SuperFC: Selective Data Utilization for a Sustainable and Effective Function-Calling Agent

Xinhua Yang[*], Yilun Liu[†][✉], Shimin Tao[†], Chunguang Zhao[†], Weibin Meng[†], Minggui He[†],
Chang Su[†], Rongrong Liu[†], Hongxia Ma[†], Li Zhang[†], Jingzhou Du[†],
Duan Li[†], Jian Gao[†], Hao Yang[†], Boxing Chen[‡], Chuanwen Li[*][✉]
[*]Northeastern University, China
[†]Huawei, China
[‡]Huawei Canada, Canada
[✉] liuyilun3@huawei.com, lichuanwen@mail.neu.edu.cn

***Abstract*—The function-calling agent is obtained by performing agent tuning to the large language model (LLM) on function-calling dataset. However, even state-of-the-art datasets (e.g., xlam-function-calling-60k datasets) still contain numerous misleading examples of low-quality data, wasting significant computational resources and result in an unnecessary carbon footprint. Furthermore, such inductive bad data negatively impacts the performance of the agent. In this paper, we propose a set of scoring criteria specifically tailored to evaluate function-calling data and use these criteria to develop a data filtering framework. By applying this framework to filter out low-quality data, we fine-tuned SuperFC, which demonstrates substantial improvements in both sustainability and performance. The SuperFC-7B training process reduced training time from 455 minutes to 85 minutes, resulting in a 80.02% reduction in carbon footprint. Simultaneously, fine-tuning on high-quality data subsets led to performance improvements of up to 3.68%. Additionally, we provide an in-depth analysis of the causes behind the low quality of synthetic function-calling data, offering valuable insights for future data synthesis in this domain. We have also released a high-quality function-calling dataset, available at: https://anonymous.4open.science/r/SuperFC-2F2D**

***Index Terms*—function-calling, sustainable, AI agent**

I. Introduction

An AI agent is an autonomous entity capable of perseiving its environment, making decisions, and taking actions [1], [2]. While traditional AI agents excel in specific domains, they often struggle with adaptability and generalization. In recent years, with the intense development of Large Language Models (LLMs), function-calling agents centered around LLMs have had a significant impact on the field of artificial intelligence. Closed-source models like GPT-4 [3], Gemini [4], and Mistral [5] not only understand and generate human-aligned textual content but also perform function calls based on natural language instructions. In contrast, open-source LLMs have gained widespread popularity in both academia and industry due to their transparency, customizability, and strong community support. Although these models demonstrate superior capabilities in instruction following, reasoning, planning, and tool utilization after alignment training [6], [39], they still fall short compared to closed-source models. Agent tuning, a streamlined and versatile method for supervised fine-tuning (SFT), enhances the agent capabilities of open-source LLMs while preserving their general functionality [8]. Researchers have applied this technique to improve the overall performance of function-calling agents built on open-source models [22], [23], thus boosting their versatility and operational efficiency. As a result, the development of open-source models has made significant progress. However, we have identified the following limitations in the existing methods:

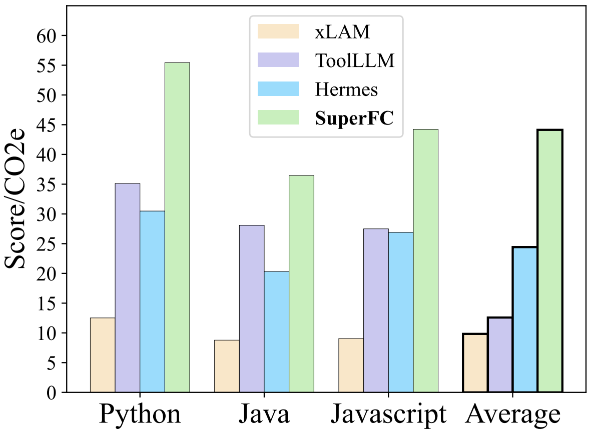


Fig. 1: Comparison of **SuperFC (ours)** and existing methods from the perspective of sustainable AI (performance per unit of carbon emission) in representative function-calling tasks from BFCL (Berkeley Function Calling Leaderboard [7]), where all involved functions are based on real-world programming languages. The metric $Score/CO_2e$ is computed as $\frac{task\ score}{carbon\ footprint}$, highlighting sustainability and effectiveness.

(1) Previous work has resulted in unsustainable function-calling agent. Building on the success of APIGen [9], prior methodologies predominantly employed a combined strategy of API sampling and model-generated data synthesis to refine open-source models by agent tuning. This approach has indeed augmented the performance and adaptability of these models. Nevertheless, the promise of such advancements is shadowed

[✉] Corresponding author.

by persistent issues related to data quality. As we all know, AI is itself a significant emitter of carbon [11], so a critical drawback of utilizing suboptimal datasets for training is the inefficient use of computational resources, leading to an unsustainable environmental footprint [12]–[14]. The training process becomes not only less productive but also more resource-intensive, squandering processing power and energy that could otherwise be allocated to higher-value activities. In the context of large-scale machine learning endeavors, where resource allocation is paramount, this inefficiency poses a significant concern, as it can needlessly increases the carbon footprint and limit potential gains in productivity and innovation.

(2) Uneven data quality leads to ineffective function-calling agent. In the practice of model performance optimization, data quality is often greater than quantity [15]. LIMA [16], CaR [17], and several other studies [18]–[20] have demonstrated that including suboptimal data in the training dataset can hinder the full potential of large language models (LLMs), preventing them from achieving optimal performance. Although progress has been made in organizing function call data in JSON format, improving the readability and structure of the data [9], a challenge of ensuring content quality of synthetic function-calling training data, still persists (*e.g.*, alignment with the user's intention and restrictions are not strictly followed, as shown in Fig 2). This task requires considerable human effort for manual review and curation, along with significant financial investment to support the infrastructure and personnel needed to carry it out [17]. This challenge is especially acute when considering the scale at which LLMs operate, where the data volume can be enormous.

To address the challenges of sustainability and effectiveness, we have concentrated on differentiating high-quality data from suboptimal data during the agent tuning process, aiming to develop a more powerful and sustainable function-calling agent. First, we performed an in-depth analysis of the function-calling dataset, which revealed five key dimensions that significantly influence the performance and reliability of function-calling agents. This analysis led to the development of a comprehensive set of criteria for evaluating and distinguishing the quality of function-calling data. Building on these criteria, we implemented an LLM-assisted data filtering framework as part of our streamlined and efficient data processing strategy. This approach incorporates fine-tuning with high-quality datasets, carefully selected through a rigorous methodological process. Upon applying this process to the 60k dataset generated by APIGen [9], we identified a considerable proportion of the data as having suboptimal quality. We adopted the method of Green Algorithms [10] and Open LLM Leaderboard [21] to estimate the carbon footprint of LLMs. Using the formulas, a function-calling agent has been trained on our meticulously curated dataset demonstrated notable advantages in both resource efficiency and performance, as illustrated in Fig. 1. Our approach reduces the training time from 455 minutes to just 85 minutes on 4× NVIDIA L40 (48GB) GPUs, resulting in a 80.02% reduction in carbon footprint. Furthermore, we validate the versatility of our method by demonstrating its effectiveness across a range of base models (*e.g.*, Qwen2-1.5B and Qwen2-7B) and LLM filters (*e.g.*, GPT-4 and Claude-3). This finding is particularly promising because it highlights how the use of selected high-quality data in agent tuning can effectively reduce the carbon footprint while further unlocking model potential. Our contributions are as follows.

- We introduce an efficient, automated filtering framework that optimizes agent tuning data selection, reducing the carbon footprint during training by up to 80.02%. By prioritizing high-quality, relevant data, the framework minimizes computational resources, accelerates training, and lowers energy consumption, fostering sustainability in machine learning.
- We propose a set of criteria for assessing and differentiating function-calling data quality, using it to curate fine-tuning data that improves performance by 2.17% to 3.68% over the original approach.
- We open-source a high-quality agent tuning dataset curated by our framework, offering a valuable resource for future industrial and academic use, and promoting transparency, reproducibility, and further advancements in the field.

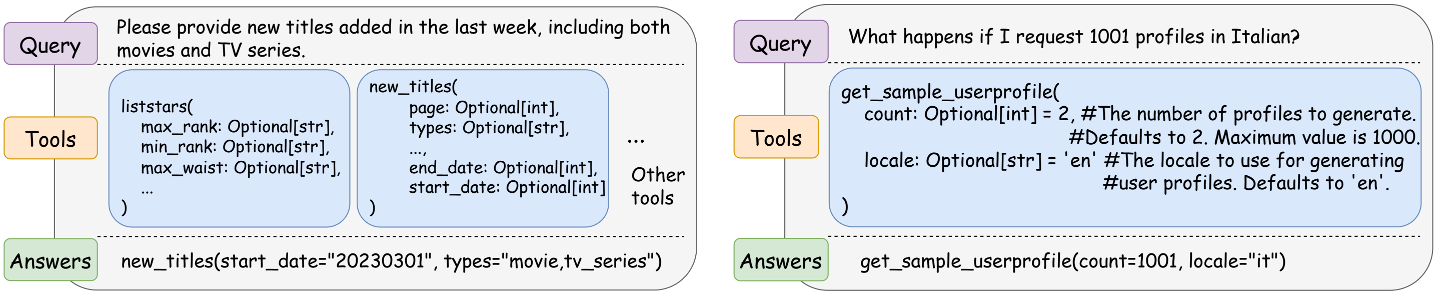
II. Related Work

A. LLM as Aegnt

Autonomous agents have consistently been viewed as a key pathway toward achieving artificial general intelligence (AGI), with the potential to perform tasks through self-guided planning and decision-making. Earlier research often relied on agents operating with simple, heuristic-based policy functions, typically trained within constrained and isolated environments. These prior models, while foundational, limited the agents' ability to adapt and generalize beyond predefined conditions [25]–[28]. In recent years, LLMs have achieved remarkable progress, showcasing considerable potential in reaching human-like intelligence [3], [4], [29]–[31]. This success is driven by the use of extensive training datasets and a large number of model parameters. Building on this foundation, there has been increasing research exploring the use of LLMs as central controllers for creating autonomous agents capable of human-like decision-making [32]. Unlike reinforcement learning, LLM-based agents possess a broader internal knowledge base, allowing them to make more informed decisions even without being trained on domain-specific data. Furthermore, LLM-based agents offer natural language interfaces for human interaction, providing greater flexibility and transparency in their actions.

B. Data-centric AI

Recently, the role of data in AI has been significantly magnified, giving rise to the emerging concept of data-centric AI. The attention of researchers and practitioners has gradually shifted from advancing model design to enhancing the quality and quantity of the data [33]. As noted by Chu et al. [34], effective data management across multiple domains requires data cleaning processes that are both highly automated and



Please provide new titles added in the last week, including both movies and TV series.

```
liststars(
    max_rank: Optional[str],
    min_rank: Optional[str],
    max_waist: Optional[str],
    ...
)
```

```
new_titles(
        page: Optional[int],
        types: Optional[str],
        ...,
        end_date: Optional[int],
        start_date: Optional[int]
)
```

...
Other tools

```
new_titles(start_date="20230301", types="movie,tv_series")
```

What happens if I request 1001 profiles in Italian?

```
get_sample_userprofile(
    count: Optional[int] = 2, #The number of profiles to generate.
                              #Defaults to 2. Maximum value is 1000.
    locale: Optional[str] = 'en' #The locale to use for generating
                                  #user profiles. Defaults to 'en'.
)
```

```
get_sample_userprofile(count=1001, locale="it")
```

(a) The answer not aligned with query intent

(b) The function call is not strictly restricted

Fig. 2: Typical low-quality cases in xlam-function-calling-60k dataset.

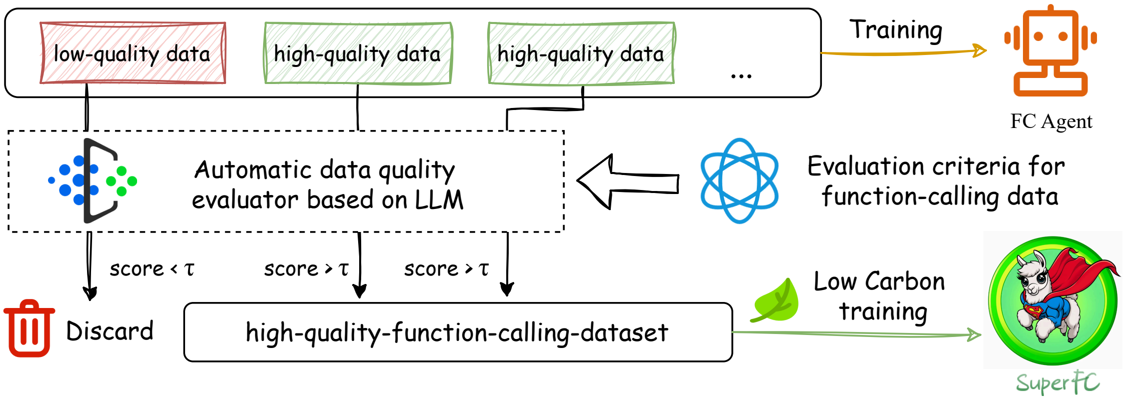


Fig. 3: The agent tuning framework of SuperFC. We use an LLM as an auto-grader, prompting it based on our criteria to assign a score (0–10) to each function-calling data point. Then, using the same hyperparameters from the xLAM agent tuning script, we train SuperFC on the filtered data that exceeds a predefined score threshold.

adaptable. The introduction of the Transformer architecture [35] marked a transformative shift in the development of language models. Models such as BERT [36], RoBERTa [37], and BART [43] have leveraged this architecture, incorporating varying numbers of transformer blocks to enhance their capabilities. This development signaled a pivotal shift in natural language processing, redirecting attention from model architecture to the critical role of data itself. Today, state-of-the-art LLMs, like ChatGPT, reflect this evolution, with a strong reliance on user data for Reinforcement Learning from Human Feedback (RLHF) [38], which aligns with the principles of Data-centric AI.

III. Methodology

A. Overview

Given the unique nature of function-calling data, we begin by focusing on accuracy as the primary criterion for evaluating its quality. To ensure a comprehensive assessment, we then extend our evaluation to five additional dimensions that collectively influence the overall quality of the data. These dimensions are alignment with user intent, argument correction, completeness, consistency, and adherence to limitations. By evaluating each data point comprehensively across these criteria, we can identify and filter out those with low scores, thereby ensuring that only high-quality function call data is retained for subsequent processing and analysis. As demonstrated in Fig. 3, our approach, using a smaller, carefully curated subset of 11k high-quality data, results in a significantly improved model, named **SuperFC**. This improvement is achieved using the same training configuration as in the original method.

B. Scoring criteria

SuperFC aims to augments the quality of generated function-calling data, thereby improving model capabilities. This is achieved by establishing a set of rigorous quality criteria and subsequently filtering out data points that fail to meet these standards. A critical prerequisite of this process involves identifying the key dimensions that define a high-quality function-calling data. We achieved this goal by mining the intrinsic problems of the dataset. Our initial dataset was sourced from xlam-function-calling-60k, a widely used function-calling dataset, which contains 60,000 data collected by APIGen [9], an automated data generation pipeline designed to produce verifiable high-quality datasets for function-calling applications. Each data in the dataset is verified through

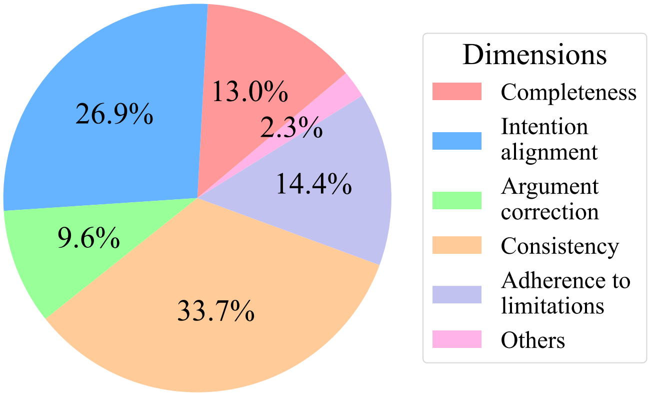


Fig. 4: Analysis of low-quality issues in function-calling data across key dimensions.

three hierarchical stages, ensuring its reliability. We first sample data from 21 different scenarios, each with limited sample sizes. These samples were then distributed to domain experts specializing in function calling, who were tasked with conducting a thorough content quality assessment of sampled data. These assessment included identifying and categorizing instances of low-quality questions. Meanwhile, we leverage an LLM to investigate and analyze quality issues within the entire dataset, and then perform sentiment analysis, text summarization and topic extraction on the model response log to obtain the deep reasons for low quality data. Finally, after collating and merging, the two methods yielded congruent results, from which we obtained the five key quality dimensions. The distribution of occurrences for these dimensions is shown in the Fig. 4. Details of them are listed below:

1) Intention alignment: When tasks involve function calls, LLMs must accurately interpret user intent and correctly map these calls to appropriate functions and parameters. Any deviation from this requirement can lead to errors. Frequent errors in the dataset indicate inadequate learning of parameter configurations, likely from limited training data or domain knowledge. Misinterpreted instructions reveal limitations in handling complex or ambiguous commands, necessitating improved semantic understanding and context awareness. The model's incomplete functionality, failing to cover all user scenarios, underscores the need for more diverse training samples and better support for edge cases to enhance generalization.

2) Argument correction: To prevent task failures due to missing or incorrect parameters, enhance preprocessing to avoid incomplete or misformatted inputs and implement strict validation to catch anomalies early. Increasing the model's sensitivity to critical parameters through importance analysis, context-aware processing, feedback loops, and robust error handling ensures key inputs are correctly processed, thereby reducing failures and improving system reliability.

3) Completeness: In function call tasks, executing each step as expected is crucial. Omissions or incomplete tasks degrade user experience and system credibility. Incomplete results indicate poor data structuring. To improve, optimize task descriptions for clarity and enrich dataset annotations, ensuring the model captures and presents key elements accurately, thus enhancing response quality and user satisfaction.

4) Consistency: Clear and consistent communication is vital for guiding the model to use functions correctly. Ambiguous outputs can lead to incorrect decisions and higher cognitive load. Enhancing text normalization in datasets reduces ambiguity and improves consistency. Adequate metadata (e.g., timestamps, locations) enhances context sensitivity. Effective result presentation and visualization are crucial for clarity. Further research should aim to make the model's output more intuitive and understandable.

5) Adherence to limitations: Strict adherence to constraints and robust input-output validation mechanisms are fundamental to building a stable and reliable system. This is especially critical in function call scenarios, where ensuring the security and correctness of each step is paramount. Insufficient input-output validation mechanisms can make the system vulnerable to anomalies or malicious attacks. It is recommended to incorporate more boundary test cases during the data generation process to enhance the model's robustness. Limited ability to follow constraints can lead to decreased correctness and security of responses, which means that data generation must not only accomplish the current task but also maintain strict compliance. For instance, this can be achieved by simulating user requests under various scenarios to enrich the diversity of training samples.

C. Rating and Filtering

Given a generated function-calling dataset D, consisting of items $x = (\text{query}, \text{tools}, \text{answers})$, where each $x \in D$, and an open-source large language model (LLM) θ (for example, Qwen2), our primary objective is to enhance the performance of an function-calling agent by filtering and selecting a high-quality subset $S \subset D$, such that fine-tuning the agent on this subset results in a better function-calling agent θ_S compared to an agent θ_D that is fine-tuned on the entire dataset D. The intuition is that not all data in D contributes equally to improving the agent's capabilities; therefore, selecting the right data is critical.

As shown in Algorithm 1, to select the high-quality subset S, we introduce a three-step process involving automated evaluation using a scoring function $G(\cdot)$. In the first step, we fused our scoring criteria into a targeted, high-quality prompt dedicated to evaluating the quality of function-calling data; Then it will be passed to an API LLM $G(\cdot)$, which serves as an auto-evaluator.

The scoring function evaluates each example $x \in D$ and assigns a score $G(x, p_G)$, where p_G is the specific scoring prompt designed to capture the essential qualities of a high-quality data point. For example, the prompt p_G might ask the LLM to evaluate how well the response aligns with its instruction.

The model G is typically a large, powerful LLM such as GPT-4, which has been shown to exhibit strong evaluative capabilities when fine-tuned on specific tasks. The evaluation

Algorithm 1: Rating and filtering

```
Input  : original dataset $D$, scoring criteria $C$, scoring
         prompt template $p_t$
Output: High-quality subset $S$
Step1: Format the prompt template according to the
  scoring criteria.
$p_G \leftarrow TemplateFormat(p_t, C)$;
for $x \in D$ do
    Step2: With the help of LLM as an automatic
      evaluator to score.
    $score \leftarrow G(x, p_G)$;
    Step3: Preserve the data that meets the criteria.
    if $score > \tau$ then
        $Collect(x)$;
    end
end
$S \leftarrow Collect$;
```

performed by $G(\cdot)$ is expected to provide insights into how likely each data item is to improve agent performance.

Once the scores are assigned to each data point $x \in D$, the next step is to filter the dataset based on these scores. We define a threshold τ, which is used to decide which items are of high enough quality to be included in the subset S. Specifically, we select items x_i whose scores exceed the threshold τ:

$$S \triangleq \{x \in D : G(x, p_G) \geq \tau\} \tag{1}$$

This operation ensures that only those items which meet the predefined quality criteria (as evaluated by the LLM G) are included in the subset S. The threshold τ is a hyperparameter that can be adjusted depending on how conservative or aggressive the selection process should be.

After identifying the high-quality subset S, we proceed with fine-tuning the agent on S, yielding a new model θ_S. The expectation is that training on S will result in an agent that is more capable of handling function-calling tasks compared to the model θ_D. This is because the subset S is curated to include only the most relevant, well-formed, and useful examples, thus improving the efficiency and efficacy of the fine-tuning process. The performance of θ_S is then evaluated using standard metrics for function-calling tasks.

We achieve θ_S by agent tuning θ on S using the same hyperparameters from the xLAM agent tuning script with our framework.

D. Result Filtered from xlam-60k

Correspondingly, we define τ in Eq. (1) as the accuracy threshold for subsequent experiments. The score distribution of the xlam-function-calling-60k dataset is shown in Fig. 5, revealing a strong skew towards higher ratings. Most of the data falls within the range of 8 to 10, with 26,121 instances scoring 8, 19,169 scoring 9, and 11,107 scoring 10. In contrast, lower scores (7 and below) are significantly less frequent, with only 2,689 and 914 instances, respectively. This distribution reflects the effectiveness of the rule-based validation applied to the generated dataset, which substantially minimizes low-level errors.

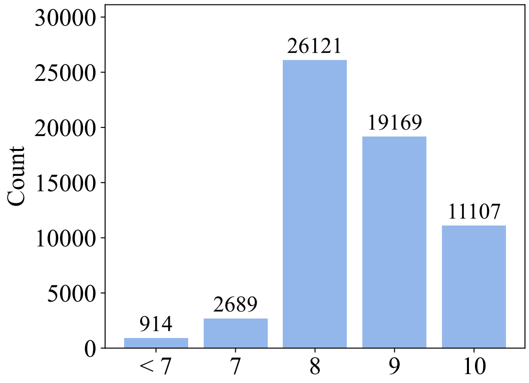


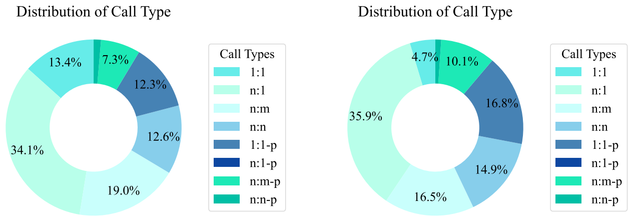
Fig. 5: Histogram of Scores.

In particular, we choose the threshold $\tau = 9.5$ according to the score histogram. For the xLAM dataset D with 60000 samples, this filtering criterion leads to a subset S of 11107 samples, and the scale was reduced by 81% compared with the original dataset.

Inspired by the work of CaR [16] and Phased IFT [40], we specifically examined the diversity of the data before and after filtering in terms of call types and task difficulty.

As shown in Fig. 6, in the original dataset, function call types can be broadly classified into two categories: non-parallel calls and parallel calls. These categories can be further subdivided into eight distinct types based on the number of provided functions and the number of functions actually used. Among these, the most common type involves selecting one function from a set of n functions, and this feature is retained in our selected dataset. However, due to its limited flexibility, this type may lead to forced outputs when the model generates corresponding data, which can negatively impact data quality. As a result, after our filtering process, the proportion of this type of data is reduced. For the multi-select call type, the likelihood of introducing low-quality issues increases because a larger number of functions are involved. This is evident from the data proportions before and after filtering. Consequently, the proportion of other call types increases as a result of the filtering process.

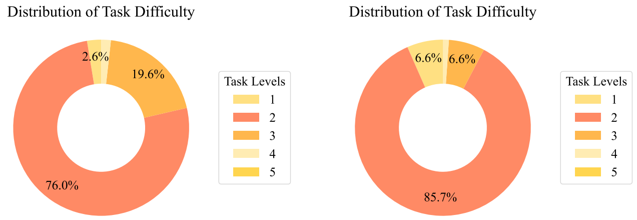
For task difficulty, we adopt the methodology used in Phased IFT [40], where the model is utilized to assess the difficulty of each data instance on a five-point scale. The difficulty levels range from 1 (very easy) to 5 (extremely difficult), as shown in Fig. 7. In our analysis, we were surprised to find that the majority of tasks fall between difficulty levels 2 and 3, indicating that most tasks are of moderate difficulty. Only a small fraction, fewer than a few dozen instances, were rated with the highest difficulty level of 5, suggesting that extremely challenging tasks are relatively rare in the dataset. This distribution pattern is not only observed in the



(a) Full dataset (b) High quality subset

Fig. 6: Distribution comparison of call type.

original dataset but is also evident in our filtered data. The filtering process, which aims to retain high-quality and relevant examples, does not significantly alter the general difficulty distribution. In fact, the proportion of tasks at difficulty levels 2 and 3 remains predominant even after filtering, keep the trend. The observed distribution of task difficulties, with a concentration in the moderate range, may be attributed to the characteristics of the APIs used or the inherent complexity of function-calling tasks themselves. In addition, this distribution may have a significant impact on model performance, and it also highlights the importance of considering task difficulty diversity. Based on this observation, we plan to moderately increase the proportion of more difficult tasks in the subsequent optimization process for function-calling data generation.



(a) Full dataset (b) High quality subset

Fig. 7: Distribution comparison of task difficulty.

To sum up, according to our statistics, the diversity of the original dataset was not destroyed by rating filtering, and the data of various call types and task difficulties were preserved.

IV. Experimental Setup

A. Baseline Models

We compare our SuperFC with the following three recent function-calling agents.

1) ToolLLM: ToolLLM [41] enhances open-source LLMs' ability to execute instructions with external tools through specialized datasets, training, and evaluation methods, aiming to match the performance of SOTA closed-source models like ChatGPT in tool usage.

2) Hermes: Hermes-2-Pro-Mistral-7B [42], built on Nous Hermes 2, is optimized with an updated OpenHermes 2.5 dataset and new function calls via a JSON schema. This upgrade enhances intelligent dialogue systems to better understand and respond to complex user needs more naturally and accurately.

3) xLAM: xLAM [22], a series of large action models, enhances AI agent tasks by providing high-quality datasets and standard protocols. Using a scalable pipeline to integrate diverse datasets, it aims to improve agents' generalizability and performance across environments.

B. Benchmark

We evaluate the performance of the trained models using the Berkeley Function-Calling Benchmark (BFCL) [7], a comprehensive evaluation framework designed to assess the function-calling capabilities of large language models (LLMs) across a wide range of programming languages and application domains. The BFCL includes 2,000 test cases that simulate real-world scenarios, encompassing complex tasks such as parallel processing and multi-function calls. Importantly, the benchmark relies on real-world APIs from multiple programming languages—such as Java, JavaScript, and Python—to rigorously evaluate each model's ability to accurately interpret and execute commands within diverse, real-world contexts. As a robust and scalable benchmark, the BFCL provides valuable insights into function-calling performance and features a leaderboard that highlights the latest advancements in both open-source and commercial LLMs.

C. Evaluation Metrics

1) Sustainability: Green Algorithms [10] is a methodological framework and online tool to estimate and report the carbon footprint of computational tasks reliably and standardly. This tool aims to raise awareness and facilitate greener computation practices. Adopting Green Algorithms as a benchmark ensures that our experiment is environmentally conscious while maintaining scientific rigor. It provides a practical solution to monitor and reduce the environmental impact of computational activities, aligning with global sustainability goals.

In this study, we use **carbon footprint (C)** as a key metric to evaluate the environmental impact of the experimental setup. The carbon footprint is calculated by combining the energy consumption (E) and the carbon intensity (CI) of energy production, as shown in the following equations:

$$C = E \times CI \tag{2}$$

$$E = t \times (n_c \times P_c \times u_c + n_m \times P_m) \times PUE \times 0.001 \tag{3}$$

Here, t represents the runtime of the experiment (in hours), n_c and n_m denote the number of compute nodes and memory modules, respectively, P_c and P_m are their power consumption (in watts), and u_c is the utilization rate of the compute nodes. PUE (Power Usage Effectiveness) is a metric for data center energy efficiency, and the factor 0.001 converts the energy consumption into kilowatt-hours (kWh). By combining these equations, we account for both energy usage and environmental impact, providing a comprehensive evaluation of the carbon emissions of the experiment.

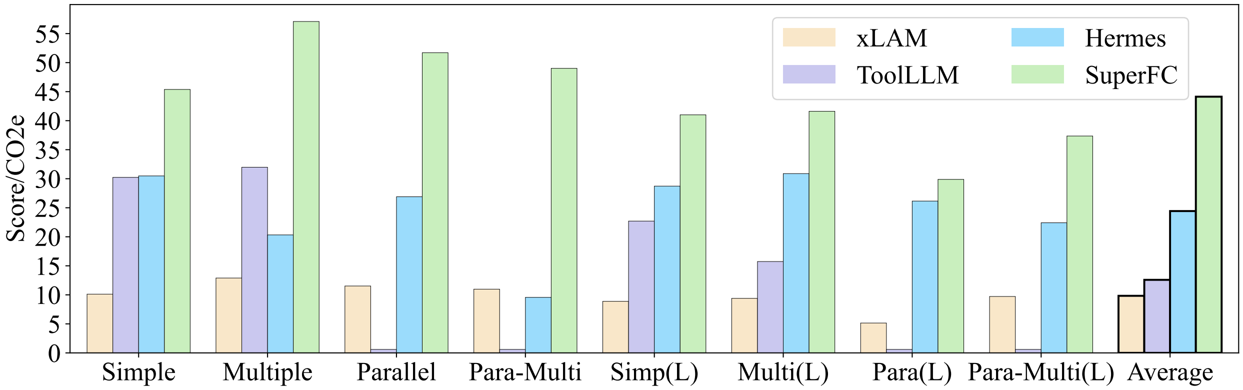


Fig. 8: Comparison of SuperFC and existing methods from the perspective of sustainable AI (performance per unit of carbon emission). The metric $Score/CO_2e$ highlights sustainability and effectiveness. Where "(L)" indicates that this is a Live test.

2) Effectiveness: In general, BFCL evaluation is divided into two categories: **Non-Live** and **Live**. In more detail, Non-live set includes 100 Java, 50 JavaScript, 70 REST API, 100 SQL, and 1,680 Python cases for various simple, parallel, multiple functions calling scenarios. The evaluation JSON functions are scraped and generated from various online sources. The dataset intentionally includes domains such as computing, cloud, sports, law, and others. A total of 40 subdomains of functions are incorporated into the generic evaluations. This approach enables a comprehensive assessment of model performance, not only in data-abundant domains like computing and cloud but also in niche domains such as sports and law. The Live test set incorporates user-contributed queries that are diverse in tone, involve multiple rounds of interaction, cover specialized use cases, and support multiple languages. We also cover rare and challenging cases such as complex function documents with nested parameters and complex user queries that simulate real-world scenarios. This diverse composition ensures a comprehensive evaluation of function execution performance in both common and specialized use cases.

The majority of the each evaluation set is divided into four categories:

- **Simple**: This category involves single function evaluations, representing the simplest but most common format. In this case, the user provides a single JSON function document, and only one function call is invoked.
- **Multiple**: The multiple query style includes user queries that invoke only one function call from a set of 2 to 4 JSON function documents. The agent is required to select the most appropriate function to invoke based on the context provided by the user.
- **Parallel**: Parallel category involves invoking multiple function calls in parallel with a single user query. The agent must determine how many function calls need to be made, and the query can consist of either a single sentence or multiple sentences.
- **Parallel Multiple**: This category is a combination of parallel and multiple functions. The agent is provided with multiple function documents, and each corresponding function call may be invoked zero or more times.

Each category adopts Abstract Syntax Tree (AST) evaluations. The AST evaluation method is a systematic framework designed to assess the function-calling capabilities of large language models (LLMs). This method involves parsing the model-generated function calls into AST structures to extract function names, parameters, and values, which are then compared against the target function documentation and expected answers. The evaluation ensures that function names match correctly, all required parameters are present, and parameter types and values align with the specifications. It includes strict type checking for various data types (e.g., integers, strings, lists, dictionaries) and handles nested structures recursively. For optional parameters, the evaluation allows flexibility in providing default values or omitting them, depending on their specification in the function documentation. The AST evaluation extends seamlessly to multiple or parallel function calls, validating each function individually while maintaining consistency across all outputs. By identifying errors such as mismatched function names, missing parameters, or incorrect values, this method provides a detailed and reliable benchmark for evaluating the accuracy and robustness of function-calling capabilities in LLMs.

V. Experimental Results

A. Sustainability

In Fig. 8, we compare SuperFC with xLAM [22], ToolLLM [41], and Hermes [42]. The horizontal axis represents different types of tasks taken from BFCL (Berkeley Function Calling Leaderboard [7]), including categories such as Simple, Multiple, Parallel, and Parallel-Multiple tasks, as well as their respective large-scale equivalents evaluated on the Live test sets, culminating in the average performance across all task types. The vertical axis measures $Score/CO_2e$, a metric derived from task score and carbon footprint, highlighting both sustainability and effectiveness in model performance.

The results clearly demonstrate that SuperFC outperforms all other agents in every category, achieving the highest $Score/CO_2e$ values across the board. This indicates that SuperFC is not only highly capable of addressing a wide variety of task types but also does so with remarkable efficiency in terms of carbon emissions. Hermes follows as the second-best method in most categories, although it consistently lags behind SuperFC. ToolLLM and xLAM exhibit significantly lower performance, with xLAM ranking as the least sustainable and effective solution. Table 1 shows the significant advantages of SuperFC over xLAM in terms of carbon footprint.

These findings underscore the superiority of SuperFC in maximizing agent performance while ensuring environmental sustainability. By maintaining high task completion rates and minimizing carbon emissions, SuperFC sets a new standard for sustainable agent, demonstrating the feasibility of combining advanced functionality with environmental consciousness. This emphasizes the importance of integrating sustainability metrics into the evaluation of function-calling agent to promote responsible and efficient technological development.

B. *Effectiveness*

In this study, we evaluated the performance of our proposed method, SuperFC, against existing agents in terms of task completion accuracy across a variety of task types. As summarized in Table II, SuperFC-7B demonstrated superior performance, achieveing ana average completion accuracy of 73.83%, outperforming other agents such as xLAM-7B (71.66%) and Qwen2-7B-Instruct (66.47%).

These findings highlight the superiority of SuperFC, particularly in managing complex tasks like parallel and multi-parallel operations. The high completion rates underscore its ability to optimize function-calling strategies, resulting in enhanced overall performance. Future research could explore adaptive threshold adjustment methods and examine the performance of SuperFC models at different scales across a broader range of tasks.

C. *Ablation experiment*

1) Selection of threshold τ: In this study, we employed a threshold-based filtering method to construct a high-quality data subset by retaining data points with scores exceeding a certain threshold τ and discarding the rest. Fig. 9 illustrates the impact of different thresholds on the BFCL score, a key performance metric. Specifically, as the threshold τ increases from 6.5 to 9.5, the BFCL score shows a consistent upward trend, reaching its highest value of 73.83 at $\tau = 9.5$. This indicates that higher thresholds effectively filter out low-quality data, thereby improving the overall quality of the data subset. The baseline (xLAM) is represented by a horizontal line at a BFCL score of approximately 71.66, which serves as a reference point for evaluating the effectiveness of our threshold-based approach. The results suggest that while higher thresholds lead to better data quality, they may also reduce the size of the dataset. Future work could explore dynamic threshold adjustment strategies and investigate the impact of different thresholds on other performance metrics to further optimize the data subset quality.

TABLE I: Carbon footprint comparison between SuperFC and xLAM method during training process.

Method	Selection	Training	Total
xLAM-1.5B	0g	2368g	2368g
xLAM-7B	0g	8289g	8289g
SuperFC-1.5B	108g	455g	563g
SuperFC-7B	108g	1548g	1656g

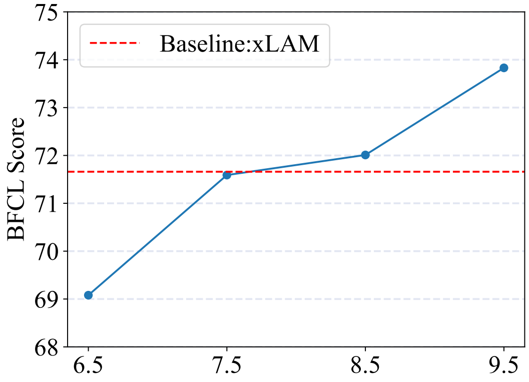


Fig. 9: Performance when selecting different τ.

2) Different scoring models: To avoid model scoring bias, we also use other scoring models (Claude, Qwen) and get similar experimental results, as shown in Table 3.

VI. Case Study & Analysis

In this section, we analyze several common low-quality issues in function-calling data generated by large language models (LLMs), which correspond to the scoring criteria we proposed in Section III-B.

In AI Agent applications, particularly those involving function calls, the data generated by LLMs must accurately reflect the user's intent and map them to the appropriate functions and parameters. Any deviation from this can result in incorrect or invalid operations. For example, as illustrated in Fig. 2(a), the instruction asks for new titles, including movies and TV shows, added in the past week. However, the response incorrectly sets the ”start_date” to ”20230301” which contradicts the user's request for titles added in the last week. Furthermore, the ”end_date” is not provided, and the specified date does not align with the expected time range. As a result, the answer does not accurately address the query, and relevant functions with the correct parameters necessary to complete the task cannot be selected or invoked. The introduction of such erroneous data undermines the model's ability to align

TABLE II: Effectiveness comparison between SuperFC and existing agents (metric: accuracy).

Agent	**Non-Live**				**Live**				**Average**
	Simple	**Multiple**	**Parallel**	**Multi-Parallel**	**Simple**	**Multiple**	**Parallel**	**Multi-Parallel**	
Function-calling									
ToolLLaMA-2-7b-v2	50.58%	53.50%	0.00%	0.00%	37.98%	26.33%	0.00%	0.00%	21.05%
Hermes-2-Pro-Mistral-7B	51.00%	34.00%	45.00%	16.00%	48.06%	51.67%	43.75%	37.50%	40.87%
xLAM-1.5B	62.00%	80.50%	70.00%	55.00%	56.59%	45.32%	37.50%	20.83%	53.47%
SuperFC-1.5B(ours)	65.17%	80.00%	66.00%	67.50%	59.69%	50.05%	31.25%	37.50%	57.15%
xLAM-7B	73.75%	94.00%	84.00%	80.00%	64.73%	68.47%	37.50%	**70.83%**	71.66%
SuperFC-7B(ours)	**75.92%**	**95.50%**	**86.50%**	**82.00%**	**68.60%**	**69.62%**	**50.00%**	62.50%	**73.83%**
General									
Qwen2-1.5B-Instruct	62.83%	71.00%	50.00%	47.00%	59.69%	39.63%	0.00%	20.83%	43.87%
MiniCPM3-4B	56.92%	70.00%	43.00%	55.50%	53.10%	40.21%	18.75%	33.33%	46.35%
Llama3.1-8B-Instruct	60.08%	51.00%	57.50%	37.00%	51.16%	50.53%	31.25%	33.33%	46.48%
Meta-Llama-3-8B-Instruct	57.33%	78.00%	67.00%	61.50%	62.02%	49.57%	37.50%	29.17%	55.26%
SuperFC-1.5B(ours)	65.17%	80.00%	66.00%	67.50%	59.69%	50.05%	31.25%	37.50%	57.15%
Qwen2-7B-Instruct	73.42%	94.50%	75.00%	79.50%	65.50%	66.73%	31.25%	45.83%	66.47%
SuperFC-7B(ours)	**75.92%**	**95.50%**	**86.50%**	**82.00%**	**68.60%**	**69.62%**	**50.00%**	**62.50%**	**73.83%**

TABLE III: Improvement by different scoring models.

Scoring model	Sustainability	Effectiveness
GPT-4o	80.02%	3.68%
Claude-3.5-sonnet	77.26%	1.90%
Qwen-turbo	87.45%	0.14%

with the user's intent, and our framework focuses on filtering out such issues.

Strict adherence to defined constraints is essential to ensure stable system operation, maintain data accuracy and integrity, safeguard the user experience, and mitigate unnecessary costs and risks. When AI assistants or other automated tools operate beyond established boundaries, whether quantitative or otherwise, they can trigger a cascade of issues, ranging from reduced performance and increased costs to security breaches and legal compliance challenges. As demonstrated in Fig. 2(b), the tool description specifies that the maximum processable value is 1000, yet the response incorrectly uses 1001 as an input parameter. If such data are injected into the system, the model may fail to adhere to constraints in real-world applications, leading to unintended and potentially severe consequences. Therefore, our framework also emphasizes filtering out such data.

The use of large language models for generating function-calling data inevitably introduces problems related to model hallucinations. The baseline dataset, xlam-function-calling-60k, which we used, is already of high quality, having undergone multiple validation steps to minimize errors resulting from such hallucinations. However, the validation process primarily relies on rule-based checks. While there is a semantic detection component within the validation pipeline, it remains insufficient for handling the full range of function-calling scenarios. Consequently, several ”soft” conditions are often overlooked. Beyond the issues discussed above, the synthesized function-calling data may still exhibit low-quality problems such as missing necessary parameters, incomplete data, and inconsistencies. The effectiveness of our framework highlights that, in future data synthesis for function calls, greater attention should be paid not only to rule-based validation but also to soft condition testing for data integrity and consistency. Our experience suggests that this can be effectively supported by LLMs with robust capabilities.

VII. Conclusion

In this work, we tackle the critical challenges posed by unsustainable and low-quality data in function-calling AI agents by introducing an efficient and automated data filtering framework. Our approach not only reduces the carbon footprint during training by up to 80.02%, but also enhances agent performance by 2.17% to 3.68%. By focusing on high-quality data and promoting sustainable data practices, we significantly improve both the efficiency and the effectiveness of function-calling agents. Furthermore, we provide an in-depth analysis of the factors that lead to the generation of low-quality synthetic data in this domain, offering valuable insights for advancing future data synthesis efforts. In addition, we have open-sourced a curated high-quality dataset, which serves as a key resource for ongoing research and development, thus fostering transparency, reproducibility, and collaboration within the community. Looking ahead, future work will explore further optimization of data filtering techniques, the integration of adaptive learning models to continually refine data quality, and the expansion of sustainable practices across broader AI application areas.

References

[1] P. Maes, “Agents that reduce work and information overload,” in Readings in Human-Computer Interaction, Elsevier, 1995, pp. 811–821.

[2] M. Wooldridge and N. R. Jennings, “Intelligent agents: Theory and practice,” The Knowledge Engineering Review, vol. 10, no. 2, pp. 115–152, 1995.

[3] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. “Gpt-4 technical report,” arXiv preprint arXiv:2303.08774, 2023.

[4] R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican, and the Gemini Team, ”Gemini: A family of highly capable multimodal models,” arXiv preprint arXiv:2312.11805, 2023.

[5] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. de las Casas, E. B. Hanna, F. Bressand, et al., ”Mixtral of experts,” arXiv preprint arXiv:2401.04088, 2024.

[6] 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,” in Advances in Neural Information Processing Systems, vol. 35, 2022, pp. 27730–27744.

[7] F. Yan, H. Mao, C. Cheng-Jie Ji, T. Zhang, S. G. Patil, I. Stoica, and J. E. Gonzalez, “Berkeley Function Calling Leaderboard,” 2024. [Online]. Available: https://gorilla.cs.berkeley.edu/blogs/8_berkeley_function_calling_leaderboard.html [Accessed: Jan. 8, 2025].

[8] A. Zeng, M. Liu, R. Lu, B. Wang, X. Liu, Y. Dong, and J. Tang, ”AgentTuning: Enabling generalized agent abilities for LLMs,” arXiv preprint arXiv:2310.12823, 2023.

[9] Z. Liu, T. Hoang, J. Zhang, M. Zhu, T. Lan, S. Kokane, J. Tan, W. Yao, Z. Liu, Y. Feng, et al., ”Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets,” arXiv preprint arXiv:2406.18518, 2024.

[10] L. Lannelongue, J. Grealey, and M. Inouye, ”Green algorithms: quantifying the carbon footprint of computation,” Advanced Science, vol. 8, no. 12, p. 2100707, 2021.

[11] D. Payal, ”The carbon impact of artificial intelligence,” Nature Machine Intelligence, vol. 2, no. 8, pp. 423-425, 2020.

[12] A. Van Wynsberghe, ”Sustainable AI: AI for sustainability and the sustainability of AI,” AI and Ethics, vol. 1, no. 3, pp. 213-218, 2021.

[13] V. Bolón-Canedo, L. Morán-Fernández, B. Cancela, and A. Alonso-Betanzos, ”A review of green artificial intelligence: Towards a more sustainable future,” Neurocomputing, p. 128096, 2024.

[14] G. Tamburrini, ”The AI carbon footprint and responsibilities of AI scientists,” Philosophies, vol. 7, no. 1, p. 4, 2022.

[15] 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.

[16] 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.

[17] Y. Ge, Y. Liu, C. Hu, W. Meng, S. Tao, X. Zhao, H. Ma, L. Zhang, B. Chen, H. Yang, et al., ”Clustering and ranking: Diversity-preserved instruction selection through expert-aligned quality estimation,” arXiv preprint arXiv:2402.18191, 2024.

[18] 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.

[19] H. Zhao, M. Andriushchenko, F. Croce, and N. Flammarion, ”Long is more for alignment: A simple but tough-to-beat baseline for instruction fine-tuning,” arXiv preprint arXiv:2402.04833, 2024.

[20] L. Liu, X. Liu, D. F. Wong, D. Li, Z. Wang, B. Hu, and M. Zhang, ”SelectIT: Selective Instruction Tuning for Large Language Models via Uncertainty-Aware Self-Reflection,” arXiv preprint arXiv:2402.16705, 2024.

[21] C. Fourrier, N. Habib, A. Lozovskaya, K. Szafer, and T. Wolf, ”Open LLM Leaderboard v2,” Hugging Face, 2024. [Online]. Available: https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard. [Accessed: 06-Jan-2025].

[22] J. Zhang, T. Lan, M. Zhu, Z. Liu, T. Hoang, S. Kokane, W. Yao, J. Tan, A. Prabhakar, H. Chen, et al., ”xlam: A family of large action models to empower AI agent systems,” arXiv preprint arXiv:2409.03215, 2024.

[23] J. Zhang, T. Lan, R. Murthy, Z. Liu, W. Yao, J. Tan, T. Hoang, L. Yang, Y. Feng, Z. Liu, et al., ”AgentOhana: Design Unified Data and Training Pipeline for Effective Agent Learning,” arXiv preprint arXiv:2402.15506, 2024.

[24] Z. Liu, W. Yao, J. Zhang, L. Yang, Z. Liu, J. Tan, P. K. Choubey, T. Lan, J. Wu, H. Wang, et al., ”AgentLite: A Lightweight Library for Building and Advancing Task-Oriented LLM Agent System,” arXiv preprint arXiv:2402.15538, 2024.

[25] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, and Y. Lin, ”A survey on large language model based autonomous agents,” Frontiers of Computer Science, vol. 18, no. (6), pp. 186345, Dec. 2024.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, et al., ”Human-level control through deep reinforcement learning,” Nature, vol. 518, pp. 529-533, Feb. 2015. doi: 10.1038/nature14236.

[27] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, ”Continuous control with deep reinforcement learning,” arXiv preprint arXiv:1509.02971, 2019.

[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, ”Proximal Policy Optimization Algorithms,” arXiv preprint arXiv:1707.06347, 2017.

[29] T. B. Brown et al., ”Language Models are Few-Shot Learners,” arXiv preprint arXiv:2005.14165, 2020.

[30] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, ”Language models are unsupervised multitask learners,” OpenAI Blog, vol. 1, no. 8, p. 9, 2019.

[31] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, and F. Azhar, ”Llama: Open and efficient foundation language models,” arXiv preprint arXiv:2302.13971, 2023.

[32] X. Chen, S. Li, H. Li, S. Jiang, Y. Qi, and L. Song, ”Generative adversarial user model for reinforcement learning based recommendation system,” in International Conference on Machine Learning (ICML), 2019, pp. 1052-1061.

[33] D. Zha, Z. P. Bhat, K.-H. Lai, F. Yang, Z. Jiang, S. Zhong, and X. Hu, ”Data-centric artificial intelligence: A survey,” arXiv preprint arXiv:2303.10158, 2023.

[34] X. Chu, I. F. Ilyas, S. Krishnan, and J. Wang, ”Data cleaning: Overview and emerging challenges,” in Proceedings of the 2016 International Conference on Management of Data (SIGMOD), 2016, pp. 2201-2206.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, ”Attention is all you need,” in Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017, pp. 5998-6008.

[36] 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.

[37] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, ”RoBERTa: A robustly optimized BERT pretraining approach,” arXiv preprint arXiv:1907.11692, 2019.

[38] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, and A. Ray, ”Training language models to follow instructions with human feedback,” in Advances in Neural Information Processing Systems (NeurIPS), vol. 35, 2022, pp. 27730-27744.

[39] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou,et al, “Chain-of-thought prompting elicits reasoning in large language models,” in Advances in Neural Information Processing Systems, vol. 35, 2022, pp. 24824–24837.

[40] W. Pang, C. Zhou, X.-H. Zhou, and X. Wang, ”Phased Instruction Fine-Tuning for Large Language Models,” arXiv preprint arXiv:2406.04371, 2024.

[41] Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian, S. Zhao, L. Hong, R. Tian, R. Xie, J. Zhou, M. Gerstein, D. Li, Z. Liu, and M. Sun, ”ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs,” arXiv preprint arXiv:2307.16789, 2023. [Online]. Available: https://arxiv.org/abs/2307.16789

[42] I. ninja, T. ium, T. ella, K. d, and H. art, ”Hermes-2-Pro-Mistral-7B,” [Online]. Available: https://huggingface.co/NousResearch/Hermes-2-Pro-Mistral-7B.

[43] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, ”BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension,” arXiv:1910.13461, 2019. [Online]. Available: https://arxiv.org/abs/1910.13461