Octopus v4: Graph of language models

Wei Chen *
Nexa AI
{alexchen}@nexa4ai.com

Zhiyuan Li
Nexa AI
{zack}@nexa4ai.com

Abstract

Language models have been effective in a wide range of applications, yet the most sophisticated models are often proprietary. For example, GPT-4 by OpenAI and various models by Anthropic are expensive and consume substantial energy. In contrast, the open-source community has produced competitive models, like Llama3. Furthermore, niche-specific smaller language models, such as those tailored for legal, medical or financial tasks, have outperformed their proprietary counterparts. This paper introduces a novel approach that employs *functional tokens* to integrate **multiple open-source models**, each optimized for particular tasks. Our newly developed Octopus v4 model leverages *functional tokens* to intelligently direct user queries to the most appropriate vertical model and reformat the query to achieve the best performance. Octopus v4, an evolution of the Octopus v1, v2, and v3 models, excels in selection and parameter understanding and reformatting. Additionally, we explore the use of graph as a versatile data structure that effectively coordinates multiple open-source models by harnessing the capabilities of the Octopus model and *functional tokens*. Use our open-sourced GitHub (`https://github.com/NexaAI/octopus-v4`) to try Octopus v4 models (`https://huggingface.co/NexaAIDev/Octopus-v4`), and contrite to a larger graph of language models. By activating models around 10B parameters, we achieved SOTA MMLU score of 74.8 among the same level models.

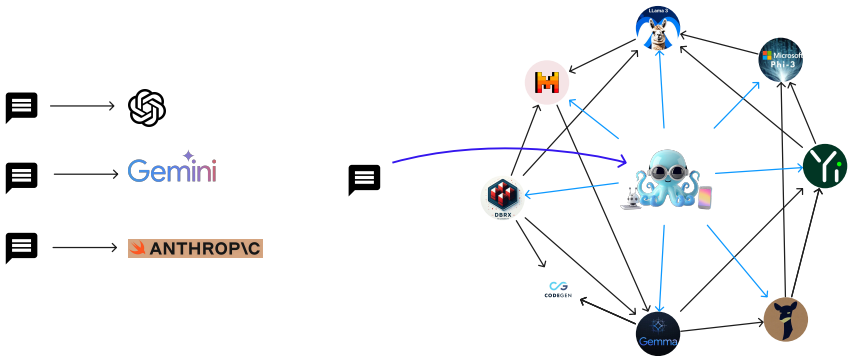

Figure 1: The shift from single model inference, employing a trillion-parameter model, to multi-node collaboration coordinated by Octopus model. This framework optimizes the inference process by selecting the most suitable specialized models based on the user's query, activating only two models that each has fewer than 10B parameters for one-step inference. We only show a small graph here, but the framework can support a large graph. See the demonstration of the graph (`https://graph.nexa4ai.com/`) here.

*Corresponding author

Wei (Alex) Chen and Zhiyuan (Zack) Li are also authors of Octopus v1, v2 and v3, visit our website for more information: `https://www.nexa4ai.com`. We have open sourced Octopus v4 model used in our experiment (`https://huggingface.co/NexaAIDev/Octopus-v4`) on Hugging Face, and we have an open sourced github repo (`https://github.com/NexaAI/octopus-v4`) to build the graph of language models. Join us!

1 Introduction

The rapid progress in large language models (LLMs) has revolutionized natural language processing, enabling AI systems to understand and generate human-like text with remarkable accuracy. LLMs like GPT-4 [29] and Anthropic [5], trained on vast datasets, can capture the nuances of language, context, and meaning, leading to a wide range of potential use cases across industries. These models excel at various substream tasks, such as highly accurate language translation, sentiment analysis to gauge customer opinions and monitor brand reputation, question answering by drawing from extensive knowledge bases, and summarizing lengthy documents and articles. In healthcare, LLMs can process patient data and medical literature to support disease diagnosis, treatment planning, and drug discovery, even generating personalized patient reports [34, 10]. The finance industry leverages LLMs for risk assessment, fraud detection, and automating financial report generation [44, 20], while legal domains benefit from streamlined contract analysis, legal research, and document drafting [16, 26]. LLMs also have significant potential in education [34, 37], providing personalized learning experiences, generating content, and offering instant feedback. As LLMs continue to advance, their impact across various domains is expected to grow significantly, automating tasks, enhancing decision-making, and driving innovation.

Since Meta's release of the Llama3 model and its successor, Llama 2 [3], in 2023, the open-source landscape for large language models (LLMs) has seen significant growth. This shift catalyzed the development of numerous innovative LLMs, each released at an unprecedented rate. As key players in this dynamic field, these models have significantly influenced natural language processing. We highlight the most impactful open-source LLMs, including Mistral's sparse Mixture of Experts model Mixtral-8x7B [4, 21], Alibaba Cloud's multilingual Qwen1.5 series [6], Abacus AI's Smaug [32], and 01.AI's Yi models [49] that focus on data quality. Other notable models include Databricks' fine-grained MoE model DBRX [40] , Upstage AI's depth-upscaled SOLAR-10.7B [23] , Allen Institute for AI's alignment-focused TULU 2 and the collaborative OLMo series [18], Microsoft's WizardLM powered by Evol-Instruct [46], Berkeley's Starling-7B [53], Google DeepMind's Gemini-inspired Gemma models [17], xAI's Grok [30], and Deci AI's high-efficiency DeciLM-7B [38]. In April 2024, we witnessed the most powerful open-source model up-to-date, Meta's Llama3 [39] , and its 70B parameter version achieved an impressive inference speed of approximately 300 tokens per second using Groq [39]. Shortly thereafter, more powerful open-sourced on-device models were released, including Microsoft's Phi-3-mini with 3.8 billion parameters [1] and Apple's OpenELM family [28] of models which range from 1 to 3 billion parameters. These diverse models cater to various use cases, so that the users may select the optimal model based on the use case.

Graph data structures have emerged as a powerful tool for representing complex relationships and dependencies in various domains. In computer science, a graph consists of a set of nodes (or vertices) connected by edges, which can be directed or undirected. This flexible structure allows for the representation of intricate connections and hierarchies that are difficult to capture using linear or tabular formats. Graphs offer several advantages over other data structures, including efficient traversal, pattern discovery, and the ability to model real-world networks. Many prominent companies have leveraged graph-based approaches to enhance their products and services. For example, Pinterest uses a graph structure to represent the relationships between users, boards, and pins, enabling personalized content recommendations and improved user engagement. Similarly, social networks like Facebook and LinkedIn rely on graph representations to model user connections, facilitating features such as friend suggestions and professional networking. In the context of integrating open-source language models, graph structures can be employed to represent the relationships between different models, their capabilities, and their optimal use cases. By treating each language model as a node in the graph and establishing edges based on their compatibility, complementary features, or task-specific performance, we can create a powerful framework for seamless model integration, intelligent query routing, and optimized performance.

The advent of on-device AI models has revolutionized the landscape of natural language processing, offering a host of advantages over traditional cloud-based approaches. These models, exemplified by Google's Gemma2B and the Llama7B model, are designed to run directly on user devices, ensuring data privacy by processing queries locally and eliminating the need for network communication with distant servers. This local processing not only enhances security but also reduces latency, enabling real-time interactions and improved user experiences. On-device AI agents, such as Octopus v2 [11, 13], v3 [12], leverage these capabilities to provide intelligent assistance. However, the true

potential of on-device AI lies in its seamless integration with cloud-based models, giving rise to the concept of **cloud-on-device collaboration** [48, 36]. By harnessing the power of both on-device and cloud-based models, AI systems can achieve unprecedented levels of performance, scalability, and flexibility. This collaboration allows for the efficient allocation of computational resources, with on-device models handling lighter and private tasks and cloud-based models tackling more complex or resource-intensive operations. Moreover, the Internet of Things (IoT) plays a crucial role in enabling this collaboration, connecting a vast network of devices and facilitating the exchange of data and insights between on-device and cloud-based models. The integration of on-device AI, cloud-based models, and IoT represents a paradigm shift in AI development. This approach combines the strengths of each component, creating a synergistic ecosystem that can adapt to the diverse needs of users and applications. As we continue to explore the potential of cloud-on-device collaboration, we can expect to see significant advancements in the field of AI.

In this paper, we introduce a new framework to use language models by constructing a graph with different vertical language models as the nodes. We use the feature of Octopus v2 model and use it as the coordinator. The transition from the single model inference to the multi-node inference is demonstrated in Figure (1).

2 Related works

Graph data format Graph algorithms have been a fundamental area of research in computer science, with a wide range of applications spanning from social network analysis to recommendation systems and bioinformatics. Classic graph algorithms, such as breadth-first search (BFS) and depth-first search (DFS), have been extensively studied and optimized for various graph representations. Dijkstra's shortest path algorithm and its variations have been crucial in solving routing problems and finding optimal paths in weighted graphs. The PageRank algorithm [7], famously used by Google to rank web pages, has revolutionized the field of information retrieval and inspired numerous graph-based ranking techniques. Recent advancements in graph neural networks (GNNs) [51, 45, 35, 47] have pushed the boundaries of graph-based learning, enabling the processing of graph-structured data for tasks such as node classification, link prediction, and graph generation. Frontier research in this area includes the development of more expressive and efficient GNN architectures, such as Graph Attention Networks (GATs) [42, 8] and Graph Convolutional Networks (GCNs) [50, 43], which have achieved state-of-the-art performance on various graph-related tasks.

Enhancing AI agents with functional tokens Building on the Octopus series (v1 to v3)[13, 11, 12], this research extends the capabilities of AI agents by utilizing functional tokens, and uniting all the open source models. These earlier versions effectively harnessed these tokens for advanced functionalities. We now investigate their extended use in integrating diverse open source language models. Our studies indicate that functional tokens exceed mere precision in classification tasks, such as selecting suitable functions or models for processing queries. Importantly, they amplify the Octopus model's ability to interpret and reshape user queries into an optimal format for the designated function, enhancing performance. This synergy between functional tokens and the Octopus models' capabilities in classification and query reformulation has been further applied in graph structures. Here, a pivotal aspect involves transferring information between nodes and selecting the appropriate neighborhood for this transfer. Our enhanced Octopus model efficiently selects the best neighbor, restructures the information at the current node, and transmits optimized information to subsequent nodes.

Multi-agent LLMs Multi-agent LLMs mark a pivotal evolution in AI, facilitating collaborative problem-solving through the integration of multiple specialized agents [52]. Unlike traditional single-agent LLMs, these multi-agent systems harness collective intelligence from agents specialized in various domains. This collaborative approach yields more comprehensive solutions to complex issues. Multi-agent LLMs excel in delivering domain-specific expertise, enhancing problem-solving abilities, and offering robustness, reliability, and adaptability. These systems promise transformative impacts across sectors like healthcare, finance, education, and customer service by providing tailored expertise, personalized interactions, and efficient decision-making processes. However, deploying multi-agent LLMs involves challenges such as integration difficulties, data sharing issues, and maintaining smooth coordination between agents. Ongoing research into multi-agent LLMs is exploring possibilities like cross-domain expertise and real-time collaboration while considering ethical aspects. Additionally, the adoption of graph architectures in our paper is also inspired by the multi-agent system. Advanced

functionalities like parallel function calling can be achieved through **self-connections** and sequential action processing via **graph traversal**, enhancing their operational efficiency and scalability.

LLM scaling law Scaling laws [22] for Large Language Models (LLMs) have revolutionized our understanding of the relationship between model size, dataset size, computational resources, and performance. These laws indicate that larger models trained on vast datasets with ample computational power generally outperform smaller ones. However, as LLMs continue to scale up, they face significant challenges related to server capacity and power consumption, which limit their further expansion. Our proposed architecture addresses these scalability issues by leveraging distributed computing and node expansion techniques, enabling **nearly unlimited node scalability**. We can create more powerful language model system by adding more nodes, bypassing the limitations imposed by server quantity and power supply.

3 Methodology

This section outlines the primary methods for incorporating language models as nodes within a graph and provides details on the system architecture tailored for real applications. It also discusses the training strategy for the Octopus model using a synthetic dataset. Also, we highlight the system design for such a graph of language models in a production environment.

3.1 Language model for classification from Octopus v2

In the Octopus v2 paper, we introduced a method named *functional token* for classification within a fixed pool. The Octopus v2 model effectively handles the task of

$$P(f, params|q), \tag{1}$$

where f denotes a choice from the set F, and $params$ represents the reformulated information derived from the query q. The paper illustrates this method's application in the task of function calling. Additionally, the functional token can be adapted to other similar scenarios that require selecting the optimal choice from a specified pool and reformulating the query to transfer information to subsequent nodes. In typical use cases involving a predefined graph, each node, represented as a language model, has a fixed number of neighbors. To perform language model inference, the best neighboring node is selected, and information from the current node is passed to the next. Thus, the Octopus v2 model is well-suited for handling this problem, demonstrating both rapid execution and high accuracy as documented in the Octopus v2 paper.

3.2 Language models as nodes in graph

Consider a directed and heterogeneous graph defined as:

$$G = (N, E), \tag{2}$$

where N denotes the various nodes within the graph, and E represents the edges that connect these nodes. Nodes are distinguished into two types: *master nodes*, N^m, which coordinate queries by directing them to the suitable *worker nodes*, N^w, and transfer necessary information for task execution. Worker nodes receive information from the master node and execute the required tasks, using an Octopus model to facilitate further coordination. The process of the node information transfer is demonstrated in the Figure (2). For processing user queries q and generating responses y, we model the probability as:

$$P(y|q) = P(y|q; G). \tag{3}$$

For a single-step task involving only one worker node, the procedure can be defined as follows:

$$P(y|q; G) = \underbrace{P(N^w, q_h|q; N^m)}_{\text{The Octopus v2 pattern problem}} \quad P(y|q_h; N^w) \tag{4}$$

Here, $P(N^w, q_h|q; N^m)$ uses an Octopus v2 model to select the best neighboring worker node for N^m and reformat the query into q_h, which is the reformated query. This expression is a typical

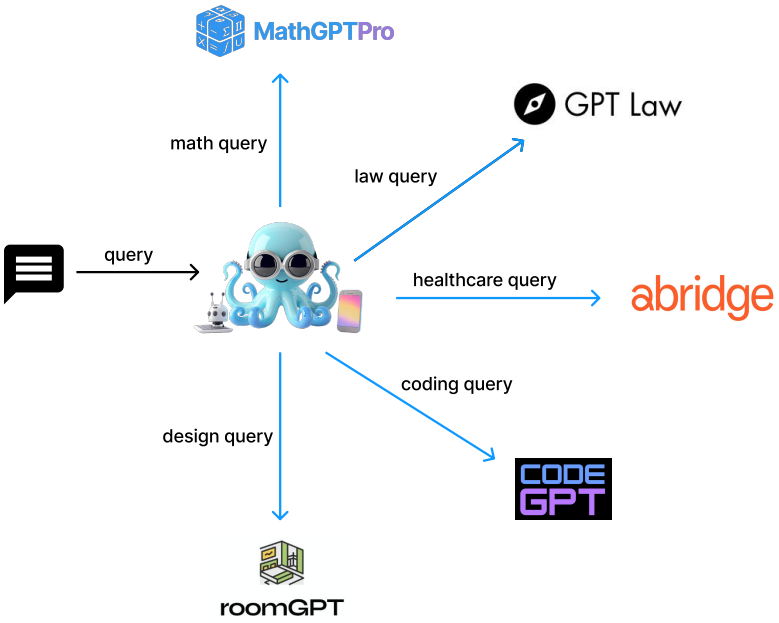

Figure 2: The Octopus model is utilized to determine the optimal neighboring node and generate appropriate information for transmission. Consider a scenario where the Octopus model's neighbors are MathGPT [27], LawGPT [14], HealthCareGPT [2], CodeGPT [15], and RoomGPT [33]. The Octopus model can identify the most relevant GPT and transform the initial query into a format best suited for the selected GPT.

problem that can be solved by Octopus model, which has the same structure as the equation (1). The likelihood $P(y|q_h; N^w)$ is calculated by the language model situated at the worker node.

For multistep tasks, typical in multi-agent workflows, the process involves several sequential interactions among multiple nodes, as follows:

$$P(y|q; G) = \prod_{i=0}^{k-1} \underbrace{P(N_i^w, q_{h_i}|q; N_i^m)}_{\text{The Octopus v2 pattern problem}} \quad P(y|q_{h_i}; N_i^w) \tag{5}$$

This formula expands the single-step task to multiple steps, each handled by potentially different worker nodes, coordinated by their respective master nodes. Each step processes a part of the query and contributes to the final outcome, with k representing the number of steps or interactions in the multi-agent process. This method exemplifies a coordination and execution pattern in a distributed AI system, leveraging the capabilities of multiple specialized agents within a graph-based framework.

For the graph, it would be a predefined graph based on the available language models. Each worker model can also be an Octopus model that can take actions. If we are going to take a parallel function calling, the master node will direct the query to the same nodes multiple times to execute the parallel function calling.

Compared to a large language model like GPT-4, this design has another advantage that to answer one query from the user, we only need to activate two small language models rather than a large language model with trillion parameters. This means we can expect faster speed and less energy cost, and fewer demands on the hardware. In Octopus v2, we have demonstrated that we can use functional token to get rid of RAG method to achieve accurate selection of functions, and fast generation. Thus, equation (4) is executed fast.

3.3 Task planning using graphs for multistep operations

In multistep task planning, incorporating numerous steps is essential. Traditionally, all available functions were listed in the context and submitted to a language model, which then generates plans based on user queries. This approach, however, faces limitations when the language model, especially

those with less than 10B parameters, attempts to process lengthy function descriptions. Such models struggle to grasp extensive descriptions effectively. Moreover, this method didn't consider the inherent relevance among different function descriptions. To address these challenges, constructing a graph that maps the relevance between various nodes (language models/agents) offers a viable solution. This graph-based approach ensures that only neighboring nodes relevant to a specific node are considered, significantly reducing the complexity of choices compared to the total number of function descriptions. By leveraging the capabilities of the Octopus v2 model, this strategy enhances efficiency, enabling rapid query redirection and reformatting. We actually have two layers of abstraction. Firstly, for each language model, we can apply the functional token to make it as a single AI agent which can take single function callings. Or, the single node/language model can be an ordinary language model like Llama3 or Phi3 which can do question and answering, writing etc. The other layer of abstraction is that we can also create another Octopus v3 model to choose from different nodes. The two layers of abstraction is demonstrated in the Figure (3)

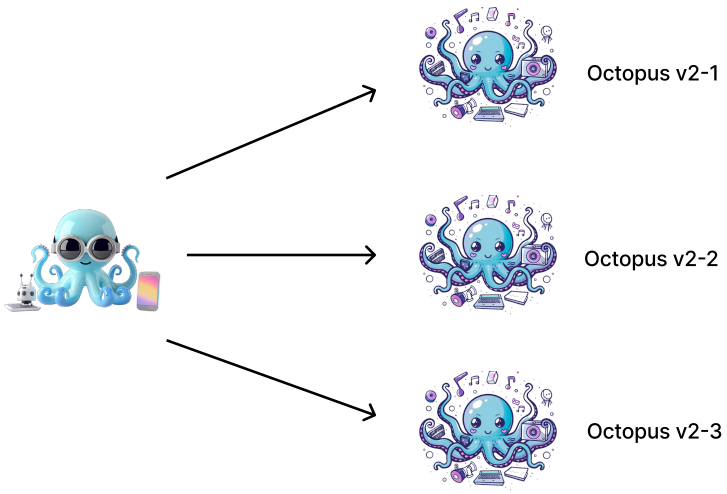

Figure 3: In our design, the architecture consists of two abstraction layers. The first layer utilizes functional tokens to represent the actions executable by the Octopus v2 model. This layer encompasses three distinct Octopus v2 models, each identified by different functional tokens, effectively differentiating them as separate AI agents. The second layer of abstraction pertains to the Octopus v4 model, where internal functional tokens are mapped to various v2 models. For simplicity, we only include three v2 models, but one can map to multiple v2 models in real use cases.

3.4 Functional token and dataset collections

Like the functional token architecture in Octopus v2, we conceptualize each model as a distinct function, utilizing functional tokens to activate specific model usage. This approach simplifies the function design for language models, requiring only a single input argument and output result. Additionally, for specific models, we can detail the required prompt template within the function's doc string. This allows the Octopus v4 model to restructure the original query to match the expected format. For instance, a function dedicated to processing legal information might be described as follows:

```
def law_gpt(query):
    """
    A specialized language model equipped to handle queries related to
    legal studies, including international law, jurisprudence, and
    professional law. This model serves law students, practicing lawyers,
    and professionals in the legal field needing detailed legal
    explanations or interpretations. This model also reformats user queries
     into professional legal language.

    Parameters:
```

```
- query (str): A detailed prompt that encapsulates a law-related
question or issue. Speak in a professional legal manner.

Returns:
- str: Comprehensive legal analyses, solutions, or information related
to the law query.
"""
```

Additionally, when we construct the dataset using similar strategy to Octopus v2 paper. Following the methodology outlined in the Octopus v2 paper, we can train multiple functional tokens corresponding to various custom language models. As in the Octopus v2 paper, the dataset collection process involves using synthetic data to train the functional tokens. To better accommodate diverse queries, it may be beneficial to increase the temperature during data generation. This adjustment helps capture the variability and potential formatting inconsistencies in user queries, which are common in certain use cases.

3.5 System design of language model graph

This section details the system architecture of a complex graph where each node represents a language model, utilizing multiple Octopus models for coordination. As we prepare for production deployment, it's crucial to integrate a load balancer to manage system demands efficiently. Below, we delineate the system into several manageable components, emphasizing the core methodologies:

- **Worker node deployment**: Each worker node N^w corresponds to an individual language model. We propose employing a serverless architecture for these nodes, specifically recommending Kubernetes [25] for its robust autoscaling capabilities based on memory usage and workload. We also limit the model parameters of all worker nodes to under 10B.
- **Master node deployment**: The master node should utilize a base model with fewer than 10B parameters (we use 3B model in the experiment), enabling deployment on edge devices. Each worker node interfaces with an Octopus model for enhanced coordination. As demonstrated in Octopus v2, a compact Lora model can be integrated to extend functional token capabilities. We suggest using a single base model supplemented by multiple Loras, one per worker node. The LoraX library, an open-source initiative, is recommended for managing the inference operations with this configuration.
- **Communication**: Worker and master nodes are distributed across various devices, not confined to a single unit. Thus, internet connectivity is essential for transmitting data between nodes. While the master node may be situated on smart devices, worker nodes are hosted in the cloud or on alternate devices, with results relayed back to the smart device. To support data caching needs, including chat history storage, we recommend utilizing Redis [9], a high-performance, in-memory database that facilitates distributed caching.

The overall system design architecture is depicted in Figure (4).

4 Experiments

In this section, we detail the experiments performed with our framework, aimed at enhancing language model performance via multi-node collaboration. We demonstrate how our framework can improve language model efficacy using the MMLU benchmark [19] for evaluation. For this purpose, we apply 17 distinct models across the MMLU tasks. Upon receiving a user query, the Octopus v4 model directs the query to the relevant specialized model, which then reformats it suitably. The following experiment employs a simple graph; a more complex graph will be provided on our GitHub repo (`https://github.com/NexaAI/octopus-v4`) in the future. And the ultimate graph needs the effort from the whole community.

4.1 Specialized models

The Multitask Multi-Language Understanding (MMLU) encompasses 57 unique tasks, further categorized into 17 consolidated groups as recommended by the authors. Tasks such as graduate-level

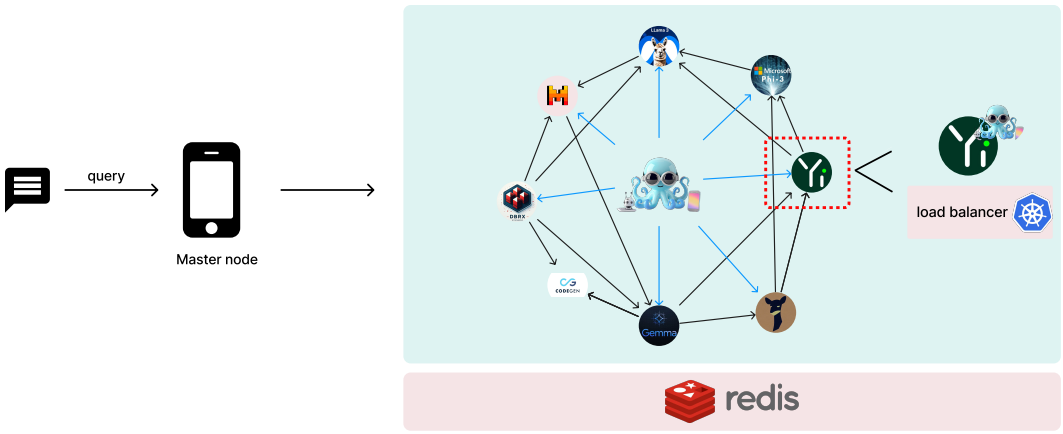

Figure 4: Our system design features a graph of language models with a master node deployed on a central device and worker nodes distributed across various devices. We employ Kubernetes (k8s) for serverless deployment of each individual worker language model. For efficient data sharing, we utilize a distributed cache mechanism supported by Redis. Note that for each worker node, we have a small Octopus v4 Lora attached to it to guide the next neighbor node for the case of multi-Agent use cases.

and high-school math have been grouped into the broader *math* category. The tasks are divided as follows:

- **STEM**: Physics, Chemistry, Biology, Computer Science, Math, Engineering;
- **Humanities**: History, Philosophy, Law;
- **Social Sciences**: Politics, Culture, Economics, Geography, Psychology;
- **Other**: Miscellaneous, Business, Health.

Specialized models are curated from Hugging Face based on benchmark scores, popularity, and user endorsements. Not all specialized tasks have corresponding models; for example, models for Humanities and Social Sciences are notably absent. Nonetheless, the Llama3 model, adjusted with tailored system prompts, serves as a base model to simulate specialized capabilities without direct fine-tuning. The following 17 models are either specifically fine-tuned or customized through prompts:

- **Physics**: *Weyaxi/Einstein-v6.1-Llama3-8B* (`https://huggingface.co/Weyaxi/Einstein-v6.1-Llama3-8B`), fine-tuned on a physics dataset (`https://huggingface.co/datasets/camel-ai/physics`);
- **Biology**: *jondurbin/bagel-8b-v1.0* (`https://huggingface.co/jondurbin/bagel-8b-v1.0`), fine-tuned on a biology dataset;
- **Computer Science**: *Llama-3-Smaug-8B* (`https://huggingface.co/abacusai/Llama-3-Smaug-8B`), tailored for various computer science forums;
- **Math**: *Open-Orca/Mistral-7B-OpenOrca*, optimized for math (`https://huggingface.co/Open-Orca/Mistral-7B-OpenOrca`);
- **Engineering**: *phi-2-electrical-engineering* (`https://huggingface.co/STEM-AI-mtl/phi-2-electrical-engineering`), fine-tuned on an electrical engineering dataset, selected for its relevance to MMLU;
- **Law**: *AdaptLLM/law-chat* (`https://huggingface.co/AdaptLLM/law-chat`), fine-tuned on a law dataset;
- **Health**: *AdaptLLM/medicine-chat* (`https://huggingface.co/AdaptLLM/medicine-chat`), optimized for medical data;
- **Psychology, History, Philosophy, Politics, Culture, Geography, Business, Chemistry, Economics**: Currently, there are no specialized models available for these areas. Custom system prompts and CoT techniques are used with Llama3 to simulate specialized models;

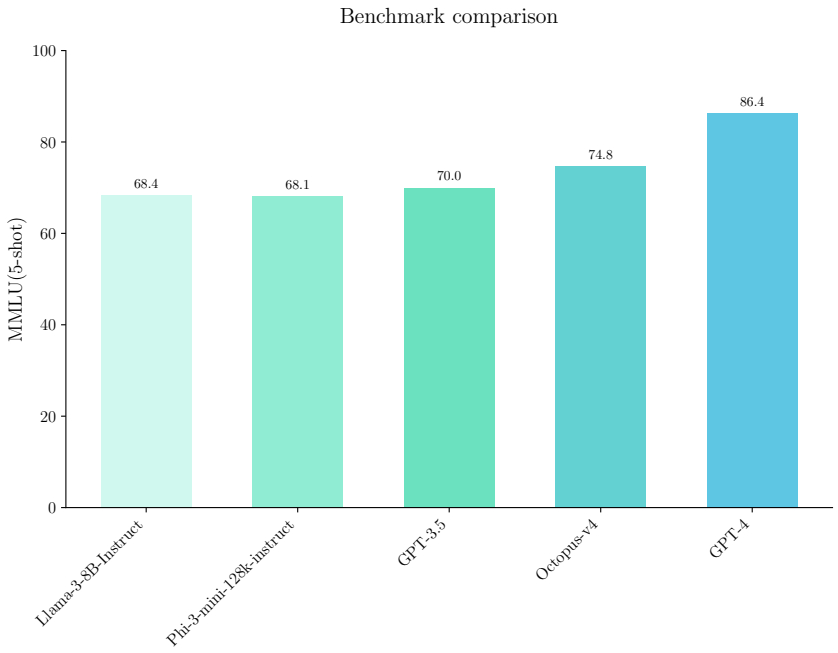

Figure 5: The comparison of MMLU scores between Octopus v4 and other models. During Octopus v4's inference, only two small language models, each with fewer than 10B parameters, are activated. Octopus v4 achieves significant improvement in MMLU scores, requiring only a small sacrifice of tokens due to the utilization of functional tokens.

- **Other**: For remaining tasks, the Phi3 model (`https://huggingface.co/microsoft/Phi-3-mini-128k-instruct`) is employed as a general-purpose model.

4.2 MMLU benchmark evaluation

This section presents a benchmark evaluation of the Octopus v4 system, comparing its performance with other renowned models using the MMLU benchmark to demonstrate our model's effectiveness. In our inference system, we utilize two compact language models: the 3B parameter Octopus v4 model and another worker language model with no more than 8B parameters. The comparison is shown in Figure (5). The procedure of the inference is demonstrated in Figure (2).

One example of the user query is highlighted below:

> Query: Tell me the result of derivative of x^3 when x is 2?
>
> Response: <nexa_4> ('Determine the derivative of the function $f(x) = x^3$ at the point where x equals 2, and interpret the result within the context of rate of change and tangent slope.')<nexa_end>

Note that `<nexa_4>` is a functional token which maps to math gpt.

5 Discussion and future works

In this section, we highlight a tutorial to train specialized model. Also, we outline the future plan of our team.

5.1 How to train a vertical model

To effectively fine-tune a large language model for domain-specific expertise, begin by gathering a substantial corpus of high-quality, domain-relevant data. This collection should include textbooks, research papers, articles, and other pertinent materials that thoroughly address the domain. It is crucial to ensure the data is diverse, well-organized, and embodies the domain knowledge intended for the model. Proceed by preprocessing this data—cleaning, consistent formatting, and addressing any specialized jargon or terminology.

Select a pre-trained large language model that suits your needs, and use the preprocessed domain-specific data for fine-tuning. This process adjusts the model's parameters to specialize in your chosen domain, effectively embedding the necessary expertise. Optionally, consider employing knowledge distillation to transfer insights from a larger model's API to a smaller, more efficient model. For this fine-tuning phase, the SFT trainer (`https://huggingface.co/docs/trl/sft_trainer`) provided by Hugging Face offers a user-friendly interface. We recommend to use supervised fine-tuning followed by a direct preference optimization.

5.2 Future work

Our current GitHub project focuses on developing a graphical framework for language models, currently in its starting phase. We plan to enhance this framework by integrating a variety of vertical-specific models and including the advanced Octopus v4 models with multiagent capability. Future releases will feature more robust graphical representations in this repository. And the GitHub repo will be maintained carefully by Nexa AI. Compared with scaling law of larger model, dataset, our framework is not limited and we could create a large graph.

Additionally, we are developing Octopus 3.5, a multimodal model that processes vision, audio, and video data. Subsequent versions will incorporate this AI agent into our graphical framework. Nexa AI also aims to develop compact, specialized models for diverse vertical domains.

References

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

[2] Abridge. Powering deeper understanding in healthcare, 2024. URL `https://www.abridge.com/`. Accessed on April 26, 2024.

[3] Meta AI. Introducing meta llama 3: The most capable openly available llm to date, 2024. URL `https://ai.meta.com/blog/meta-llama-3/`. Accessed on April 26, 2024.

[4] Mistral AI. Mixtral of experts: A high quality sparse mixture-of-experts, 2024. URL `https://mistral.ai/news/mixtral-of-experts/`. Accessed on April 26, 2024.

[5] Anthropic. Introducing the next generation of claude, 2024. URL `https://www.anthropic.com/news/claude-3-family`. Accessed on April 26, 2024.

[6] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[7] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

[8] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

[9] Josiah Carlson. *Redis in action*. Simon and Schuster, 2013.

[10] Marco Cascella, Jonathan Montomoli, Valentina Bellini, and Elena Bignami. Evaluating the feasibility of chatgpt in healthcare: an analysis of multiple clinical and research scenarios. *Journal of medical systems*, 47(1):33, 2023.

[11] Wei Chen and Zhiyuan Li. Octopus v2: On-device language model for super agent. *arXiv preprint arXiv:2404.01744*, 2024.

[12] Wei Chen and Zhiyuan Li. Octopus v3: Technical report for on-device sub-billion multimodal ai agent. *arXiv preprint arXiv:2404.11459*, 2024.

[13] Wei Chen, Zhiyuan Li, and Mingyuan Ma. Octopus: On-device language model for function calling of software apis. *arXiv preprint arXiv:2404.01549*, 2024.

[14] Daixuan Cheng, Shaohan Huang, and Furu Wei. Adapting large language models via reading comprehension. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=y886UXPEZ0`.

[15] CodeGPT. Empower your business with a team of ai copilots, 2024. URL `https://codegpt.co/`. Accessed on April 26, 2024.

[16] Jiaxi Cui, Zongjian Li, Yang Yan, Bohua Chen, and Li Yuan. Chatlaw: Open-source legal large language model with integrated external knowledge bases. *arXiv preprint arXiv:2306.16092*, 2023.

[17] Google DeepMind Gemma Team. Gemma: Open models based on gemini research and technology, 2023. URL `https://goo.gle/GemmaReport`.

[18] Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*, 2024.

[19] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[20] Allen H Huang, Hui Wang, and Yi Yang. Finbert: A large language model for extracting information from financial text. *Contemporary Accounting Research*, 40(2):806–841, 2023.

[21] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[22] 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. *arXiv preprint arXiv:2001.08361*, 2020.

[23] Dahyun Kim, Chanjun Park, Sanghoon Kim, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, et al. Solar 10.7 b: Scaling large language models with simple yet effective depth up-scaling. *arXiv preprint arXiv:2312.15166*, 2023.

[24] Dahyun Kim, Yungi Kim, Wonho Song, Hyeonwoo Kim, Yunsu Kim, Sanghoon Kim, and Chanjun Park. sdpo: Don't use your data all at once. *arXiv preprint arXiv:2403.19270*, 2024.

[25] T Kubernetes. Kubernetes. *Kubernetes. Retrieved May*, 24:2019, 2019.

[26] Aditya Kuppa, Nikon Rasumov-Rahe, and Marc Voses. Chain of reference prompting helps llm to think like a lawyer. In *Generative AI+ Law Workshop*, 2023.

[27] Ramesh M. llama-2-13b-mathgpt-v4, 2024. URL `https://huggingface.co/rameshm/llama-2-13b-mathgpt-v4`. Accessed on April 26, 2024.

[28] Sachin Mehta, Mohammad Sekhavat, Qingqing Cao, Max Horton, Yanzi Jin, Frank Sun, Iman Mirzadeh, Mahyar Najibikohnehshahri, Dmitry Belenko, Peter Zatloukal, and Mohammad Rastegari. Openelm: An efficient language model family with open-source training and inference framework, 2024. URL `https://arxiv.org/abs/2404.14619`.

[29] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian

Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024.

[30] XAI Organization. Grok-1: This repository contains jax example code for loading and running the grok-1 open-weights model, 2024. URL `https://github.com/xai-org/grok-1`. Accessed on April 26, 2024.

[31] Lawrence Page. Method for node ranking in a linked database. *USA Patent*, 6, 2019.

[32] Arka Pal, Deep Karkhanis, Samuel Dooley, Manley Roberts, Siddartha Naidu, and Colin White. Smaug: Fixing failure modes of preference optimisation with dpo-positive. *arXiv preprint arXiv:2402.13228*, 2024.

[33] RoomsGPT. Design your dream space with roomsgpt ai tools: Create your dream home or living space with roomgpt's free ai online design tools, 2024. URL `https://www.roomsgpt.io/`. Accessed on April 26, 2024.

[34] Malik Sallam. Chatgpt utility in healthcare education, research, and practice: Systematic review on the promising perspectives and valid concerns. *Healthcare*, 11(6):887, 2023.

[35] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[36] Zhichuang Sun, Ruimin Sun, Long Lu, and Alan Mislove. Mind your weight (s): A large-scale study on insufficient machine learning model protection in mobile apps. In *30th USENIX security symposium (USENIX security 21)*, pages 1955–1972, 2021.

[37] Ruixiang Tang, Yu-Neng Chuang, and Xia Hu. The science of detecting llm-generated text. *Communications of the ACM*, 67(4):50–59, 2024.

[38] DeciAI Research Team. Decilm-7b, 2023. URL `https://huggingface.co/Deci/DeciLM-7B`.

[39] LLama AI Team. Llama 3 on groq: Achieving unprecedented token generation speeds with integration on groq cloud, 2024. URL `https://llama-2.ai/llama-3-on-groq/`. Accessed on April 26, 2024.

[40] The Mosaic Research Team. Introducing dbrx: A new state-of-the-art open llm, 2024. URL `https://www.databricks.com/blog/introducing-dbrx-new-state-art-open-llm`. Accessed on April 26, 2024.

[41] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[42] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.

[43] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

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

[45] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1): 4–24, 2020.

[46] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2023.

[47] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[48] Mengwei Xu, Feng Qian, Qiaozhu Mei, Kang Huang, and Xuanzhe Liu. Deeptype: On-device deep learning for input personalization service with minimal privacy concern. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(4):1–26, 2018.

[49] Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*, 2024.

[50] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23, 2019.

[51] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

[52] Ziyuan Zhou, Guanjun Liu, and Ying Tang. Multi-agent reinforcement learning: Methods, applications, visionary prospects, and challenges. *arXiv preprint arXiv:2305.10091*, 2023.

[53] Banghua Zhu, Evan Frick, Tianhao Wu, Hanlin Zhu, and Jiantao Jiao. Starling-7b: Improving llm helpfulness & harmlessness with rlaif, November 2023.

Appendix

The functions used in our experiments are shown below:

```
def physics_gpt(query):
    """
    A specialized language model designed to answer questions and provide
    insights on physics-related topics, including conceptual physics,
    college physics, high school physics, and astronomy. This model caters
    to learners at different educational stages, from high school to
    college levels. This model also reformat user queries into professional
     physics language.

    Parameters:
    - query (str): A detailed prompt that encapsulates a physics-related
    question or problem. It is designed to support a deep and professional
    discussion of physics topics.

    Returns:
    - str: Detailed explanations, solutions, or information related to the
    physics query.
    """


def chemistry_gpt(query):
    """
    A specialized language model tailored to assist with chemistry topics,
    including high school chemistry, college chemistry, and related
    chemical sciences. This tool aids students and researchers in deepening
     their understanding of chemical concepts and practices. This model
    also reformats user queries into professional chemistry language.

    Parameters:
    - query (str): A detailed prompt that encapsulates a chemistry-related
    question or problem. The language used is intended for a sophisticated
    exploration of chemistry.

    Returns:
    - str: Detailed explanations, solutions, or information related to the
    chemistry query.
    """


def biology_gpt(query):
```

```
    """
    This language model is dedicated to providing insights and answers on
    biology, encompassing high school biology, college biology, human
    anatomy, and related fields. It is an essential resource for students
    across educational levels and biology enthusiasts. This model also
    reformats user queries into professional biology language.

    Parameters:
    - query (str): A detailed prompt that encapsulates a biology-related
    question or problem, suitable for detailed and expert-level discussion.

    Returns:
    - str: Detailed explanations, solutions, or information related to the
    biology query.
    """


def computer_science_gpt(query):
    """
    Designed for computer science queries, this language model covers
    topics such as college computer science, high school computer science,
    computer security, and machine learning. It supports both academic and
    professional needs, enhancing learning and research in the field of
    computer science. This model also reformats user queries into
    professional computer science language.

    Parameters:
    - query (str): A detailed prompt related to computer science topics,
    suitable for academic and professional discussions.

    Returns:
    - str: Detailed responses that enhance understanding and provide
    solutions in computer science.
    """


def math_gpt(query):
    """
    A specialized language model designed to answer questions and provide
    insights on math-related topics, including abstract algebra, elementary
     mathematics, high school mathematics, college mathematics, and high
    school statistics. This model supports learners at various educational
    levels from high school to college. This model also reformats user
    queries into professional math language.

    Parameters:
    - query (str): A detailed prompt that encapsulates a math-related
    question or problem. Speak in a professional mathematician manner.

    Returns:
    - str: Detailed explanations, solutions, or information related to the
    math query.
    """


def electrical_engineering_gpt(query):
    """
    This language model offers expert guidance on electrical engineering
    topics, designed to support students, educators, and professionals in
```

```
    the field. It addresses questions related to fundamental and advanced
    electrical engineering concepts. This model also reformats user queries
     into professional electrical engineering language.

    Parameters:
    - query (str): A detailed prompt that encapsulates an electrical
    engineering-related question or problem, fostering professional-level
    discussions.

    Returns:
    - str: Comprehensive responses, solutions, or information related to
    the electrical engineering query.
    """


def history_gpt(query):
    """
    A specialized language model designed to answer questions and provide
    insights on history-related topics. This model covers a broad range of
    historical subjects including high school European history, high school
     US history, high school world history, and prehistory. It aims to
    support learners and enthusiasts from various educational backgrounds.
    This model also reformats user queries into professional history
    language.

    Parameters:
    - query (str): A detailed prompt that encapsulates a history-related
    question or problem. Speak in a manner suited for historians or history
     students.

    Returns:
    - str: Detailed explanations, historical analyses, or information
    related to the history query.
    """


def philosophy_gpt(query):
    """
    A specialized language model designed to provide expert responses on
    various philosophy-related topics, including formal logic, logical
    fallacies, moral disputes, moral scenarios, and world religions. This
    model is useful for students, educators, and philosophy enthusiasts
    seeking deep philosophical discussions and insights. This model also
    reformats user queries into professional philosophy language.

    Parameters:
    - query (str): A detailed prompt that encapsulates a philosophy-related
     question or problem. Speak in a professional philosopher manner.

    Returns:
    - str: In-depth philosophical analysis or discussions relevant to the
    query.
    """


def law_gpt(query):
    """
    A specialized language model equipped to handle queries related to
    legal studies, including international law, jurisprudence, and
```

```
    professional law. This model serves law students, practicing lawyers,
    and professionals in the legal field needing detailed legal
    explanations or interpretations. This model also reformats user queries
     into professional legal language.

    Parameters:
    - query (str): A detailed prompt that encapsulates a law-related
    question or issue. Speak in a professional legal manner.

    Returns:
    - str: Comprehensive legal analyses, solutions, or information related
    to the law query.
    """


def politics_gpt(query):
    """
    A specialized language model designed to delve into topics related to
    politics and public relations, including high school government and
    politics, security studies, and US foreign policy. This model aids
    political science students, professionals, and enthusiasts in gaining a
     better understanding of political dynamics and theories. This model
    also reformats user queries into professional politics language.

    Parameters:
    - query (str): A detailed prompt that encapsulates a politics-related
    question or discussion. Speak in a manner suitable for political
    analysts.

    Returns:
    - str: Detailed political analysis, insights, or information pertaining
     to the politics query.
    """


def culture_gpt(query):
    """
    A specialized language model designed to explore cultural and societal
    topics, particularly focusing on human sexuality and sociology. This
    model is ideal for cultural studies students, sociologists, and anyone
    interested in understanding the dynamics of human societies and
    cultures. This model also reformats user queries into professional
    sociocultural analyst language.

    Parameters:
    - query (str): A detailed prompt that encapsulates a culture-related
    question or topic. Speak in a professional sociocultural analyst manner.


    Returns:
    - str: Detailed cultural insights, analyses, or information related to
    the cultural query.
    """


def economics_gpt(query):
    """
    A specialized language model designed to tackle questions and provide
    insights into economics, including econometrics, high school
```

```
    macroeconomics, and high school microeconomics. This model assists
    students, economists, and financial analysts in understanding economic
    theories and applications. This model also reformats user queries into
    professional economics language.

    Parameters:
    - query (str): A detailed prompt that encapsulates an economics-related
     question or problem. Speak in a manner suitable for economists.

    Returns:
    - str: Detailed economic explanations, analyses, or solutions relevant
    to the economics query.
    """


def geography_gpt(query):
    """
    A specialized language model developed to address inquiries related to
    geography, specifically focusing on high school geography. This model
    supports students and educators in understanding geographical concepts,
     theories, and real-world applications. This model also reformats user
    queries into professional geography language.

    Parameters:
    - query (str): A detailed prompt that encapsulates a geography-related
    question or topic. Speak in an educational manner suitable for
    geographers.

    Returns:
    - str: Detailed geographical information, analyses, or insights related
     to the geography query.
    """


def psychology_gpt(query):
    """
    A specialized language model focused on providing expert responses on
    topics related to psychology, including high school psychology,
    professional psychology, and human aging. This model is particularly
    valuable for psychology students, clinicians, and researchers seeking
    to understand various psychological theories and practices. This model
    also reformats user queries into professional psychologist language.

    Parameters:
    - query (str): A detailed prompt that encapsulates a psychology-related
     question or discussion. Speak in a professional psychologist manner.

    Returns:
    - str: In-depth psychological analyses, solutions, or information
    relevant to the psychology query.
    """


def business_gpt(query):
    """
    A specialized language model designed to address topics related to
    business, including business ethics, management, and marketing. This
    model supports business students, professionals, and entrepreneurs in
```

```
    understanding business practices, theories, and market dynamics. This
    model also reformats user queries into professional business language.

    Parameters:
    - query (str): A detailed prompt that encapsulates a business-related
    question or problem. Speak in a professional business manner.

    Returns:
    - str: Detailed business insights, strategies, or information relevant
    to the business query.
    """


def health_gpt(query):
    """
    A specialized language model designed to provide answers and insights
    on health-related topics, including anatomy, clinical knowledge,
    college medicine, medical genetics, nutrition, and virology. This model
     assists medical students, health professionals, and researchers in
    understanding complex medical and health issues. This model also
    reformats user queries into professional medical language.

    Parameters:
    - query (str): A detailed prompt that encapsulates a health-related
    question or issue. Speak in a professional medical manner.

    Returns:
    - str: Detailed medical explanations, solutions, or information related
     to the health query.
    """


def general_gpt(query):
    """
    A general-purpose language model designed to provide answers and
    insights across a wide array of topics not specifically categorized
    under other specialized models. This tool is specifically useful for
    users seeking information on miscellaneous and diverse topics that do
    not fall into the standard academic or professional categories such as
    physics, chemistry, biology, computer science, math, electrical
    engineering, history, philosophy, law, politics, culture, economics,
    geography, psychology, business, or health.

    Parameters:
    - query (str): A general prompt encompassing any topic of interest
    outside the specified categories. Speak in a broad and inclusive manner.


    Returns:
    - str: Comprehensive explanations or information pertaining to the
    general query, ensuring a focus away from the excluded fields.
    """
```