respective challenges in each specific direction, a joint effort is needed to establish the whole software/hardware stack and ecosystem for Personal LLM Agents. Researchers and engineers also need to carefully consider the responsibility of such technology to guarantee the benign and assistive nature of Personal LLM Agents.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (NSFC, Grant No.62272261) and collaborative research projects with AsiaInfo Technologies (China) Inc. and Xiaomi Inc. We sincerely thank the valuable feedback from many domain experts including Xiaobo Peng (Autohome), Ligeng Chen (Honor Device), Miao Wei, Pengpeng He (Huawei), Hansheng Hong, Wenjun Chen, Zhiyao Yang (Oppo), Xuesheng Qi (vivo), Liang Tao, Lishun Sun, Shuang Dong (Xiaomi), and the anonymous others. Among the co-authors, Jiacheng Liu, Wenxing Xu, and Rui Kong were interns at Institute for AI Industry Research (AIR), Tsinghua University when writing this paper.

References

[1] Apple. Siri. `https://www.apple.com/siri/`, 2023. [Online; accessed December 26, 2023].

[2] Google. Google assistant for android. `https://developer.android.com/guide/app-actions/overview`, 2023. [Online; accessed December 24, 2023].

[3] Amazon. Alexa. `https://www.alexa.com`, 2023. [Online; accessed December 26, 2023].

[4] Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. Mapping natural language instructions to mobile ui action sequences, 2020.

[5] Yuanchun Li and Oriana Riva. Glider: A reinforcement learning approach to extract ui scripts from websites. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 1420–1430, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380379. doi: 10.1145/3404835.3462905.

[6] Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. *ArXiv*, abs/1802.08802, 2018.

[7] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models, 2023.

[8] IBM. Ibm shoebox. `https://www.ibm.com/ibm/history/exhibits/specialprod1/specialprod1_7.html`, 2023. [Online; accessed December 26, 2023].

[9] Bruce Lowerre and R Reddy. The harpy speech recognition system: performance with large vocabularies. *The Journal of the Acoustical Society of America*, 60(S1):S10–S11, 1976.

[10] Helene Cerf-Danon, Steven DeGennaro, Marco Ferretti, Jorge Gonzalez, and Eric Keppel. 1. 0 TANGORA - a large vocabulary speech recognition system for five languages. In *Proc. 2nd European Conference on Speech Communication and Technology (Eurospeech 1991)*, pages 183–192, 1991. doi: 10.21437/Eurospeech.1991-44.

[11] L. Rabiner and B. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986. doi: 10.1109/MASSP.1986.1165342.

[12] Paul G. Bamberg, Yen lu Chow, Larry Gillick, Robert Roth, and Dean G. Sturtevant. The dragon continuous speech recognition system: A real-time implementation. In *Human Language Technology - The Baltic Perspectiv*, 1990.

[13] Wikipedia. Speakable items. `https://en.wikipedia.org/wiki/Speakable_items`, 2023. [Online; accessed January 5, 2023].

[14] Jennifer Lai and John Vergo. Medspeak: Report creation with continuous speech recognition. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, CHI '97, page 431–438, New York, NY, USA, 1997. Association for Computing Machinery. ISBN 0897918029. doi: 10.1145/258549.258829.

[15] Microsoft. Speech transcript - jim allchin, winhec 2002. `https://news.microsoft.com/speeches/speech-transcript-jim-allchin-winhec-2002/`, 2002. [Online; accessed January 5, 2023].

[16] John Markoff. Google is taking questions (spoken, via iphone). `https://www.nytimes.com/2008/11/14/technology/internet/14voice.html`, 2008. [Online; accessed January 5, 2024].

[17] Microsoft. Cortana. `https://www.microsoft.com/en-us/cortana`, 2023. [Online; accessed December 26, 2023].

[18] OpenAI. Introduce chatgpt. `https://openai.com/blog/chatgpt`, 2022. [Online; accessed November 28, 2023].

[19] Microsoft. Announcing microsoft copilot, your everyday ai companion. `https://blogs.microsoft.com/blog/2023/09/21/announcing-microsoft-copilot-your-everyday-ai-companion/`, 2023. [Online; accessed December 4, 2023].

[20] Apple. Sirikit: Empower users to interact with their devices through voice, intelligent suggestions, and personalized workflows. `https://developer.apple.com/documentation/sirikit/`, 2023. [Online; accessed December 24, 2023].

[21] Apple. Shortcuts user guide. `https://support.apple.com/en-hk/guide/shortcuts/welcome/ios`, 2023. [Online; accessed December 24, 2023].

[22] Joaoapps. Tasker: Total automation for android. `https://tasker.joaoapps.com`, 2023. [Online; accessed December 24, 2023].

[23] Absinthe. Anywhere shortcuts. `https://play.google.com/store/apps/details?id=com.absinthe.anywhere_&hl=en_US&pli=1`, 2023. [Online; accessed December 24, 2023].

[24] Toby Jia-Jun Li, Yuanchun Li, Fanglin Chen, and Brad A Myers. Programming iot devices by demonstration using mobile apps. In *End-User Development: 6th International Symposium, IS-EUD 2017, Eindhoven, The Netherlands, June 13-15, 2017, Proceedings 6*, pages 3–17. Springer, 2017.

[25] Tanzirul Azim, Oriana Riva, and Suman Nath. Ulink: Enabling user-defined deep linking to app content. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '16, page 305–318, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342698. doi: 10.1145/2906388.2906416.

[26] Benjamin R. Cowan, Nadia Pantidi, David Coyle, Kellie Morrissey, Peter Clarke, Sara Al-Shehri, David Earley, and Natasha Bandeira. "what can i help you with?": Infrequent users' experiences of intelligent personal assistants. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '17, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450350754. doi: 10.1145/3098279.3098539.

[27] Amanda Baughan, Xuezhi Wang, Ariel Liu, Allison Mercurio, Jilin Chen, and Xiao Ma. A mixed-methods approach to understanding user trust after voice assistant failures. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394215. doi: 10.1145/3544548.3581152.

[28] Ewa Luger and Abigail Sellen. "like having a really bad pa": The gulf between user expectation and experience of conversational agents. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, page 5286–5297, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450333627. doi: 10.1145/2858036.2858288.

[29] Matthew B. Hoy. Alexa, siri, cortana, and more: An introduction to voice assistants. *Medical Reference Services Quarterly*, 37(1):81–88, 2018. doi: 10.1080/02763869.2018.1404391. PMID: 29327988.

[30] Yuanchun Li, Ziyue Yang, Yao Guo, and Xiangqun Chen. Humanoid: A deep learning-based approach to automated black-box android app testing. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1070–1073. IEEE, 2019.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[32] Zecheng He, Srinivas Sunkara, Xiaoxue Zang, Ying Xu, Lijuan Liu, Nevan Wichers, Gabriel Schubiner, Ruby B. Lee, and Jindong Chen. Actionbert: Leveraging user actions for semantic understanding of user interfaces. In *AAAI Conference on Artificial Intelligence*, 2020.

[33] Yang Li, Gang Li, Xin Zhou, Mostafa Dehghani, and Alexey A. Gritsenko. Vut: Versatile ui transformer for multi-modal multi-task user interface modeling. *ArXiv*, abs/2112.05692, 2021.

[34] Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, and Blaise Agüera y Arcas. Uibert: Learning generic multimodal representations for ui understanding. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 1705–1712. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/235. Main Track.

[35] Gang Li and Yang Li. Spotlight: Mobile ui understanding using vision-language models with a focus. *ArXiv*, abs/2209.14927, 2022.

[36] Pratyay Banerjee, Shweti Mahajan, Kushal Arora, Chitta Baral, and Oriana Riva. Lexi: Self-supervised learning of the ui language. *ArXiv*, abs/2301.10165, 2023.

[37] Wei Li, Fu-Lin Hsu, Will Bishop, Folawiyo Campbell-Ajala, Oriana Riva, and Max Lin. Uinav: A maker of ui automation agents. *arXiv preprint arXiv:2312.10170*, 2023.

[38] Tianlin Tim Shi, Andrej Karpathy, Linxi Jim Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3135–3144. JMLR.org, 2017.

[39] Izzeddin Gur, Ulrich Rückert, Aleksandra Faust, and Dilek Z. Hakkani-Tür. Learning to navigate the web. *ArXiv*, abs/1812.09195, 2018.

[40] Sheng Jia, Jamie Ryan Kiros, and Jimmy Ba. Dom-q-net: Grounded rl on structured language. *ArXiv*, abs/1902.07257, 2019.

[41] Peter C Humphreys, David Raposo, Tobias Pohlen, Gregory Thornton, Rachita Chhaparia, Alistair Muldal, Josh Abramson, Petko Georgiev, Adam Santoro, and Timothy Lillicrap. A data-driven approach for learning to control computers. In *International Conference on Machine Learning*, pages 9466–9482. PMLR, 2022.

[42] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

[43] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

[44] Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2023.

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

[46] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. Webgpt: Browser-assisted question-answering with human feedback, 2022.

[47] Hiroki Furuta, Ofir Nachum, Kuang-Huei Lee, Yutaka Matsuo, Shixiang Shane Gu, and Izzeddin Gur. Multimodal web navigation with instruction-finetuned foundation models. *ArXiv*, abs/2305.11854, 2023.

[48] 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 *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE, 2023.

[49] Yue Zhen, Sheng Bi, Lu Xing-tong, Pan Wei-qin, Shi Hai-peng, Chen Zi-rui, and Fang Yi-shu. Robot task planning based on large language model representing knowledge with directed graph structures, 2023.

[50] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents, 2022.

[51] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face, 2023.

[52] Ke Wang, Houxing Ren, Aojun Zhou, Zimu Lu, Sichun Luo, Weikang Shi, Renrui Zhang, Linqi Song, Mingjie Zhan, and Hongsheng Li. Mathcoder: Seamless code integration in llms for enhanced mathematical reasoning. *ArXiv*, abs/2310.03731, 2023.

[53] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, I. Evtimov, Joanna Bitton, Manish P Bhatt, Cristian Cantón Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre D'efossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code. *ArXiv*, abs/2308.12950, 2023.

[54] Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, and Hongsheng Li. Solving challenging math word problems using gpt-4 code interpreter with code-based self-verification, 2023.

[55] OpenAI. Gpt-4 technical report, 2023.

[56] Microsoft. Reinventing search with a new ai-powered microsoft bing and edge, your copilot for the web. `https://blogs.microsoft.com/blog/2023/02/07/reinventing-search-with-a-new-ai-powered-microsoft-bing-and-edge-your-copilot-for-the-web/`, 2023. [Online; accessed December 8, 2023].

[57] Google. Bard: A conversational ai tool by google. `https://bard.google.com`, 2023. [Online; accessed December 26, 2023].

[58] Google. Introducing gemini: our largest and most capable ai model. `https://blog.google/technology/ai/google-gemini-ai/`, 2023. [Online; accessed December 26, 2023].

[59] Huawei. Reshaping industries with ai: Huawei cloud launches pangu models 3.0 and ascend ai cloud services. https://www.huaweicloud.com/intl/en-us/news/20230707180809498.html, 2023. [Online; accessed November 28, 2023].

[60] XiaoMi. Milm-6b. https://github.com/XiaoMi/MiLM-6B, 2023. [Online; accessed December 24, 2023].

[61] Sayed Naem Bokhari. The linux operating system. *Computer*, 28(8):74–79, 1995.

[62] Wikipedia. Borda count. https://en.wikipedia.org/wiki/Borda_count, 2023. [Online; accessed December 13, 2023].

[63] Jinyu Li. Recent advances in end-to-end automatic speech recognition, 2022.

[64] Rohit Prabhavalkar, Takaaki Hori, Tara N. Sainath, Ralf Schlüter, and Shinji Watanabe. End-to-end speech recognition: A survey, 2023.

[65] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A survey on large language model based autonomous agents, 2023.

[66] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. The rise and potential of large language model based agents: A survey, 2023.

[67] Zhuosheng Zhang, Yao Yao, Aston Zhang, Xiangru Tang, Xinbei Ma, Zhiwei He, Yiming Wang, Mark Gerstein, Rui Wang, Gongshen Liu, and Hai Zhao. Igniting language intelligence: The hitchhiker's guide from chain-of-thought reasoning to language agents, 2023.

[68] Steve Young, Milica Gašić, Blaise Thomson, and Jason D. Williams. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179, 2013. doi: 10.1109/JPROC.2012.2225812.

[69] Abhinav Rastogi, Raghav Gupta, and Dilek Hakkani-Tur. Multi-task learning for joint language understanding and dialogue state tracking. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 376–384, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5045.

[70] Toby Jia-Jun Li and Oriana Riva. Kite: Building conversational bots from mobile apps. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '18, page 96–109, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450357203. doi: 10.1145/3210240.3210339.

[71] Toby Jia-Jun Li, Amos Azaria, and Brad A. Myers. Sugilite: Creating multimodal smartphone automation by demonstration. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 6038–6049, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450346559. doi: 10.1145/3025453.3025483.

[72] Sang-Woo Lee, Sungdong Kim, Donghyeon Ko, Donghoon Ham, Youngki Hong, Shin Ah Oh, Hyunhoon Jung, Wangkyo Jung, Kyunghyun Cho, Donghyun Kwak, Hyungsuk Noh, and Woomyoung Park. Can current task-oriented dialogue models automate real-world scenarios in the wild?, 2023.

[73] Willy Chung, Samuel Cahyawijaya, Bryan Wilie, Holy Lovenia, and Pascale Fung. Instructtods: Large language models for end-to-end task-oriented dialogue systems, 2023.

[74] Zhiyuan Hu, Yue Feng, Yang Deng, Zekun Li, See-Kiong Ng, Anh Tuan Luu, and Bryan Hooi. Enhancing large language model induced task-oriented dialogue systems through look-forward motivated goals, 2023.

[75] Vojtěch Hudeček and Ondřej Dušek. Are llms all you need for task-oriented dialogue?, 2023.

[76] Zhiyuan Hu, Yue Feng, Anh Tuan Luu, Bryan Hooi, and Aldo Lipani. Unlocking the potential of user feedback: Leveraging large language model as user simulators to enhance dialogue system. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, page 3953–3957, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701245. doi: 10.1145/3583780.3615220.

[77] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[78] Microsoft. Bing web search api. `https://www.microsoft.com/en-us/bing/apis/bing-web-search-api`, 2023.

[79] Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023.

[80] Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. Gpt4tools: Teaching large language model to use tools via self-instruction, 2023.

[81] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis, 2023.

[82] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022.

[83] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

[84] Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, Dor Muhlgay, Noam Rozen, Erez Schwartz, Gal Shachaf, Shai Shalev-Shwartz, Amnon Shashua, and Moshe Tenenholtz. Mrkl systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning, 2022.

[85] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large scale language model society, 2023.

[86] Geunwoo Kim, Pierre Baldi, and Stephen Marcus McAleer. Language models can solve computer tasks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[87] Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. Chameleon: Plug-and-play compositional reasoning with large language models. In *The 37th Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[88] Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. ToolkenGPT: Augmenting frozen language models with massive tools via tool embeddings. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[89] Bryan Wang, Gang Li, and Yang Li. Enabling conversational interaction with mobile ui using large language models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394215. doi: 10.1145/3544548.3580895.

[90] Hao Wen, Hongming Wang, Jiaxuan Liu, and Yuanchun Li. Droidbot-gpt: Gpt-powered ui automation for android. *arXiv preprint arXiv:2304.07061*, 2023.

[91] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web, 2023.

[92] Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. Empowering llm to use smartphone for intelligent task automation, 2023.

[93] Maryam Taeb, Amanda Swearngin, Eldon Schoop, Ruijia Cheng, Yue Jiang, and Jeffrey Nichols. Axnav: Replaying accessibility tests from natural language, 2023.

[94] Sunjae Lee, Junyoung Choi, Jungjae Lee, Hojun Choi, Steven Y. Ko, Sangeun Oh, and Insik Shin. Explore, select, derive, and recall: Augmenting llm with human-like memory for mobile task automation. *arXiv preprint arXiv:2312.03003*, 2023.

[95] Liangtai Sun, Xingyu Chen, Lu Chen, Tianle Dai, Zichen Zhu, and Kai Yu. Meta-gui: Towards multi-modal conversational agents on mobile gui. *arXiv preprint arXiv:2205.11029*, 2022.

[96] Zhizheng Zhang, Xiaoyi Zhang, Wenxuan Xie, and Yan Lu. Responsible task automation: Empowering large language models as responsible task automators, 2023.

[97] Zhuosheng Zhan and Aston Zhang. You only look at screens: Multimodal chain-of-action agents. *arXiv preprint arXiv:2309.11436*, 2023.

[98] Zhizheng Zhang, Wenxuan Xie, Xiaoyi Zhang, and Yan Lu. Reinforced ui instruction grounding: Towards a generic ui task automation api. *ArXiv*, abs/2310.04716, 2023.

[99] An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian McAuley, Jianfeng Gao, et al. Gpt-4v in wonderland: Large multimodal models for zero-shot smartphone gui navigation. *arXiv preprint arXiv:2311.07562*, 2023.

[100] Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxuan Zhang, Juanzi Li, Bin Xu, Yuxiao Dong, Ming Ding, and Jie Tang. Cogagent: A visual language model for gui agents, 2023.

[101] Sijie Cheng, Zhicheng Guo, Jingwen Wu, Kechen Fang, Peng Li, Huaping Liu, and Yang Liu. Can vision-language models think from a first-person perspective?, 2023.

[102] Lilian Weng. Llm powered autonomous agents. `https://lilianweng.github.io/posts/2023-06-23-agent/`, 2023.

[103] Autogpt. `https://github.com/Significant-Gravitas/AutoGPT`, 2023.

[104] Langchain. `https://github.com/langchain-ai/langchain`, 2023.

[105] Babyagi. `https://github.com/yoheinakajima/babyagi`, 2023.

[106] Anton Osika. Gpt-engineer. `https://github.com/AntonOsika/gpt-engineer`, 2023.

[107] Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Sesay Jaward, Karlsson Börje, Jie Fu, and Yemin Shi. Autoagents: The automatic agents generation framework. *arXiv preprint*, 2023.

[108] Tianbao Xie, Fan Zhou, Zhoujun Cheng, Peng Shi, Luoxuan Weng, Yitao Liu, Toh Jing Hua, Junning Zhao, Qian Liu, Che Liu, Leo Z. Liu, Yiheng Xu, Hongjin Su, Dongchan Shin, Caiming Xiong, and Tao Yu. Openagents: An open platform for language agents in the wild, 2023.

[109] KillianLucas. Open interpreter. `https://github.com/KillianLucas/open-interpreter`, 2023.

[110] Jerry Liu. LlamaIndex, 11 2022. URL `https://github.com/jerryjliu/llama_index`.

[111] Deshraj Yadav Taranjeet Singh. Embedchain: Data platform for llms - load, index, retrieve, and sync any unstructured data. `https://github.com/embedchain/embedchain`, 2023.

[112] Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, Shi Qiu, Jintian Zhang, Jing Chen, Ruipu Wu, Shuai Wang, Shiding Zhu, Jiyu Chen, Wentao Zhang, Ningyu Zhang, Huajun Chen, Peng Cui, and Mrinmaya Sachan. Agents: An open-source framework for autonomous language agents, 2023.

[113] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework, 2023.

[114] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, 2023.

[115] Forrest Huang, Gang Li, Tao Li, and Yang Li. Automatic macro mining from interaction traces at scale, 2023.

[116] Daniel Toyama, Philippe Hamel, Anita Gergely, Gheorghe Comanici, Amelia Glaese, Zafarali Ahmed, Tyler Jackson, Shibl Mourad, and Doina Precup. Androidenv: A reinforcement learning platform for android. *arXiv preprint arXiv:2105.13231*, 2021.

[117] Danyang Zhang, Lu Chen, Zihan Zhao, Ruisheng Cao, and Kai Yu. Mobile-Env: An evaluation platform and benchmark for interactive agents in llm era. *CoRR*, abs/2305.08144, 2023.

[118] Panupong Pasupat, Tian-Shun Jiang, Evan Liu, Kelvin Guu, and Percy Liang. Mapping natural language commands to web elements. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4970–4976, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1540.

[119] Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, and Bryan A. Plummer. A dataset for interactive vision language navigation with unknown command feasibility. In *European Conference on Computer Vision (ECCV)*, 2022.

[120] Sagar Gubbi Venkatesh, Partha Talukdar, and Srini Narayanan. Ugif: Ui grounded instruction following, 2023.

[121] Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Android in the wild: A large-scale dataset for android device control, 2023.

[122] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pages 20744–20757. Curran Associates, Inc., 2022.

[123] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

[124] Difei Gao, Lei Ji, Zechen Bai, Mingyu Ouyang, Peiran Li, Dongxing Mao, Qinchen Wu, Weichen Zhang, Peiyi Wang, Xiangwu Guo, Hengxu Wang, Luowei Zhou, and Mike Zheng Shou. Assistgui: Task-oriented desktop graphical user interface automation, 2023.

[125] Joseph Breda, Mastafa Springston, Alex Mariakakis, and Shwetak Patel. Feverphone: Accessible core-body temperature sensing for fever monitoring using commodity smartphones. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 7(1):1–23, 2023.

[126] Bhawana Chhaglani, Camellia Zakaria, Adam Lechowicz, Jeremy Gummeson, and Prashant Shenoy. Flowsense: Monitoring airflow in building ventilation systems using audio sensing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(1):1–26, 2022.

[127] Yongquan Hu, Hui-Shyong Yeo, Mingyue Yuan, Haoran Fan, Don Samitha Elvitigala, Wen Hu, and Aaron Quigley. Microcam: Leveraging smartphone microscope camera for context-aware contact surface sensing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 7(3):1–28, 2023.

[128] Jingzhi Hu, Tianyue Zheng, Zhe Chen, Hongbo Wang, and Jun Luo. Muse-fi: Contactless muti-person sensing exploiting near-field wi-fi channel variation. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pages 1–15, 2023.

[129] Jian Gong, Xinyu Zhang, Yuanjun Huang, Ju Ren, and Yaoxue Zhang. Robust inertial motion tracking through deep sensor fusion across smart earbuds and smartphone. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(2):1–26, 2021.

[130] Luca Arrotta, Gabriele Civitarese, and Claudio Bettini. Dexar: Deep explainable sensor-based activity recognition in smart-home environments. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(1):1–30, 2022.

[131] Haoyang Wen, Zhenxin Xiao, Eduard Hovy, and Alexander G Hauptmann. Towards open-domain twitter user profile inference. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3172–3188, 2023.

[132] Filippo Maria Bianchi, Antonello Rizzi, Alireza Sadeghian, and Corrado Moiso. Identifying user habits through data mining on call data records. *Engineering Applications of Artificial Intelligence*, 54:49–61, 2016.

[133] Jaemin Shin, Hyungjun Yoon, Seungjoo Lee, Sungjoon Park, Yunxin Liu, Jinho D Choi, and Sung-Ju Lee. Fedtherapist: Mental health monitoring with user-generated linguistic expressions on smartphones via federated learning. *arXiv preprint arXiv:2310.16538*, 2023.

[134] Rafael Wampfler, Severin Klingler, Barbara Solenthaler, Victor R Schinazi, Markus Gross, and Christian Holz. Affective state prediction from smartphone touch and sensor data in the wild. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2022.

[135] Yu-Chun Chen, Yu-Jen Lee, Kuei-Chun Kao, Jie Tsai, En-Chi Liang, Wei-Chen Chiu, Faye Shih, and Yung-Ju Chang. Are you killing time? predicting smartphone users' time-killing moments via fusion of smartphone sensor data and screenshots. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–19, 2023.

[136] Tousif Ahmed, Md Mahbubur Rahman, Ebrahim Nemati, Mohsin Yusuf Ahmed, Jilong Kuang, and Alex Jun Gao. Remote breathing rate tracking in stationary position using the motion and acoustic sensors of earables. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–22, 2023.

[137] Vimal Mollyn, Karan Ahuja, Dhruv Verma, Chris Harrison, and Mayank Goel. Samosa: Sensing activities with motion and subsampled audio. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(3):1–19, 2022.

[138] Elena Di Lascio, Shkurta Gashi, Juan Sebastian Hidalgo, Beatrice Nale, Maike E Debus, and Silvia Santini. A multi-sensor approach to automatically recognize breaks and work activities of knowledge workers in academia. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(3):1–20, 2020.

[139] Minhao Cui, Binbin Xie, Qing Wang, and Jie Xiong. Dancingant: Body-empowered wireless sensing utilizing pervasive radiations from powerline. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pages 1–15, 2023.

[140] Yinghui He, Jianwei Liu, Mo Li, Guanding Yu, Jinsong Han, and Kui Ren. Sencom: Integrated sensing and communication with practical wifi. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2023.

[141] Camellia Zakaria, Gizem Yilmaz, Priyanka Mary Mammen, Michael Chee, Prashant Shenoy, and Rajesh Balan. Sleepmore: Inferring sleep duration at scale via multi-device wifi sensing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(4):1–32, 2023.

[142] Nan Gao, Zhuolei Yu, Chun Yu, Yuntao Wang, Flora D Salim, and Yuanchun Shi. Automated mobile sensing strategies generation for human behaviour understanding. *arXiv preprint arXiv:2311.05457*, 2023.

[143] Sirat Samyoun, Md Mofijul Islam, Tariq Iqbal, and John Stankovic. M3sense: Affect-agnostic multitask representation learning using multimodal wearable sensors. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(2):1–32, 2022.

[144] Shohreh Deldari, Hao Xue, Aaqib Saeed, Daniel V Smith, and Flora D Salim. Cocoa: Cross modality contrastive learning for sensor data. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(3):1–28, 2022.

[145] Alireza Abedin, Mahsa Ehsanpour, Qinfeng Shi, Hamid Rezatofighi, and Damith C Ranasinghe. Attend and discriminate: Beyond the state-of-the-art for human activity recognition using wearable sensors. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(1):1–22, 2021.

[146] Haroon Rashid, Sanjana Mendu, Katharine E Daniel, Miranda L Beltzer, Bethany A Teachman, Mehdi Boukhechba, and Laura E Barnes. Predicting subjective measures of social anxiety from sparsely collected mobile sensor data. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4 (3):1–24, 2020.

[147] Jeong-Kyun Kim, Da-Som Oh, Kangbok Lee, and Sang Gi Hong. Fall detection based on interpretation of important features with wrist-wearable sensors. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pages 823–825, 2022.

[148] Kaikai Liu, Xinxin Liu, and Xiaolin Li. Guoguo: Enabling fine-grained indoor localization via smartphone. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pages 235–248, 2013.

[149] Selina Chu, Shrikanth Narayanan, and C-C Jay Kuo. Environmental sound recognition with time–frequency audio features. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(6):1142–1158, 2009.

[150] S Chandrakala and SL Jayalakshmi. Environmental audio scene and sound event recognition for autonomous surveillance: A survey and comparative studies. *ACM Computing Surveys (CSUR)*, 52(3):1–34, 2019.

[151] Karim Assi, Lakmal Meegahapola, William Droz, Peter Kun, Amalia De Götzen, Miriam Bidoglia, Sally Stares, George Gaskell, Altangerel Chagnaa, Amarsanaa Ganbold, et al. Complex daily activities, country-level diversity, and smartphone sensing: A study in denmark, italy, mongolia, paraguay, and uk. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–23, 2023.

[152] Lakmal Meegahapola, William Droz, Peter Kun, Amalia De Götzen, Chaitanya Nutakki, Shyam Diwakar, Salvador Ruiz Correa, Donglei Song, Hao Xu, Miriam Bidoglia, et al. Generalization and personalization of mobile sensing-based mood inference models: An analysis of college students in eight countries. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(4):1–32, 2023.

[153] Zhiyuan Wang, Maria A Larrazabal, Mark Rucker, Emma R Toner, Katharine E Daniel, Shashwat Kumar, Mehdi Boukhechba, Bethany A Teachman, and Laura E Barnes. Detecting social contexts from mobile sensing indicators in virtual interactions with socially anxious individuals. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 7(3):1–26, 2023.

[154] Lakmal Meegahapola, Florian Labhart, Thanh-Trung Phan, and Daniel Gatica-Perez. Examining the social context of alcohol drinking in young adults with smartphone sensing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(3):1–26, 2021.

[155] Lakmal Meegahapola, Salvador Ruiz-Correa, Viridiana del Carmen Robledo-Valero, Emilio Ernesto Hernandez-Huerfano, Leonardo Alvarez-Rivera, Ronald Chenu-Abente, and Daniel Gatica-Perez. One more bite? inferring food consumption level of college students using smartphone sensing and self-reports. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(1):1–28, 2021.

[156] Yuebing Liang, Yichao Liu, Xiaohan Wang, and Zhan Zhao. Exploring large language models for human mobility prediction under public events. *arXiv preprint arXiv:2311.17351*, 2023.

[157] Huatao Xu, Liying Han, Mo Li, and Mani Srivastava. Penetrative ai: Making llms comprehend the physical world. *arXiv preprint arXiv:2310.09605*, 2023.

[158] Xing Su, Hanghang Tong, and Ping Ji. Activity recognition with smartphone sensors. *Tsinghua science and technology*, 19(3):235–249, 2014.

[159] Sayma Akther, Nazir Saleheen, Mithun Saha, Vivek Shetty, and Santosh Kumar. mteeth: Identifying brushing teeth surfaces using wrist-worn inertial sensors. *Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies*, 5(2):1–25, 2021.

[160] Yetong Cao, Fan Li, Huijie Chen, Xiaochen liu, Li Zhang, and Yu Wang. Guard your heart silently: Continuous electrocardiogram waveform monitoring with wrist-worn motion sensor. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(3):1–29, 2022.

[161] Zongyu Lin, Shiqing Lyu, Hancheng Cao, Fengli Xu, Yuqiong Wei, Hanan Samet, and Yong Li. Healthwalks: Sensing fine-grained individual health condition via mobility data. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(4):1–26, 2020.

[162] Xiao Zhang, Wenzhong Li, Xu Chen, and Sanglu Lu. Moodexplorer: Towards compound emotion detection via smartphone sensing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1 (4):1–30, 2018.

[163] Daniel A Adler, Vincent W-S Tseng, Gengmo Qi, Joseph Scarpa, Srijan Sen, and Tanzeem Choudhury. Identifying mobile sensing indicators of stress-resilience. *Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies*, 5(2):1–32, 2021.

[164] Weichen Wang, Subigya Nepal, Jeremy F Huckins, Lessley Hernandez, Vlado Vojdanovski, Dante Mack, Jane Plomp, Arvind Pillai, Mikio Obuchi, Alex Dasilva, et al. First-gen lens: Assessing mental health of first-generation students across their first year at college using mobile sensing. *Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies*, 6(2):1–32, 2022.

[165] Rui Wang, Gabriella Harari, Peilin Hao, Xia Zhou, and Andrew T Campbell. Smartgpa: how smartphones can assess and predict academic performance of college students. In *Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing*, pages 295–306, 2015.

[166] Nan Gao, Wei Shao, and Flora D Salim. Predicting personality traits from physical activity intensity. *Computer*, 52(7):47–56, 2019.

[167] Subigya Nepal, Shayan Mirjafari, Gonzalo J Martinez, Pino Audia, Aaron Striegel, and Andrew T Campbell. Detecting job promotion in information workers using mobile sensing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(3):1–28, 2020.

[168] Onur Yürüten, Jiyong Zhang, and Pearl HZ Pu. Predictors of life satisfaction based on daily activities from mobile sensor data. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 497–500, 2014.

[169] Weichen Wang, Shayan Mirjafari, Gabriella Harari, Dror Ben-Zeev, Rachel Brian, Tanzeem Choudhury, Marta Hauser, John Kane, Kizito Masaba, Subigya Nepal, et al. Social sensing: assessing social functioning of patients living with schizophrenia using mobile phone sensing. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, pages 1–15, 2020.

[170] Rui Wang, Weichen Wang, Min SH Aung, Dror Ben-Zeev, Rachel Brian, Andrew T Campbell, Tanzeem Choudhury, Marta Hauser, John Kane, Emily A Scherer, et al. Predicting symptom trajectories of schizophrenia using mobile sensing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1 (3):1–24, 2017.

[171] Soujanya Chatterjee, Alexander Moreno, Steven Lloyd Lizotte, Sayma Akther, Emre Ertin, Christopher P Fagundes, Cho Lam, James M Rehg, Neng Wan, David W Wetter, et al. Smokingopp: Detecting the smoking'opportunity'context using mobile sensors. *Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies*, 4(1):1–26, 2020.

[172] Zheng Chen. Palr: Personalization aware llms for recommendation. *arXiv preprint arXiv:2305.07622*, 2023.

[173] Wenxuan Zhang, Hongzhi Liu, Yingpeng Du, Chen Zhu, Yang Song, Hengshu Zhu, and Zhonghai Wu. Bridging the information gap between domain-specific model and general llm for personalized recommendation. *arXiv preprint arXiv:2311.03778*, 2023.

[174] Xiaofei Sun, Xiaoya Li, Shengyu Zhang, Shuhe Wang, Fei Wu, Jiwei Li, Tianwei Zhang, and Guoyin Wang. Sentiment analysis through llm negotiations. *arXiv preprint arXiv:2311.01876*, 2023.

[175] Mahyar Abbasian, Iman Azimi, Amir M Rahmani, and Ramesh Jain. Conversational health agents: A personalized llm-powered agent framework. *arXiv preprint arXiv:2310.02374*, 2023.

[176] Cathal Gurrin, Alan F Smeaton, Aiden R Doherty, et al. Lifelogging: Personal big data. *Foundations and Trends® in information retrieval*, 8(1):1–125, 2014.

[177] Martin Dodge and Rob Kitchin. 'outlines of a world coming into existence': pervasive computing and the ethics of forgetting. *Environment and planning B: planning and design*, 34(3):431–445, 2007.

[178] Djamila Romaissa Beddiar, Brahim Nini, Mohammad Sabokrou, and Abdenour Hadid. Vision-based human activity recognition: a survey. *Multimedia Tools and Applications*, 79(41-42):30509–30555, 2020.

[179] Clemens Stachl, Quay Au, Ramona Schoedel, Samuel D Gosling, Gabriella M Harari, Daniel Buschek, Sarah Theres Völkel, Tobias Schuwerk, Michelle Oldemeier, Theresa Ullmann, et al. Predicting personality from patterns of behavior collected with smartphones. *Proceedings of the National Academy of Sciences*, 117 (30):17680–17687, 2020.

[180] Navonil Majumder, Soujanya Poria, Alexander Gelbukh, and Erik Cambria. Deep learning-based document modeling for personality detection from text. *IEEE Intelligent Systems*, 32(2):74–79, 2017.

[181] Sanja Štajner and Seren Yenikent. A survey of automatic personality detection from texts. In *Proceedings of the 28th international conference on computational linguistics*, pages 6284–6295, 2020.

[182] Akriti Jaiswal, A Krishnama Raju, and Suman Deb. Facial emotion detection using deep learning. In *2020 international conference for emerging technology (INCET)*, pages 1–5. IEEE, 2020.

[183] Samira Zad, Maryam Heidari, H James Jr, and Ozlem Uzuner. Emotion detection of textual data: An interdisciplinary survey. In *2021 IEEE World AI IoT Congress (AIIoT)*, pages 0255–0261. IEEE, 2021.

[184] Xiaoli Tang, Tengyun Wang, Haizhi Yang, and Hengjie Song. Akupm: Attention-enhanced knowledge-aware user preference model for recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1891–1899, 2019.

[185] Yuanchun Li, Ziyue Yang, Yao Guo, Xiangqun Chen, Yuvraj Agarwal, and Jason I Hong. Automated extraction of personal knowledge from smartphone push notifications. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 733–742. IEEE, 2018.

[186] Garima Singh and Arun Solanki. An algorithm to transform natural language into sql queries for relational databases. *Selforganizology*, 3(3):100–116, 2016.

[187] Kevin Lin, Ben Bogin, Mark Neumann, Jonathan Berant, and Matt Gardner. Grammar-based neural text-to-sql generation. *arXiv preprint arXiv:1905.13326*, 2019.

[188] Yuanchun Li, Fanglin Chen, Toby Jia-Jun Li, Yao Guo, Gang Huang, Matthew Fredrikson, Yuvraj Agarwal, and Jason I Hong. Privacystreams: Enabling transparency in personal data processing for mobile apps. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(3):76, 2017.

[189] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22, 2023.

[190] Xiaonan Li and Xipeng Qiu. Mot: Memory-of-thought enables chatgpt to self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6354–6374, 2023.

[191] Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. *arXiv preprint arXiv:2306.07174*, 2023.

[192] Zhicheng Guo, Sijie Cheng, Yile Wang, Peng Li, and Yang Liu. Prompt-guided retrieval augmentation for non-knowledge-intensive tasks. *arXiv preprint arXiv:2305.17653*, 2023.

[193] Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

[194] Theodore Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L Griffiths. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427*, 2023.

[195] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[196] Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*, 2023.

[197] Jens Tuyls, Shunyu Yao, Sham Kakade, and Karthik Narasimhan. Multi-stage episodic control for strategic exploration in text games. *arXiv preprint arXiv:2201.01251*, 2022.

[198] Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. Keep calm and explore: Language models for action generation in text-based games. *arXiv preprint arXiv:2010.02903*, 2020.

[199] 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. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR, 2022.

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

[201] Wenjia Joyce Zhao, Russell Richie, and Sudeep Bhatia. Process and content in decisions from memory. *Psychological Review*, 129(1):73, 2022.

[202] Austin W Hanjie, Victor Y Zhong, and Karthik Narasimhan. Grounding language to entities and dynamics for generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 4051–4062. PMLR, 2021.

[203] Meenal Parakh, Alisha Fong, Anthony Simeonov, Abhishek Gupta, Tao Chen, and Pulkit Agrawal. Human-assisted continual robot learning with foundation models. *arXiv preprint arXiv:2309.14321*, 2023.

[204] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[205] Kevin Ellis, Lionel Wong, Maxwell Nye, Mathias Sable-Meyer, Luc Cary, Lore Anaya Pozo, Luke Hewitt, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: growing generalizable, interpretable knowledge with wake–sleep bayesian program learning. *Philosophical Transactions of the Royal Society A*, 381(2251): 20220050, 2023.

[206] Jesse Zhang, Jiahui Zhang, Karl Pertsch, Ziyi Liu, Xiang Ren, Minsuk Chang, Shao-Hua Sun, and Joseph J Lim. Bootstrap your own skills: Learning to solve new tasks with large language model guidance. *arXiv preprint arXiv:2310.10021*, 2023.

[207] Xisen Jin, Dejiao Zhang, Henghui Zhu, Wei Xiao, Shang-Wen Li, Xiaokai Wei, Andrew Arnold, and Xiang Ren. Lifelong pretraining: Continually adapting language models to emerging corpora. *arXiv preprint arXiv:2110.08534*, 2021.

[208] Natawut Monaikul, Giuseppe Castellucci, Simone Filice, and Oleg Rokhlenko. Continual learning for named entity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13570–13577, 2021.

[209] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, et al. Tool learning with foundation models. *arXiv preprint arXiv:2304.08354*, 2023.

[210] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

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

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

[213] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`, 2022.

[214] Yaqing Wang, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. Adamix: Mixture-of-adapter for parameter-efficient tuning of large language models. *arXiv preprint arXiv:2205.12410*, 1(2):4, 2022.

[215] Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*, 2023.

[216] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

[217] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

[218] Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*, 2023.

[219] Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183, 2022.

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

[221] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *arXiv preprint arXiv:2305.11627*, 2023.

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

[223] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

[224] Inar Timiryasov and Jean-Loup Tastet. Baby llama: knowledge distillation from an ensemble of teachers trained on a small dataset with no performance penalty. *arXiv preprint arXiv:2308.02019*, 2023.

[225] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*, 2023.

[226] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*, 2023.

[227] Liunian Harold Li, Jack Hessel, Youngjae Yu, Xiang Ren, Kai-Wei Chang, and Yejin Choi. Symbolic chain-of-thought distillation: Small models can also" think" step-by-step. *arXiv preprint arXiv:2306.14050*, 2023.

[228] Zhewei Yao, Xiaoxia Wu, Cheng Li, Stephen Youn, and Yuxiong He. Zeroquant-v2: Exploring post-training quantization in llms from comprehensive study to low rank compensation, 2023.

[229] Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. Losparse: Structured compression of large language models based on low-rank and sparse approximation. *arXiv preprint arXiv:2306.11222*, 2023.

[230] Yucheng Li, Bo Dong, Chenghua Lin, and Frank Guerin. Compressing context to enhance inference efficiency of large language models, 2023.

[231] Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023)*, December 2023.

[232] Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. *ArXiv*, abs/2305.14788, 2023.

[233] Sotiris Anagnostidis, Dario Pavllo, Luca Biggio, Lorenzo Noci, Aurelien Lucchi, and Thomas Hoffmann. Dynamic context pruning for efficient and interpretable autoregressive transformers. *arXiv preprint arXiv:2305.15805*, 2023.

[234] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *arXiv preprint arXiv:2306.14048*, 2023.

[235] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[236] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

[237] Ke Hong, Guohao Dai, Jiaming Xu, Qiuli Mao, Xiuhong Li, Jun Liu, Kangdi Chen, Hanyu Dong, and Yu Wang. Flashdecoding++: Faster large language model inference on gpus. *arXiv preprint arXiv:2311.01282*, 2023.

[238] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

[239] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.

[240] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Daniel Y. Fu, Zhiqiang Xie, Beidi Chen, Clark Barrett, Joseph E. Gonzalez, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. Flexgen: High-throughput generative inference of large language models with a single gpu, 2023.

[241] Qualcomm. Snapdragon 8 gen 3 mobile platform. https://www.qualcomm.com/products/mobile/snapdragon/smartphones/snapdragon-8-series-mobile-platforms/snapdragon-8-gen-3-mobile-platform, 2023.

[242] Brendan C Reidy, Mohammadreza Mohammadi, Mohammed E Elbtity, and Ramtin Zand. Efficient deployment of transformer models on edge tpu accelerators: A real system evaluation. In *Architecture and System Support for Transformer Models (ASSYST@ ISCA 2023)*, 2023.

[243] Seongmin Hong, Seungjae Moon, Junsoo Kim, Sungjae Lee, Minsub Kim, Dongsoo Lee, and Joo-Young Kim. Dfx: A low-latency multi-fpga appliance for accelerating transformer-based text generation. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 616–630. IEEE, 2022.

[244] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. *CoRR*, abs/1902.00751, 2019.

[245] Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models, 2023.

[246] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[247] Kai Lv, Yuqing Yang, Tengxiao Liu, Qinghui Gao, Qipeng Guo, and Xipeng Qiu. Full parameter fine-tuning for large language models with limited resources, 2023.

[248] Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training, 2023.

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

[250] Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: phi-1.5 technical report, 2023.

[251] Mojan Javaheripi and Sébastien Bubeck. Phi-2: The surprising power of small language models. https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/, 2023.

[252] Yuhan Liu, Hanchen Li, Kuntai Du, Jiayi Yao, Yihua Cheng, Yuyang Huang, Shan Lu, Michael Maire, Henry Hoffmann, Ari Holtzman, Ganesh Ananthanarayanan, and Junchen Jiang. Cachegen: Fast context loading for language model applications, 2023.

[253] Qdrant team. Qdrant. https://github.com/qdrant/qdrant, 2021.

[254] Vespa.ai team. Vespa. https://github.com/vespa-engine/vespa, 2016.

[255] Chuangxian Wei, Bin Wu, Sheng Wang, Renjie Lou, Chaoqun Zhan, Feifei Li, and Yuanzhe Cai. Analyticdb-v: A hybrid analytical engine towards query fusion for structured and unstructured data. *Proc. VLDB Endow.*, 13 (12):3152–3165, aug 2020. ISSN 2150-8097. doi: 10.14778/3415478.3415541.

[256] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*, SIGMOD '21, page 2614–2627, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383431. doi: 10.1145/3448016.3457550.

[257] Wei Wu, Junlin He, Yu Qiao, Guoheng Fu, Li Liu, and Jin Yu. Hqann: Efficient and robust similarity search for hybrid queries with structured and unstructured constraints. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, page 4580–4584, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392365. doi: 10.1145/3511808.3557610.

[258] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[259] Fabien Andre, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. Quicker adc: Unlocking the hidden potential of product quantization with simd. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5): 1666–1677, May 2021. ISSN 1939-3539. doi: 10.1109/tpami.2019.2952606.

[260] Vald team. Vald. `https://github.com/vdaas/vald`, 2019.

[261] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, page 253–262, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138857. doi: 10.1145/997817.997857.

[262] Sanjoy Dasgupta and Yoav Freund. Random projection trees and low dimensional manifolds. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, page 537–546, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605580470. doi: 10.1145/1374376.1374452.

[263] Sanjoy Dasgupta and Kaushik Sinha. Randomized partition trees for exact nearest neighbor search, 2013.

[264] Qi Chen, Bing Zhao, Haidong Wang, Mingqin Li, Chuanjie Liu, Zengzhong Li, Mao Yang, and Jingdong Wang. SPANN: Highly-efficient billion-scale approximate nearest neighborhood search. In *Advances in Neural Information Processing Systems*, 2021.

[265] Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. Approximate nearest neighbor algorithm based on navigable small world graphs. *Inf. Syst.*, 45:61–68, 2014.

[266] Yu A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(4):824–836, apr 2020. ISSN 0162-8828. doi: 10.1109/TPAMI.2018.2889473.

[267] Suhas Jayaram Subramanya, Fnu Devvrit, Harsha Vardhan Simhadri, Ravishankar Krishnawamy, and Rohan Kadekodi. Diskann: Fast accurate billion-point nearest neighbor search on a single node. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[268] Jiongkang Ni, Xiaoliang Xu, Yuxiang Wang, Can Li, Jiajie Yao, Shihai Xiao, and Xuecang Zhang. Diskann++: Efficient page-based search over isomorphic mapped graph index using query-sensitivity entry vertex. *ArXiv*, abs/2310.00402, 2023.

[269] Junhyeok Jang, Hanjin Choi, Hanyeoreum Bae, Seungjun Lee, Miryeong Kwon, and Myoungsoo Jung. Cxl-anns: Software-hardware collaborative memory disaggregation and computation for billion-scale approximate nearest neighbor search. In *USENIX Annual Technical Conference*, 2023.

[270] Wenqi Jiang, Shigang Li, Yu Zhu, Johannes de Fine Licht, Zhenhao He, Runbin Shi, Cédric Renggli, Shuai Zhang, Theodoros Rekatsinas, Torsten Hoefler, and Gustavo Alonso. Co-design hardware and algorithm for vector search. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2023.

[271] Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. Understanding and overcoming the challenges of efficient transformer quantization. *arXiv preprint arXiv:2109.12948*, 2021.

[272] Xiuying Wei, Yunchen Zhang, Xiangguo Zhang, Ruihao Gong, Shanghang Zhang, Qi Zhang, Fengwei Yu, and Xianglong Liu. Outlier suppression: Pushing the limit of low-bit transformer language models. *Advances in Neural Information Processing Systems*, 35:17402–17414, 2022.

[273] llama.cpp developers. ggerganov/llama.cpp: Port of facebook's llama model in c/c++. `https://github.com/ggerganov/llama.cpp`, 2023.

[274] MLC team. MLC-LLM, 2023. URL `https://github.com/mlc-ai/mlc-llm`.

[275] Zhihang Yuan, Lin Niu, Jiawei Liu, Wenyu Liu, Xinggang Wang, Yuzhang Shang, Guangyu Sun, Qiang Wu, Jiaxiang Wu, and Bingzhe Wu. Rptq: Reorder-based post-training quantization for large language models. *arXiv preprint arXiv:2304.01089*, 2023.

[276] Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling. *arXiv preprint arXiv:2304.09145*, 2023.

[277] Jing Liu, Ruihao Gong, Xiuying Wei, Zhiwei Dong, Jianfei Cai, and Bohan Zhuang. Qllm: Accurate and efficient low-bitwidth quantization for large language models. *arXiv preprint arXiv:2310.08041*, 2023.

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

[279] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.

[280] Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *arXiv preprint arXiv:2305.17118*, 2023.

[281] Joshua Ainslie, Tao Lei, Michiel de Jong, Santiago Ontan'on, Siddhartha Brahma, Yury Zemlyanskiy, David C. Uthus, Mandy Guo, James Lee-Thorp, Yi Tay, Yun-Hsuan Sung, and Sumit K. Sanghai. Colt5: Faster long-range transformers with conditional computation. In *Conference on Empirical Methods in Natural Language Processing*, 2023.

[282] Luciano Del Corro, Allie Del Giorno, Sahaj Agarwal, Bin Yu, Ahmed Awadallah, and Subhabrata Mukherjee. Skipdecode: Autoregressive skip decoding with batching and caching for efficient llm inference. *arXiv preprint arXiv:2307.02628*, 2023.

[283] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *ArXiv*, abs/2006.04768, 2020.

[284] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Rae Ying Yee Wong, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. Specinfer: Accelerating generative llm serving with speculative inference and token tree verification. *arXiv preprint arXiv:2305.09781*, 2023.

[285] Benjamin Spector and Chris Re. Accelerating llm inference with staged speculative decoding. *arXiv preprint arXiv:2308.04623*, 2023.

[286] Sehoon Kim, Karttikeya Mangalam, Suhong Moon, Jitendra Malik, Michael W Mahoney, Amir Gholami, and Kurt Keutzer. Speculative decoding with big little decoder. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[287] Wenhua Ye, Xu Zhou, Joey Zhou, Cen Chen, and Kenli Li. Accelerating attention mechanism on fpgas based on efficient reconfigurable systolic array. *ACM Transactions on Embedded Computing Systems*, 22(6):1–22, 2023.

[288] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning, 2022.

[289] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation, 2021.

[290] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks, 2022.

[291] Renrui Zhang, Jiaming Han, Chris Liu, Peng Gao, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, and Yu Qiao. Llama-adapter: Efficient fine-tuning of language models with zero-init attention, 2023.

[292] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too, 2023.

[293] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[294] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1188–1196, Bejing, China, 22–24 Jun 2014. PMLR.

[295] Jiongnan Liu, Jiajie Jin, Zihan Wang, Jiehan Cheng, Zhicheng Dou, and Ji-Rong Wen. Reta-llm: A retrieval-augmented large language model toolkit, 2023.

[296] Eric Melz. Enhancing llm intelligence with arm-rag: Auxiliary rationale memory for retrieval augmented generation, 2023.

[297] Zexuan Zhong, Tao Lei, and Danqi Chen. Training language models with memory augmentation. *ArXiv*, abs/2205.12674, 2022. URL `https://api.semanticscholar.org/CorpusID:249062699`.

[298] Yikun Han, Chunjiang Liu, and Pengfei Wang. A comprehensive survey on vector database: Storage and retrieval technique, challenge. *arXiv preprint arXiv:2310.11703*, 2023.

[299] James Jie Pan, Jianguo Wang, and Guoliang Li. Survey of vector database management systems, 2023.

[300] Toni Taipalus. Vector database management systems: Fundamental concepts, use-cases, and current challenges. *ArXiv*, abs/2309.11322, 2023.

[301] Yuhuai Wu, Markus N. Rabe, DeLesley S. Hutchins, and Christian Szegedy. Memorizing transformers. *ArXiv*, abs/2203.08913, 2022.

[302] Ali Modarressi, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. Ret-llm: Towards a general read-write memory for large language models. *arXiv preprint arXiv:2305.14322*, 2023.

[303] Yao Tian, Ziyang Yue, Ruiyuan Zhang, Xi Zhao, Bolong Zheng, and Xiaofang Zhou. Approximate nearest neighbor search in high dimensional vector databases: Current research and future directions, 2023.

[304] Siddharth Gollapudi, Neel Karia, Varun Sivashankar, Ravishankar Krishnaswamy, Nikit Begwani, Swapnil Raz, Yiyong Lin, Yin Zhang, Neelam Mahapatro, Premkumar Srinivasan, Amit Singh, and Harsha Vardhan Simhadri. Filtered-diskann: Graph algorithms for approximate nearest neighbor search with filters. In *Proceedings of the ACM Web Conference 2023*, WWW '23, page 3406–3416, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394161. doi: 10.1145/3543507.3583552.

[305] Weijie Zhao, Shulong Tan, and Ping Li. Song: Approximate nearest neighbor search on gpu. *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1033–1044, 2020.

[306] Fabian Groh, Lukas Ruppert, Patrick Wieschollek, and Hendrik P. A. Lensch. Ggnn: Graph-based gpu nearest neighbor search. *IEEE Transactions on Big Data*, 9:267–279, 2019.

[307] Hiroyuki Ootomo, Akira Naruse, Corey J. Nolet, Ray Wang, Tamas B. Fehér, and Y. Wang. Cagra: Highly parallel graph construction and approximate nearest neighbor search for gpus. *ArXiv*, abs/2308.15136, 2023.

[308] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[309] BlueLM Team. Bluelm: An open multilingual 7b language model. `https://github.com/vivo-ai-lab/BlueLM`, 2023.

[310] Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023.

[311] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[312] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[313] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International conference on machine learning*, pages 201–210. PMLR, 2016.

[314] Tianyu Chen, Hangbo Bao, Shaohan Huang, Li Dong, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. The-x: Privacy-preserving transformer inference with homomorphic encryption. *arXiv preprint arXiv:2206.00216*, 2022.

[315] Brandon Reagen, Woo-Seok Choi, Yeongil Ko, Vincent T Lee, Hsien-Hsin S Lee, Gu-Yeon Wei, and David Brooks. Cheetah: Optimizing and accelerating homomorphic encryption for private inference. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 26–39. IEEE, 2021.

[316] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (Csur)*, 51(4):1–35, 2018.

[317] Florian Tramer and Dan Boneh. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. *arXiv preprint arXiv:1806.03287*, 2018.

[318] Shufan Fei, Zheng Yan, Wenxiu Ding, and Haomeng Xie. Security vulnerabilities of sgx and countermeasures: A survey. *ACM Computing Surveys (CSUR)*, 54(6):1–36, 2021.

[319] Erika McCallister. *Guide to protecting the confidentiality of personally identifiable information*, volume 800. Diane Publishing, 2010.

[320] Maximin Coavoux, Shashi Narayan, and Shay B Cohen. Privacy-preserving neural representations of text. *arXiv preprint arXiv:1808.09408*, 2018.

[321] Xin Zhou, Jinzhu Lu, Tao Gui, Ruotian Ma, Zichu Fei, Yuran Wang, Yong Ding, Yibo Cheung, Qi Zhang, and Xuan-Jing Huang. Textfusion: Privacy-preserving pre-trained model inference via token fusion. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8360–8371, 2022.

[322] Xin Zhou, Yi Lu, Ruotian Ma, Tao Gui, Yuran Wang, Yong Ding, Yibo Zhang, Qi Zhang, and Xuan-Jing Huang. Textobfuscator: Making pre-trained language model a privacy protector via obfuscating word representations. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5459–5473, 2023.

[323] Xiaodong Liu, Hao Cheng, Pengcheng He, Weizhu Chen, Yu Wang, Hoifung Poon, and Jianfeng Gao. Adversarial training for large neural language models. *arXiv preprint arXiv:2004.08994*, 2020.

[324] Franziska Roesner, Tadayoshi Kohno, Alexander Moshchuk, Bryan Parno, Helen J Wang, and Crispin Cowan. User-driven access control: Rethinking permission granting in modern operating systems. In *2012 IEEE Symposium on Security and Privacy*, pages 224–238. IEEE, 2012.

[325] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2):1–29, 2014.

[326] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

[327] Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K. Jain. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17(2): 151–178, 2020. doi: 10.1007/s11633-019-1211-x.

[328] Aounon Kumar, Chirag Agarwal, Suraj Srinivas, Aaron Jiaxun Li, Soheil Feizi, and Himabindu Lakkaraju. Certifying llm safety against adversarial prompting, 2023.

[329] Yunqing Zhao, Tianyu Pang, Chao Du, Xiao Yang, Chongxuan Li, Ngai-Man Cheung, and Min Lin. On evaluating adversarial robustness of large vision-language models. *arXiv preprint arXiv:2305.16934*, 2023.

[330] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? *arXiv preprint arXiv:2307.02483*, 2023.

[331] Christian Schlarmann and Matthias Hein. On the adversarial robustness of multi-modal foundation models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3677–3685, 2023.

[332] Xiaohan Fu, Zihan Wang, Shuheng Li, Rajesh K. Gupta, Niloofar Mireshghallah, Taylor Berg-Kirkpatrick, and Earlence Fernandes. Misusing tools in large language models with visual adversarial examples, 2023.

[333] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019. doi: 10.1109/ACCESS.2019.2909068.

[334] Yizhen Yuan, Rui Kong, Shenghao Xie, Yuanchun Li, and Yunxin Liu. Patchbackdoor: Backdoor attack against deep neural networks without model modification. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 9134–9142, 2023.

[335] Nikhil Kandpal, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. Backdoor attacks for in-context learning with language models. *arXiv preprint arXiv:2307.14692*, 2023.

[336] Shuai Zhao, Jinming Wen, Luu Anh Tuan, Junbo Zhao, and Jie Fu. Prompt as triggers for backdoor attack: Examining the vulnerability in language models, 2023.

[337] Hongwei Yao, Jian Lou, and Zhan Qin. Poisonprompt: Backdoor attack on prompt-based large language models, 2023.

[338] Xiaofei Sun, Xiaoya Li, Yuxian Meng, Xiang Ao, Lingjuan Lyu, Jiwei Li, and Tianwei Zhang. Defending against backdoor attacks in natural language generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 5257–5265, 2023.

[339] Sahar Abdelnabi, Kai Greshake, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, AISec '23, page 79–90, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702600. doi: 10.1145/3605764.3623985.

[340] Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models, 2022.

[341] Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, and Yang Liu. Prompt injection attack against llm-integrated applications, 2023.

[342] Erfan Shayegani, Yue Dong, and Nael Abu-Ghazaleh. Jailbreak in pieces: Compositional adversarial attacks on multi-modal language models, 2023.

[343] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries, 2023.

[344] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650. USENIX Association, August 2021. ISBN 978-1-939133-24-3.

[345] Alexander Robey, Eric Wong, Hamed Hassani, and George J. Pappas. Smoothllm: Defending large language models against jailbreaking attacks, 2023.

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

[347] Vipula Rawte, Amit Sheth, and Amitava Das. A survey of hallucination in large foundation models. *arXiv preprint arXiv:2309.05922*, 2023.

[348] Varun Nair, Elliot Schumacher, Geoffrey Tso, and Anitha Kannan. Dera: enhancing large language model completions with dialog-enabled resolving agents. *arXiv preprint arXiv:2303.17071*, 2023.

[349] Yifan Zhang, Jingqin Yang, Yang Yuan, and Andrew Chi-Chih Yao. Cumulative reasoning with large language models. *arXiv preprint arXiv:2308.04371*, 2023.

[350] Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. Learning from mistakes makes llm better reasoner. *arXiv preprint arXiv:2310.20689*, 2023.

[351] Zhaocheng Zhu, Yuan Xue, Xinyun Chen, Denny Zhou, Jian Tang, Dale Schuurmans, and Hanjun Dai. Large language models can learn rules. *arXiv preprint arXiv:2310.07064*, 2023.

[352] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

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

[354] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2021.

[355] Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023.

[356] Guan Wang, Sijie Cheng, Xianyuan Zhan, Xiangang Li, Sen Song, and Yang Liu. Openchat: Advancing open-source language models with mixed-quality data. *arXiv preprint arXiv:2309.11235*, 2023.

[357] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

[358] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023.

[359] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023.

[360] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*, 2023.

[361] Potsawee Manakul, Adian Liusie, and Mark JF Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896*, 2023.

[362] Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023.

[363] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR, 2020.

[364] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[365] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2, 2019.

[366] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR, 2023.

[367] Wenhao Yu, Hongming Zhang, Xiaoman Pan, Kaixin Ma, Hongwei Wang, and Dong Yu. Chain-of-note: Enhancing robustness in retrieval-augmented language models. *arXiv preprint arXiv:2311.09210*, 2023.

[368] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*, 2023.

[369] Yile Wang, Peng Li, Maosong Sun, and Yang Liu. Self-knowledge guided retrieval augmentation for large language models. *arXiv preprint arXiv:2310.05002*, 2023.

[370] Zhiruo Wang, Jun Araki, Zhengbao Jiang, Md Rizwan Parvez, and Graham Neubig. Learning to filter context for retrieval-augmented generation. *arXiv preprint arXiv:2311.08377*, 2023.

[371] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738*, 2023.

[372] Sachin Kumar, Biswajit Paria, and Yulia Tsvetkov. Gradient-based constrained sampling from language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2251–2277, 2022.

[373] Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6834–6842, 2019.

[374] Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023.

[375] Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.

[376] Marina Danilevsky, Kun Qian, Ranit Aharonov, Yannis Katsis, Ban Kawas, and Prithviraj Sen. A survey of the state of explainable ai for natural language processing. *arXiv preprint arXiv:2010.00711*, 2020.

[377] Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. Explainability for large language models: A survey. *arXiv preprint arXiv:2309.01029*, 2023.

[378] Sarah Wiegreffe and Ana Marasović. Teach me to explain: A review of datasets for explainable natural language processing. *arXiv preprint arXiv:2102.12060*, 2021.

[379] Samuel Carton, Surya Kanoria, and Chenhao Tan. What to learn, and how: Toward effective learning from rationales. In *Findings of the Association for Computational Linguistics: ACL 2022*, 2022.

[380] Sai Gurrapu, Ajay Kulkarni, Lifu Huang, Ismini Lourentzou, and Feras A Batarseh. Rationalization for explainable nlp: A survey. *Frontiers in Artificial Intelligence*, 6, 2023.

[381] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[382] Jiashuo Sun, Yi Luo, Yeyun Gong, Chen Lin, Yelong Shen, Jian Guo, and Nan Duan. Enhancing chain-of-thoughts prompting with iterative bootstrapping in large language models. *arXiv preprint arXiv:2304.11657*, 2023.

[383] Danny Halawi, Jean-Stanislas Denain, and Jacob Steinhardt. Overthinking the truth: Understanding how language models process false demonstrations. *arXiv preprint arXiv:2307.09476*, 2023.

[384] Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *arXiv preprint arXiv:2306.03341*, 2023.

[385] Liam van der Poel, Ryan Cotterell, and Clara Meister. Mutual information alleviates hallucinations in abstractive summarization. *arXiv preprint arXiv:2210.13210*, 2022.