Enhanced Fine-Tuning of Lightweight Domain-Specific Q&A model Based on Large Language Models

Shenglin Zhang*, Pengtian Zhu*, Minghua Ma†, Jiagang Wang‡, Yongqian Sun*,
Dongwen Li*, Jingyu Wang*, Qianying Guo§, Xiaolei Hua§, Lin Zhu§, Dan Pei‡
*Nankai University, † Microsoft,‡ Tsinghua University,§ China Mobile Research Institute

***Abstract*—Large language models (LLMs) excel at general question-answering (Q&A) but often fall short in specialized domains due to a lack of domain-specific knowledge. Commercial companies face the dual challenges of privacy protection and resource constraints when involving LLMs for fine-tuning. This paper propose a novel framework, Self-Evolution, designed to address these issues by leveraging lightweight open-source LLMs through multiple iterative fine-tuning rounds. To enhance the efficiency of iterative fine-tuning, Self-Evolution employ a strategy that filters and reinforces the knowledge with higher value during the iterative process. We employed Self-Evolution on Qwen1.5-7B-Chat using 4,000 documents containing rich domain knowledge from China Mobile, achieving a performance score 174% higher on domain-specific question-answering evaluations than without using Self-Evolution and even 22% higher than Qwen1.5-72B-Chat. Self-Evolution has been deployed in China Mobile's daily operation and maintenance for 117 days, and it improves the efficiency of locating alarms, fixing problems, and finding related reports, with an average efficiency improvement of over 18.6%. In addition, we release Self-Evolution framework code in https://github.com/Zero-Pointer/Self-Evolution.**

***Index Terms*—large language model, question answering, domain alignment, data mining**

I. Introduction

With the emergence of large language models (LLMs) such as Qwen [1], LLaMA [2], and GPT [3], their exceptional generation, understanding of complex language structures and dialogue capabilities have garnered widespread attention [4], [5]. However, in specific domains, their performance often fails to meet practical requirements. For instance, GPT-4 may cite incorrect legal provisions when answering legal questions, leading to erroneous analytical conclusions. ChatLaw-MoE [6], fine-tuned on high-quality law data, has outperformed GPT-4 across multiple application scenarios. Therefore, enabling general models to acquire domain-specific knowledge allows for deploying a domain model with minimal computational resources, potentially outperforming general models with ten times the number of parameters.

State-of-the-art approaches extensively utilize instruction fine-tuning (IFT) to align general-purpose models with specific application domains and maximize their effectiveness. InstructGPT [7] employed instruction fine-tuning to bridge the performance gap between models with a hundredfold difference in parameter count. In the absence of instruction data, certain approaches [8]–[11] use advanced LLMs to construct instruction datasets, achieving performance close to GPT-3.5 and GPT-4. However, these methods cannot guarantee the correctness and diversity of the generated instruction data. Fortunately, high-quality instruction data is scarce in most scenarios, while the volume of knowledge documents is enormous.

In summary, applying general-purpose models to specific domains presents the following challenges:

1) **Limitation of Computational Resources.** Model performance is typically proportional to the scale of the model's parameters. However, fine-tuning and deploying powerful general-purpose language models requires substantial computational resources. For example, a LLM with 72B parameters using fp16 precision requires five Tesla V100-32GB GPUs for inference. Fine-tuning such a model incurs even greater costs. This is prohibitively expensive and impractical for tasks that must be continuously available.
2) **High-quality data scarcity.** Domain-specific high-quality instruction data is often scarce. Manually correcting instruction data requires significant human effort, making it expensive. A solution is needed to automatically construct high-quality data without human assistance.
3) **Lack of diversity and correctness.** Firstly, using a fixed model to construct instruction data tends to generate overly similar data. Additionally, relying solely on the model's internal capabilities for data generation may result in incorrect or irrelevant data for the domain. The model might need more domain understanding or have learned incorrect knowledge, leading to hallucination issues. We hope the model can dynamically learn from unsupervised domain documents, continually improving its capabilities while ensuring the diversity and accuracy of data generation.
4) **Data privacy.** Due to the inclusion of private information in domain-specific data, fine-tuning commercial LLMs poses major challenges when dealing with sensitive internal company data, including privacy leakage and high costs.

In this paper, we propose a novel framework **Self-Evolution** to address the aforementioned challenges. The contributions of

this paper are summarized as follows:

1) Considering the costs and privacy concerns during actual deployment, we select an open-source model with 7B parameters as the data generation, scoring model, and model for QA tasks in real scenarios. All phases in Self-Evolution can be completed with just one Tesla V100-32GB GPU, significantly reducing computational resource requirements. (Addressed challenges 1 and 4.)
2) Self-Evolution uses LLM to generate instrutiondata based on a large number of unlabeled knowledge documents, ensuring domain relevance and correctness while avoiding the need for manual assistance. Additionally, the LLM undergoes iterative updates, generating a new batch of data each time. This process ensures diversity between different batches of data. (Addressed challenges 2 and 3.)
3) We conducted extensive evaluation experiments using real-world data from China-Mobile, a top-tier telecommunications provider providing services for one billion+ monthly active users (MAU). Self-Evolution achieves a performance score 174% higher on domain-specific question-answering evaluations than without using Self-Evolution and even 22% higher than Qwen1.5-72B-Chat. The Self-Evolution has been deployed in China Mobile's daily operation and maintenance for 117 days.

II. Related Work

A. Instruction Fine-tuning

The potential of LLMs in the specific domain is vast and promising. For example, Microsoft deployed GPT to summarize anomalous events in its services [12]. However, as task complexity and requirements increase, instruction fine-tuning (IFT) is widely adopted to enhance model performance. FLAN [13] achieved significant improvements in generalization by fine-tuning a high-quality instruction dataset. InstructGPT [7] successfully aligned GPT-3 [3] with human intent by fine-tuning a dataset rich in real-world instruction forms and task types. OWL [14] collected numerous operation domain instructions and achieved remarkable results in log parsing and anomaly detection. However, these methods require a large amount of manually annotated data, which becomes a bottleneck for widespread application due to the high cost.

B. Instruction Data Generation

Researchers have extensively explored methods to reduce human involvement in generating instruction data. Some methods [8]–[10], [15] use advanced commercial models to create instruction datasets. For instance, Alpaca [8] uses a small amount of manually constructed data to extract knowledge from DaVinci-003 [16], creating a 52k instruction dataset. It fine-tunes LLaMA to achieve performance close to GPT-3.5, significantly reducing the cost of obtaining instruction datasets. Peng et al. [9] extract knowledge from GPT-4, resulting in higher quality and more diverse responses.

Another class of methods [11], [17], [18] employs a self-guided approach. These methods extract knowledge from the model and then use this newly constructed data to enhance domain or task capabilities. Self-Instruct [17], for instance, proposes using self-generated samples to enhance the instruction-following ability of pre-trained language models. Self-Align [18] mainly adopts topic-guided red-blue adversarial self-guidance and principle-driven self-calibration to construct data and fine-tune models, requiring less than 300 lines of manually constructed data (including 195 seed prompts, 16 principles, and five examples) to achieve high-quality fine-tuned model. The potential of these self-guided methods is certainly worth exploring further.

However, these methods still require manually constructed supervision data and are limited by the model's inherent knowledge constraints, preventing them from generating instruction data beyond the model's capabilities.

C. Instruction Data Selection

In the early stages of IFT research, many works improved model capabilities by building large instruction datasets. However, LIMA [19] proposed that "less alignment is more" showing that fine-tuning the model with only 1,000 high-quality samples can achieve a performance comparable to GPT-4. Appropriate data filtering strategies can improve learning efficiency and help reduce hallucinations caused by overtraining [2], [20].

ALPAGASUS [21] uses ChatGPT for scoring, though this method may overlook the target model's capabilities and lacks interpretability. The forgetting score [22] tracks changes in sample classification during training. GraNd [23] prunes data based on the gradient norm of the sample. While the forgetting score and GraNd require significant overhead since they need to continuously update the scoring model, thus increasing the overall model training time.

Instruction Following Difficulty (IFD) [24] stands out for its efficiency, using the representation features of the target model to identify high-quality instruction data. It provides a simpler, cheaper, and interpretable approach by computing the generation complexity of the answer using a single fixed scoring model.

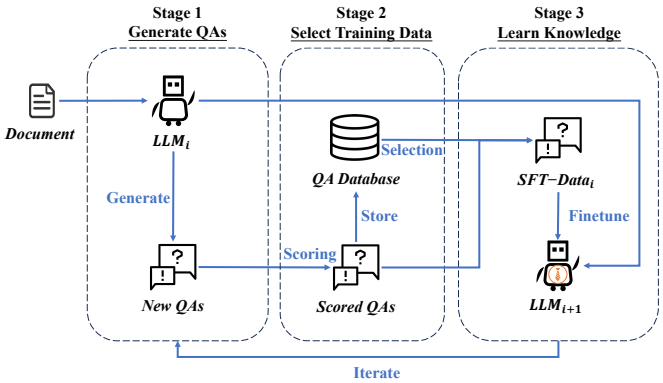

Fig. 1. Self-Evolution

III. METHOD

The overview of Self-Evolution is illustrated in Fig. 1. To start, Self-Evolution requires a LLM θ_0 as the initial QA[1] generation model and scoring model and a collection of domain-related documents T. Self-Evolution consists of three phases. In the first stage, the QA generation model generates QA pairs based on the domain-related documents. In the second phase, the scoring model and a scoring metric are employed to identify valuable samples from all historical instruction QA pairs. In the third phase, these valuable instruction samples are used to conduct a new round of IFT, reinforcing the model's domain knowledge. These three phases iterate continuously until the desired performance is achieved. The following sections will provide a detailed description of these phase.

A. QA generation

More new QA data are generated in the QA generation phase. Self-Evolution constructs new questions and answers based on each domain-related document rather than deriving them from manually constructed questions.

This questions generation process is represented as $q_{ij} = LLM(\theta_i, t_j)$, where t_j is the j-th document in T. During this process, we design delicated prompt to prioritize two key aspects: 1) Question conciseness: Preventing the generation of content with multiple sub-questions, which could lead to model hallucinations (Note 2). 2) Question validity: Ensuring each generated question is answerable (Note 6). The detailed prompt used for question generation is as follows:

TABLE I
QUESTION GENERATION PROMPT.

Domain Knowledge:
Reference document: {Knowledge}
Role Description:
You are an expert in the operations domain.
Based on your comprehensive knowledge and the information provided above.....
Rules Description:
Note 1: The question should be as concise as possible.
Note 2: The question should not contain multiple sub-questions, only one question is permitted.
.....
Note 6: Do not output declarative sentences; it must be a question!
Please formulate a question now.
Question:

This answer generation process is represented as $a_{ij} = LLM(\theta_i, t_j, q_{ij})$. Incorporating t_j, ensures that the questions are correctly answered. In this process, we emphasize response completeness, ensuring that the generated content is a complete answer rather than one containing pronouns referring back to the document. The prompt used for answer generation is as follows:

After obtaining the newly generated questions and answers, Self-Evolution combines them into new instruction data $D_i = \{(q_{i0}, a_{i0}), (q_{i1}, a_{i1}), \ldots, (q_{i|T|}, a_{i|T|})\}$.

[1]As the instruction data in this paper consistently takes the form of question-answer pairs, the terms "instruction data", "QA" and "question-answer pairs" are used interchangeably in the following text.

TABLE II
ANSWER GENERATION PROMPT.

Role Description:
You are an expert in the field of operations.....
You must generate responses based on the requirements.
Workflow Description:
1. Receive and parse the user's question.
2. Read and analyze the document provided by the user.
3. Provide a concise and comprehensive answer by combining your knowledge with the document content.
In Context Learning:
Examples:
Question: Which is the largest planet in the solar system?
Knowledge fragment: The solar system consists of eight planets, with Jupiter being the largest. Its mass is 2.5 times that of all other planets combined.
Answer: The largest planet in the solar system is Jupiter.
Warnings:
Your answer will be sent independently of the document after generation.....
Your response must ensure two points: conciseness and accuracy.
Domain Knowledge and Question:
Question: {Question}
Knowledge fragment: {Knowledge}

B. Data Selection And Training

Prior to conducting the i-th round of IFT, we can filter and select a subset of instruction data from the previous $i-1$ rounds to enhance the training process. Self-Evolution employs the IFD metric [24] to identify more valuable instruction data. Equation 3 represents the calculation method for the IFD score, while Equations 1 and 2 denote the Conditioned Answer Score and Direct Answer Score, respectively.

$$s_\theta(A \mid Q) = -\frac{1}{N}\sum_{i=1}^{N} \log P\left(w_i^A \mid Q, w_1^A, \ldots, w_{i-1}^A; \theta\right) \quad (1)$$

$$s_\theta(A) = -\frac{1}{N}\sum_{i=1}^{N} \log P\left(w_i^A \mid w_1^A, w_2^A, \ldots, w_{i-1}^A; \theta\right) \quad (2)$$

$$\mathrm{IFD}_\theta(Q, A) = \frac{s_\theta(A \mid Q)}{s_\theta(A)} \quad (3)$$

The Conditioned Answer Score quantifies a model's ability to produce responses that align with both the given instructions and the correct answers. It assesses the model's output congruence with the directive and the expected solution. The Direct Answer Score evaluates the LLM's capacity to independently generate correct answers, reflecting the answer's intrinsic complexity in the absence of contextual instructions. A high IFD score indicates the model's difficulty in aligning responses with instructions, thereby highlighting the instruction's complexity.

Therefore, Self-Evolution extract k instruction data with the highest IFD scores from $D_0, D_1, \ldots, D_{i-1}$ to form IFD_i. This set IFD_i is then combined with D_i for the i-th round of training, leveraging historical high-quality data alongside newly generated data, potentially enhancing the efficiency and effectiveness of each training iteration.

C. Next Iteration

Self-Evolution employs a model self-evolution scheme. To elucidate the principles underlying this scheme, we define a scoring function $score = f(q, a)$ that evaluates the quality

of an answer a with respect to a question q. As previously mentioned, $a = LLM(\theta_i, q)$ denote the response of model θ_i to q, and $a = LLM(\theta_i, t, q)$ represent the response of model θ_i to q given a highly relevant knowledge document t. We define $\theta_{i+1} = IFT(\theta_i, q, a)$ as the next-generation model θ_{i+1} resulting from fine-tuning θ_i on the instruction data pair (q, a). We leverage In-context Learning [25] to establish the first inequality:

$$f(q, LLM(\theta_i, q)) \leq f(q, LLM(\theta_i, t, q)) \tag{4}$$

This inequality demonstrates that the instruction data $((q, LLM(\theta_i, t, q))$ provides valuable learning opportunities for model θ_i. Consequently, we derive θ_{i+1} through $\theta_{i+1} = IFT(\theta_i, q, a)$. Post-training, we obtain the second inequality:

$$f(q, LLM(\theta_i, q)) \leq f(q, LLM(\theta_{i+1}, q)) \tag{5}$$

Thus, model θ_i completes one iteration of evolution. The iterative process can be terminated by setting an iteration threshold. Empirically, this threshold is proportionally related to the model's parameter count and inversely related to the data volume. Smaller parameter counts tend to be more susceptible to hallucinations [26], necessitating threshold adjustments based on both parameter count and data volume.

IV. Experimental Setup

A. Model and Dataset

The base model selected for our experiments is Qwen1.5-7B-Chat [1], denoted as θ_0. We use the LoRA (Low-Rank Adaptation) [27] method to fine-tune models. The LoRA hyperparameters are configured as follows: lora-rank is set to 4, and lora-alpha is set to 8. Notably, we set lora-target to "all" [11], which enables us to achieve superior training results. The model chosen for IFD scoring is Qwen1.5-7B-Chat, denoted as θ_{ifd}. It is important to note that θ_{ifd} does not participate in the subsequent training process. Its parameters remain fixed throughout the iteration process, ensuring consistent scoring criteria in each round of evaluation.

We select 4,000 valuable internal knowledge documents from China Mobile, denoted as T, where $|T| = 4000$. As shown in Table III, T contains crucial operational knowledge such as alert analysis, configuration analysis, and operational experience, enabling operation engineers to quickly familiarize themselves with and solve problems. These knowledge documents are incorporated into the training process. Specifically, they are converted into corresponding instruction data and subsequently used for IFT.

We collected 100 real-world question-answer pairs related to these documents, as shown in Table IV. These pairs correspond to the knowledge that operations engineers need to acquire on an ad hoc basis during their work. Due to their close association with the knowledge contained in the documents, we consider this set as a test set to evaluate the model's performance.

TABLE III
Example for Knowledge Document.

Alarm: {Alarm instance}
Alarm explanation:
-{This is the reason for the alarm to appear}
-{This is the condition for the alarm to be cleared}
-{This is the specific threshold for the occurrence and resolution of alarms}
Possible reasons:
-Reason 1: {This is the first possible reason that may occur}
-Reason 2: {This is the second possible reason that may occur}
Processing steps:
-Reason 1:
-{Step 2 of reason 1}
-{Step 2 of Reason 1}
-Reason 2:
-{Step 1 of Reason 2}

TABLE IV
Example for Question and Answer.

Question:
How to gradually troubleshoot and solve the problem when device A starts and device B cannot function properly?
Answer:
When the alarm of B not working properly appears after device A is started, the following steps can be followed for troubleshooting and handling:
1. Check component C.
-If C is firm, proceed to step 3.
-If it is not secure, try reinstalling component C.
2. After reinstalling component C, check if the alarm disappears.
-If the alarm disappears, the problem has been resolved, and the process ends.
-If the alarm still exists, proceed to step 3.
3. Check if component C is damaged.
-If damaged, proceed to step 4.
-If not damaged, please contact technical personnel.
4. Replace component C with a new one and check if the alarm is cleared.
Throughout the entire process, it is essential to backup data before operation to prevent data loss.

B. Baseline

1) Qwen1.5-7B-Chat-Fine-Tuned by High Quality QA: The Qwen1.5 series of language models has demonstrated exceptional performance in the Chinese language domain [1], with Qwen1.5-72B-Chat achieving capabilities comparable to GPT-3.5 on certain tasks. Consequently, we utilized Qwen1.5-72B-Chat to generate 4,000 high-quality question-answer pairs following the approach outlined in Section III-A. These pairs were subsequently used to train a Qwen1.5-7B-HQ model for evaluation purposes, denoted as θ_{HQ}. This methodology of extracting knowledge from documents using a superior model emulates the industrial scenario of constructing IFT data from operational documentation, which often yields favorable results [8], [10].

2) Original LLM: We employed the untrained Qwen1.5-7B-Chat and Qwen1.5-72B-Chat models in our evaluation to

TABLE V
Comparison of different baseline methods.

Model Name	Is Aligned?	Data Source
Qwen1.5-7B-HQ	Yes	Generated by Qwen1.5-72B-Chat with documents
Qwen1.5-7B-Chat	No	-
Qwen1.5-72B-Chat	No	-
GPT-3.4	No	-

simulate the scenario of using open-source models directly for domain-specific question answering. Additionally, we included GPT-3.5 in our evaluation to simulate the scenario of utilizing a closed-source model for domain-specific question answering.

V. Evaluation Metrics

We use the BLEU [28] score of θ_{HQ}, denoted as $BLEU(\theta_{HQ})$, as a benchmark score and calculate the relative scores of other models in comparison to it. The performance score for a model θ is calculated as:

$$Score = \frac{BLEU(\theta)}{BLEU(\theta_{HQ})} \tag{6}$$

To better illustrate the differences between methods, we let θ_{HQ} serves as a target model for comparison. We collected 100 valuable subjective questions internally from China Mobile, which are related to the knowledge documents T. These questions can reflect the model's learning of T through question-answering performance. This score represents how closely a given model's performance in the domain-specific task approaches that of the optimally fine-tuned model θ_{HQ}.

VI. Experimental Results

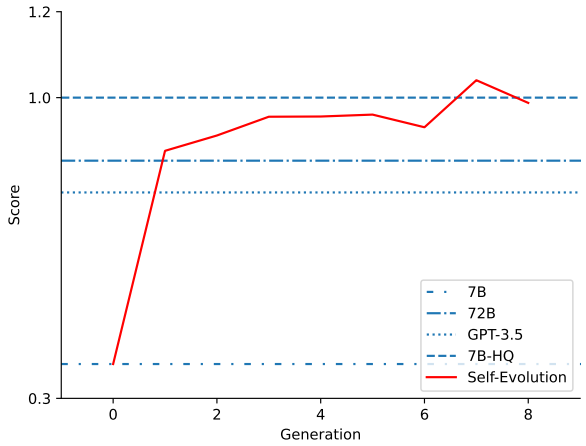

Fig. 2. The x-axis represents the number of iterations of the Self-Evolution, while the y-axis shows the performance scores of different models. The horizontal lines in the graph represent the performance of four distinct models, and the line graph depicts the performance of the Self-Evolution at each iteration.

We compare Qwen1.5-7B-Chat, trained using Self-Evolution, with multiple baseline models. As shown in Figure 2, untrained models perform poorly in domain-specific knowledge question answering tasks. The θ_{HQ} model, fine-tuned with high-quality data, demonstrates excellent performance. Notably, Self-Evolution surpasses the performance of both GPT-3.5 and Qwen1.5-72B-Chat in its first iteration. As the iterations progress, the model's performance gradually approaches that of θ_{HQ}, ultimately surpassing it by the seventh round. Based on the above experiments, we can conclude that Self-Evolution enables Qwen1.5-7B-Chat to surpass the performance of Qwen1.5-72B-Chat-assisted alignment. This demonstrates the effectiveness of the proposed method.

VII. Ablation Experiment

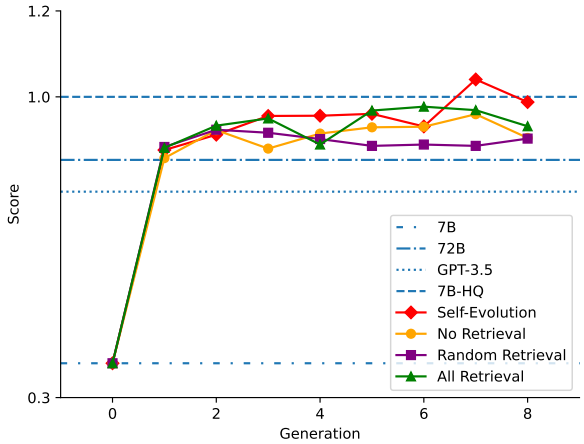

Fig. 3. Ablation Experiment Results.

A. Historical Data Retrieval Module

One of the core components of Self-Evolution is the historical data retrieval module. To investigate its specific role, we designed targeted experiments. After generating the instruction data D_i in the i-th iteration, instead of performing historical data retrieval, we directly used it as the complete training dataset. The results, as shown in Figure 3, indicate that the iterated model failed to surpass the performance of HQ, and the training effectiveness was compromised to some extent. This demonstrates that historical instruction data is valuable and needs to be retrieved and relearned.

B. Historical Data Retrieval Strategy

To validate the effectiveness of using IFD scores for efficient historical instruction data filtering in Self-Evolution, we designed two experiments.

To demonstrate the rationality of our data filtering motivation, the corresponding experiment employed a full retrieval strategy during the recall phase, using all previously generated data as a part for training. The results, as shown in Figure 3, indicate that the performance of the iterated model rapidly deteriorated. Due to repeated training on excessive low-quality data, the model quickly began to exhibit hallucinations. Moreover, in the eight-iteration experiment, the total training time for full retrieval strategy was approximately three times that of Self-Evolution. This proves that discarding a portion of the data not only accelerates training speed but also enhances training effectiveness.

To demonstrate the superiority of our data filtering strategy, we designed an experiment using a random retrieval strategy during the recall phase, where k instruction data were randomly recalled from historical instruction data and

added to the training set. As shown in Figure 3, performance improvements were observed only in the first two generations of the iteration process, with continued training producing negative effects. This proves that an appropriate data filtering strategy is necessary, as an unstable retrieval strategy can lead to a decline in model performance.

In the aforementioned experiments, we tested three alternative approaches: removing the historical data retrieval module, employing a full retrieval strategy, and using a random retrieval strategy. All of these approaches resulted in some degree of performance degradation compared to Self-Evolution. These results demonstrate that the data retrieval module in Self-Evolution is essential, and the data filtering strategy centered on IFD plays a crucial role in the method's effectiveness.

VIII. Conclusion and Future Work

In this paper, we address a key challenge in applying LLMs to the specific domain: the difficulty in utilizing vast amounts of unlabeled knowledge documents. To tackle this issue, we employ self-Alignment in Self-Evolution to rapidly construct a large volume of Instruction data. As the iteration progresses, both the model's capabilities and the quality of generated data improve. To maximize the utilization of instruction data generated in each iteration, we use IFD scores to filter out high-quality data to assist in training. In the China Mobile business question-answering evaluation, our approach, using only a 7B model throughout, outperforms solutions assisted by 72B models, conserves a significant amount of computational resources.

In current business scenarios, multi-turn dialogue capabilities are becoming increasingly important. Therefore, in future work, we plan to extend Self-Evolution to improve the model's domain-specific multi-turn dialogue capabilities using only unsupervised text data.

References

[1] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang *et al.*, "Qwen technical report," *arXiv preprint arXiv:2309.16609*, 2023.

[2] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[5] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.

[6] J. Cui, Z. Li, Y. Yan, B. Chen, and L. Yuan, "Chatlaw: Open-source legal large language model with integrated external knowledge bases," *arXiv preprint arXiv:2306.16092*, 2023.

[7] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[8] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, "Stanford alpaca: An instruction-following llama model," 2023.

[9] B. Peng, C. Li, P. He, M. Galley, and J. Gao, "Instruction tuning with gpt-4," *arXiv preprint arXiv:2304.03277*, 2023.

[10] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing, "Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality," March 2023. [Online]. Available: https://lmsys.org/blog/2023-03-30-vicuna/

[11] C. Xu, D. Guo, N. Duan, and J. McAuley, "Baize: An open-source chat model with parameter-efficient tuning on self-chat data," *arXiv preprint arXiv:2304.01196*, 2023.

[12] P. Jin, S. Zhang, M. Ma, H. Li, Y. Kang, L. Li, Y. Liu, B. Qiao, C. Zhang, P. Zhao *et al.*, "Assess and summarize: Improve outage understanding with large language models," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2023, pp. 1657–1668.

[13] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," *arXiv preprint arXiv:2109.01652*, 2021.

[14] H. Guo, J. Yang, J. Liu, L. Yang, L. Chai, J. Bai, J. Peng, X. Hu, C. Chen, D. Zhang, xu Shi, T. Zheng, liangfan zheng, B. Zhang, K. Xu, and Z. Li, "OWL: A large language model for IT operations," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=SZOQ9RKYJu

[15] C. Xu, Q. Sun, K. Zheng, X. Geng, P. Zhao, J. Feng, C. Tao, and D. Jiang, "Wizardlm: Empowering large language models to follow complex instructions," *arXiv preprint arXiv:2304.12244*, 2023.

[16] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[17] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi, "Self-instruct: Aligning language models with self-generated instructions," *arXiv preprint arXiv:2212.10560*, 2022.

[18] Z. Sun, Y. Shen, Q. Zhou, H. Zhang, Z. Chen, D. Cox, Y. Yang, and C. Gan, "Principle-driven self-alignment of language models from scratch with minimal human supervision," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[19] C. Zhou, P. Liu, P. Xu, S. Iyer, J. Sun, Y. Mao, X. Ma, A. Efrat, P. Yu, L. Yu *et al.*, "Lima: Less is more for alignment," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[20] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin *et al.*, "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions," *arXiv preprint arXiv:2311.05232*, 2023.

[21] L. Chen, S. Li, J. Yan, H. Wang, K. Gunaratna, V. Yadav, Z. Tang, V. Srinivasan, T. Zhou, H. Huang *et al.*, "Alpagasus: Training a better alpaca with fewer data," *arXiv preprint arXiv:2307.08701*, 2023.

[22] M. Toneva, A. Sordoni, R. T. d. Combes, A. Trischler, Y. Bengio, and G. J. Gordon, "An empirical study of example forgetting during deep neural network learning," *arXiv preprint arXiv:1812.05159*, 2018.

[23] M. Paul, S. Ganguli, and G. K. Dziugaite, "Deep learning on a data diet: Finding important examples early in training," *Advances in Neural Information Processing Systems*, vol. 34, pp. 20 596–20 607, 2021.

[24] M. Li, Y. Zhang, Z. Li, J. Chen, L. Chen, N. Cheng, J. Wang, T. Zhou, and J. Xiao, "From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning," *arXiv preprint arXiv:2308.12032*, 2023.

[25] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023.

[26] Z. Gekhman, G. Yona, R. Aharoni, M. Eyal, A. Feder, R. Reichart, and J. Herzig, "Does fine-tuning llms on new knowledge encourage hallucinations?" *arXiv preprint arXiv:2405.05904*, 2024.

[27] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[28] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.