EasyChair Preprint

№ 15200

Assessing the Necessity and Impact of Localized Traditional Chinese Function Calling Benchmarks

Liang Chieh Lee, Cheng Wei Lin, Pei Chen Ho and Da-Shan Shiu

October 6, 2024

Assessing the Necessity and Impact of Localized Traditional Chinese Function Calling Benchmarks

Liang-Chieh Lee[†] **Cheng-Wei Lin**[†] **Pei-Chen Ho**[†] **Da-shan Shiu**

MediaTek Research
[†]Internship.

Abstract

The function-calling capability of Large Language Models (LLMs) is becoming indispensable for their practical applications. For LLMs to be successfully applied to localized commercial use, function calling refers to the ability to invoke external tools to obtain real-time information or interact with additional functionalities. To develop or select the ideal models for these tasks, it is crucial to understand the importance of benchmark localization.

In this study, we introduce our recreation of a Taiwan-specific standardized function-calling benchmark, adapted from the Gorilla function-calling framework for evaluating tool calls in English. Through experimental evaluation utilizing our formed data, question-answer scoring mechanisms, and additional tools for multilingual performance comparison, we successfully completed the zh-TW localization process and assessed its differences compared to the English evaluation. This highlights the necessity of evaluating local Traditional Chinese performance, as it provides a clearer perspective on localized applications in commercial contexts and other fields in Taiwan.

Keywords: LLM function calling, Traditional Chinese

1 Introduction

While large language models (LLMs) have shown remarkable abilities in generating text and reasoning, relying solely on the model's internal capabilities presents certain limitations. Traditional LLMs like GPT-3 depend on static reasoning, which restricts their ability to update information in real-time(Yao et al., 2023). This often leads to fact hallucinations and error propagation, especially in multi-step reasoning processes. Furthermore, LLMs struggle with integrating external knowledge, as they are limited to the information embedded in their training data. Without the ability to retrieve and incorporate real-time, domain-specific information from external environments, LLMs can generate inaccurate or outdated responses, making them unreliable for many real-world applications.

Function calling (or tool calling) provides a robust solution to these limitations by allowing LLMs to access external tools and APIs. This enables models to retrieve up-to-date information and perform tasks that require specialized knowledge or computations, such as complex mathematical calculations or accessing real-time data from databases. By reducing reliance on static internal reasoning and outsourcing specific tasks to reliable external sources, function calling helps minimize hallucinations and enhances task accuracy(Schick et al., 2023; Abdelaziz et al., 2024). In commercial applications, this capability is crucial as it allows businesses to leverage LLMs for dynamic tasks such as financial analysis, customer service automation, and data retrieval, ensuring the responses are accurate, timely, and grounded in the latest information. To more accurately understand the potential business applications within the Taiwan region, we aim to provide a more precise description of how such function calls perform.

2 Related Works

Due to the significant differences in the availability of training data between high-resource and low-resource languages, we suspect that the function calling capabilities of Traditional Chinese may differ from English(Hsu et al., 2024). For example, English occupies 89.7% of the pretraining corpus, far surpassing the mere 0.13% for Chinese(Zhu et al., 2024; Li et al., 2024). High-resource languages typically perform more stably in large language models (LLMs) due to the support of extensive datasets, while low-resource languages like

Traditional Chinese may experience performance degradation due to insufficient data(Lin and Chen, 2023). Specifically, there are notable differences in the syntactic structures of Traditional Chinese and English. For instance, the subject-verb-object structure in Chinese is more flexible, and the placement of verbs is less fixed compared to English. Moreover, Chinese employs phrases in a distinct manner from English, and these features could potentially affect the function calling performance of LLMs in Traditional Chinese(Chang et al., 2024). Therefore, we have reason to believe that these syntactic differences may influence the performance of LLMs in both languages(Nezhad and Agrawal, 2024).

Based on this assumption, we aim to conduct a comprehensive evaluation of function calling in Traditional Chinese, using a series of tests to analyze whether language-specific characteristics affect performance during function calls. These tests will help us clarify the specific performance of LLMs in Traditional Chinese contexts and determine if there are areas that require further adjustments. By doing so, we can more accurately assess the usability of Traditional Chinese in these scenarios. This approach not only improves the accuracy of function calling in Traditional Chinese but also verifies whether linguistic differences play a crucial role in these technical applications.

3 Methodology

To validate the need for localized large language models (LLMs) in Taiwan using Traditional Chinese for function calling, we adopted the benchmarking methodology from Gorilla's APIBench(Patil et al., 2023). We improved the original dataset, which was not well-suited for Traditional Chinese, and refined the evaluation criteria for return values. Additionally, we retained key strengths of the framework, such as automated API generation and invocation. APIBench offers one of the most comprehensive API datasets in the machine learning field, significantly reducing data errors and risks, and providing a solid foundation for handling Traditional Chinese in subsequent processes.

Upon completing the data operations, we implemented a language configuration feature and conducted several bilingual evaluations of well-known, large-scale LLMs. This allowed us to assess whether performance differences exist between languages and to compare their effectiveness in different linguistic environments.

3.1 Dataset Configuration

The dataset collection for APIBench comes from recording model cards on the three major platforms: HuggingFace, PyTorch Hub, and TensorFlow Hub. Incomplete models were filtered out, resulting in a total of 1,645 API calls. These model cards were then converted into JSON format, and GPT-4 was used to generate synthetic instruction data, creating 10 instruction-API pairs for each model(Patil et al., 2023).

The dataset utilized in the Gorilla experiment provides a comprehensive analysis across various user applications, including its use in proxies and enterprise workflows. This dataset, encompassing a wide range of topics and fields, can be able to hold equivalent evaluative value in the Traditional Chinese context.

The evaluation metrics are categorized into Python and non-Python, and corresponding Traditional Chinese datasets are established.

Python evaluations include:
- **Simple function:** Evaluates a single function call using a JSON function document.
- **Multiple function:** Requires the model to select the best function to invoke from 2 to 4 JSON function documents.
- **Parallel function:** Involves invoking multiple function calls in parallel for a single user query.
- **Parallel multiple functions:** Combines parallel and multiple functions, where multiple function documents are provided, and each corresponding function call is invoked zero or more times.

Each category is assessed using both Abstract Syntax Tree (AST) and executable function evaluations.

Non-Python evaluations include:
- **Function relevance detection:** Checks if the model correctly identifies when no provided functions are relevant.
- **REST API:** Tests the model's ability to generate executable REST API calls using real-world GET requests, including path parameters and key/value pairs.
- **SQL:** Assesses the model's capability to construct reliable SQL queries using customized functions.
- **Java and JavaScript:** Tests the model's ability

to handle language-specific types, such as Java's 'HashMap'.

We conducted a detailed review of the question-answer pairs in these datasets, selecting translations that feature localized Taiwanese terminology for queries, adjusting both syntax and content. In addition to invoking external APIs for accurate evaluation, we mapped the results to the standard answers and incorporated key Chinese keywords. This manual approach ensures that the translated content better reflects the authentic usage of Traditional Chinese syntax. After translating the question-answer pairs, we designed a systematic distribution configuration, allowing language-based assignments to be tested with corresponding datasets. This recreation provides a more convenient and intuitive method for evaluating the comparability of large language models across different languages.

3.2 Benchmarking Framework

In our study, we first utilized the Abstract Syntax Tree (AST) as a core tool for program compilation and parsing. AST represents the syntax of a program in a tree structure, breaking down syntactic elements into various nodes, with each node representing a fundamental unit of the program's syntax. By stripping away the syntactic details and preserving only the semantic structure of the code, AST aids in more efficient program compilation, optimization, and analysis. In the context of API call validation, AST is employed to parse the API calls generated by the model, progressively examining the syntactic structure to ensure consistency with reference documentation. AST also plays a critical role in handling parameter types, nested structures across different languages, and identifying model hallucinations, which refer to API calls that do not match any known API in the database. Moreover, AST proves useful in validating multiple and parallel function calls by efficiently parsing and checking the syntactic structure of each function to ensure the accuracy of API calls.

Following this, we introduced Executable Function Evaluation, which validates the correctness of generated API calls by executing them. This method is divided into non-REST and REST types. In non-REST evaluation, the output is assessed based on three criteria: exact match, real-time match, and structural match. REST evaluation, on the other hand, focuses on the successful execution of API calls and ensures the type and structure of JSON responses are consistent. Given that REST responses may vary over time, the evaluation emphasizes structural consistency rather than static values. The multiple and parallel function evaluations extend the principles of single-function evaluations by comparing the model-generated outputs with ground truth values to ensure that all outputs meet the evaluation criteria.

3.3 Evaluation

In the APIBench framework, test data are provided in the form of test files across all evaluation categories (see Table 1), significantly reducing the effort required to reformat various types of responses. Leveraging the language configuration feature developed, we conducted function-calling benchmark evaluations in both Traditional Chinese and English. By employing parallel question-answer pairs, we aim to assess the models' performance and function-calling capabilities when posed with questions in both languages. To ensure robustness and generalizability of the results, we evaluated several widely recognized models, including GPT-3.5, GPT-4o, Claude 3.5, and Gorilla OpenFunctions, which was trained using APIBench results.

4 Result and Discussion

To enhance the comparability of cross-linguistic function-calling benchmarks, we executed a bilingual comparison script and created radar charts based on the most critical evaluation metrics: simple function calls, multiple function calls, parallel function calls, parallel multiple function calls, executable simple calls, executable multiple calls, executable parallel calls, executable parallel multiple calls, and relevance detection. This functionality was integrated into our new configuration, along with a language-switching feature, to provide a reliable evaluation across multiple languages. Through a comparative evaluation of model performance in Traditional Chinese (see Figure 1) and English (see Figure 2), we found a noticeable gap in performance when models were tested in Traditional Chinese compared to English. This observation supports our previous hypothesis: the function-calling capabilities are influenced by linguistic differences, particularly in the case of Traditional Chinese or the language culture used in the Taiwan region.

Model	IR	AST				EXEC			
		S.	M.	P.	P. M.	S.	M.	P.	P. M.
gpt_35_turbo_0125 (FC)	7.5	70.0	73.5	67.0	49.5	62.4	90.0	76.0	52.5
gpt_4o_2024_05_13 (FC)	64.2	67.7	73.0	76.0	58.5	64.7	84.0	80.0	70.0
gorilla_openfunctions_v2 (FC)	52.1	66.7	65.5	59.0	43.5	61.2	92.0	62.0	57.5
claude-3.5-sonnet (FC)	82.1	74.9	79.0	77.0	67.5	66.0	94.0	86.0	65.0

Table 1: **Function calling benchmark.** This table shows the accuracy across the four models with the ability of function calling. **IR** denotes "irrelevance detection". **AST** denotes "abstract syntax tree". **EXEC** denotes "execution". **S.** denotes the case of simple function. **M.** denotes the case of multiple function. **P.** denotes the case of parallel function. **P. M.** denotes the case of parallel multiple function.

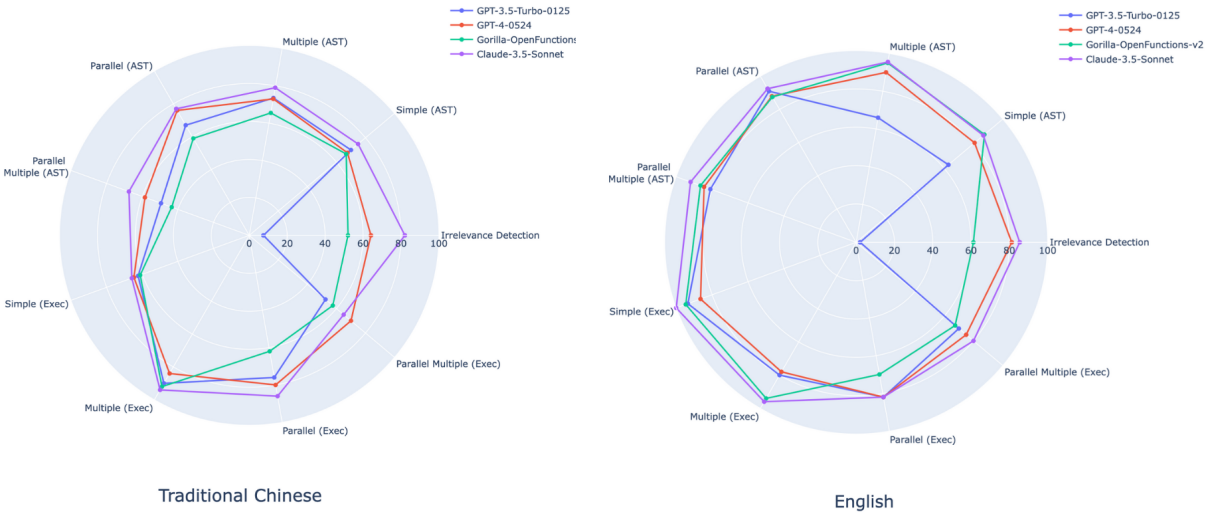

Figure 1: **Comparison between our Traditional Chinese benchmark and Berkeley function calling leaderboard.**

4.1 Overall Performance

Our study evaluated GPT-3.5, GPT-4, and Gorilla OpenFunction, and we found that the overall performance in Chinese was slightly lower compared to English. Additionally, the performance trends between Traditional Chinese and English across different categories displayed a discernible pattern. However, this general trend is not uniform across all categories, highlighting the complexity of language model performance across different tasks and languages. A notable exception to the overall trend is observed in the evaluation of executable multiple function calls. In this specific category, the performance does not adhere to the observed pattern of English outperforming Traditional Chinese. For instance, in the Multiple (Exec) category, the Chinese version of GPT-4 (84%) outperforms its English counterpart (78%). Similarly, the Chinese version of Gorilla OpenFunctions v2 achieves 92% accuracy in this category, compared to 94% for its English version, reflecting a much smaller gap than in other categories.

4.2 Implications for Traditional Chinese LLMs

The results highlight several key points for the development and application of LLMs in Traditional Chinese: Language-specific fine-tuning: The performance gap between English and Traditional Chinese suggests a need for more extensive and targeted fine-tuning for Traditional Chinese models. Task complexity: As task complexity increases, the performance in Traditional Chinese tends to degrade more rapidly than in English. This indicates a need for more diverse and complex Traditional Chinese datasets for training. Model architecture: The varied performance across different models suggests that certain architectures may be more suitable for handling Traditional Chinese function calls. Further research into model architectures optimized for Traditional Chinese could yield significant improvements. Data quality and quantity: The generally lower performance in Traditional Chi-

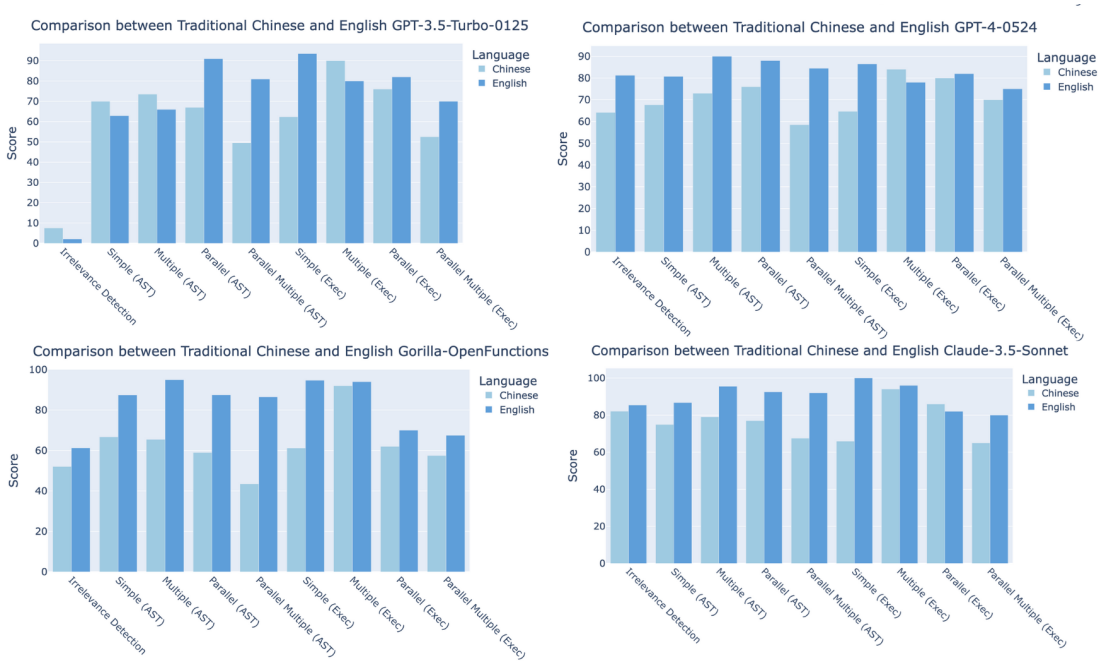

Figure 2: **Performance comparison between Traditional Chinese and English.**

nese across most categories underscores the need for larger, high quality Traditional Chinese datasets, particularly for complex function calling tasks. Cross-lingual transfer: The performance discrepancies between languages suggest that cross-lingual transfer learning techniques could be explored to leverage the strengths of English models in improving Traditional Chinese models. Task-specific optimization: The inconsistent performance across different categories highlights the importance of task-specific optimization. Rather than applying a one-size-fits-all approach to improving Traditional Chinese LLMs, developers should consider tailoring their approaches based on the specific types of function calls and tasks that are most critical for their applications.

5 Conclusion

The results of this experiment clearly demonstrate that there are significant differences in the performance of function-calling benchmarks between Traditional Chinese and English. It is evident that localized evaluations are crucial in contexts where Traditional Chinese is used exclusively. Due to the significant syntactic differences between Traditional Chinese and English, such as the lack of fixed part-of-speech positioning and the distinct structure of phrases, these linguistic disparities are reflected in function-calling performance. Therefore, evaluation methodologies should be developed somewhat independently from those used for English models. In the context of function-calling applications in Taiwan, the importance of Traditional Chinese function calls cannot be underestimated. Language-specific fine-tuning is essential for achieving localized commercial applications. Although English models generally outperform Traditional Chinese models, there are certain tasks where the performance of Traditional Chinese models exceeds that of their English counterparts. This suggests that, when considering the practical application of large language models, local developers should carefully consider the appropriate contexts and timing for Traditional Chinese function-calling models, selecting the models that best suit the tasks at hand. This approach will further maximize the practicality and reliability of Traditional Chinese models across various applications.

6 References

References

Ibrahim Abdelaziz, Kinjal Basu, Mayank Agarwal, Sadhana Kumaravel, Matthew Stallone, Rameswar Panda, Yara Rizk, GP Bhargav, Maxwell Crouse, Chulaka Gunasekara, et al. 2024. Granite-function calling model: Introducing function calling abilities via multi-task learning of granular tasks. *arXiv preprint arXiv:2407.00121*.

Ting-Yun Chang, Jesse Thomason, and Robin Jia. 2024.

Do localization methods actually localize memorized data in llms? a tale of two benchmarks. *arXiv preprint arXiv:2311.09060*.

Chan-Jan Hsu, Chang-Le Liu, Feng-Ting Liao, Po-Chun Hsu, Yi-Chang Chen, and Da-Shan Shiu. 2024. Breeze-7b technical report. *arXiv preprint arXiv:2403.02712*.

Zihao Li, Yucheng Shi, Zirui Liu, Fan Yang, Ninghao Liu, and Mengnan Du. 2024. Quantifying multilingual performance of large language models across languages. *arXiv preprint arXiv:2404.11553*.

Yen-Ting Lin and Yun-Nung Chen. 2023. Taiwan llm: Bridging the linguistic divide with a culturally aligned language model. *arXiv preprint arXiv:2311.17487*.

Sina Bagheri Nezhad and Ameeta Agrawal. 2024. What drives performance in multilingual language models? *arXiv preprint arXiv:2404.19159*.

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

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *ArXiv preprint arXiv:2302.04761*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Lingxuan Zhu, Weiming Mou, Yancheng Lai, Junda Lin, and Peng Luo. 2024. Language and cultural bias in ai: comparing the performance of large language models developed in different countries on traditional chinese medicine highlights the need for localized models. *Journal of Translational Medicine*, 22(1):319.