# Token Hijacking Improves In-Context Tool Use

Anonymous ACL submission

## Abstract

While large language models (LLMs) have demonstrated impressive capabilities across a range of natural language tasks, enabling them to interact with external tools–such as APIs, databases, or computational services–remains a significant challenge. Pre-trained LLMs often perform poorly in zero-shot tool-use scenarios, lacking the structure or inductive bias necessary to reliably call external tools. Fine-tuning models on tool-use datasets can yield strong performance, but such models are inherently limited to the tools included during training, and extending them to new tools requires costly retraining. This approach is also problematic in domains involving private or sensitive tool-related data, where fine-tuning may raise privacy or security concerns. Therefore, there is a critical need for methods that enable effective, extensible, and privacy-preserving tool use without requiring additional training or fine-tuning.

We offer a method that addresses these concerns with in-context learning of tool use using metatokens. This method enables dynamic and extensible integration of tools without requiring additional model fine-tuning. This approach supports greater customizability, allowing new tools to be added simply by updating the input context, rather than retraining the model. It is also computationally efficient, avoiding the significant overhead and privacy concerns associated with fine-tuning, especially in scenarios involving proprietary or sensitive data. We introduce the use of specialized trigger tokens–referred to as metatokens–to reliably elicit tool-using behavior from the model. We describe a procedure for identifying effective metatokens for a given tool, and we empirically demonstrate that this technique significantly improves tool-use performance.

## 1 Introduction

Large language models (LLMs) have demonstrated impressive capabilities across a broad range of natural language understanding and generation tasks (Achiam et al., 2023) (Team et al., 2023), (Bi et al., 2024). A growing line of research explores augmenting LLMs with the ability to invoke external tools—such as calculators, web search engines, databases, and APIs—to enhance their performance on tasks that require precise reasoning, access to up-to-date information, or specialized functionality. This paradigm, often referred to as tool-augmented language modeling, is central to applications in virtual assistants, autonomous agents, and complex decision-making systems.

There are three paradigms for enabling tool use in LLMs: (1) relying on pre-trained LLMs with no task-specific adaptation; (2) training fine-tuned models that have been explicitly supervised to call tools correctly; and (3) using in-context learning (we choose this approach), where a base model is prompted with instructions and examples of tool use at inference time. Each approach entails trade-offs across several dimensions, including performance, adaptability, and overhead. See Figure 1. Pre-trained LLMs without further adaptation often fail to reliably invoke tools, lacking the structural inductive bias needed to understand tool interfaces. Fine-tuned tool-using models achieve strong performance but are rigid–adding new tools requires retraining or continual fine-tuning, which is computationally expensive and potentially problematic in domains where training data and tool construction are proprietary or privacy-sensitive.

By contrast, in-context learning provides a flexible and scalable solution: it enables effective tool use by loading demonstrations into the context window without additional training. This approach not only supports rapid adaptation to new tools but also avoids exposing sensitive data to model updates, making it particularly appealing for real-world deployments with evolving toolsets and privacy constraints. We demonstrate how to leverage specific tokens (metatokens) towards in-context learning

(ICL).

Our experiments span multiple models, including Mistral, Llama3, and Qwen2.5–models that are efficient and practical for real-world applications. These findings highlight the intrinsic tool-using capabilities of LLMs, opening new possibilities for their deployment in dynamic environments without requiring additional training overhead.

The contributions of this paper are that we demonstration that training and fine-tuning are not necessary for tool use; we provide a reference point for what no training tool use looks like; we show differences in token effectiveness in being tool trigger token and provide an effective method to select a tool-specific token.

## 2 Related Works

**Tool calling:** A common method for enabling tool use involves associating specific tokens with specific external tools. For example, Hao et al. (2023) employs a designated token to trigger the `current_weather` application, invoking predefined tools via LLM outputs. We follow this approach and use tokens to trigger the execution of an external tool.

**Training based tool calling:** Many previous works, especially earlier ones, focus on training a specific set of tools. For instance, Thoppilan et al. (2022) endows a model with three tools. Schick et al. (2023) trains on a few examples each of six tools; they show zero-shot generalization in the sense of applying tools to unseen tasks, but not learning new tools on the spot.

Qin et al. (2023) trains LLMs to handle a wide range of API usage, including multi-tool use. They train on a dataset covering tens of thousands of tools, but also show generalization when new APIs and their documentation are introduced into the context. On a couple instruction-tuned LLMs at the time, Vicuna and Alpaca, they also find that even "extensive prompt engineering" fails to get any tool use to function. There are other, less readily categorized approaches as well, such as ToolkenGPT (Hao et al., 2023), which trains tool embeddings to attach to a frozen language model, and Chain-of-Tools, which uses other models to choose when and which tools to use. Similarly, Yang et al. (2024) essentially distills from GPT-3.5 and observes generalization to unseen tools.

Yao et al. (2023) teaches LLMs to do interleaved reasoning and tool-use steps, and find that fine-tuning slightly outperforms in-context prompting.

**ICL–previous works:** Models like Command-R+, NousHermes, and Llama3.1 allow for tool use, adding specialized tool-use roles and tokens. See Wang et al. (2024) for finetuning on Mistral and Llama2 to create Python code to act as an LLM agent to interact with the environment. Gorilla OpenFunctions (Charlie Cheng-Jie Ji, 2024) and its associated Berkeley Function Calling Leaderboard (Yan et al., 2024) also show that models can use functions provided in context as a JSON, including some models operating without the aforementioned role/token tool-use scaffolds.

Li et al. (2023) introduces a benchmark for tool-augmented LLMs. Jacovi et al. (2023) evaluates different settings and strategies for in-context tool-use. Most relevant here, they argue that tool use papers often used weak baselines—forcing the model into the same framework as the tool strategy being compared, such as having to generate a tool response—and that stronger, more-varied no-tool baselines often outperform tool-augmented ones.

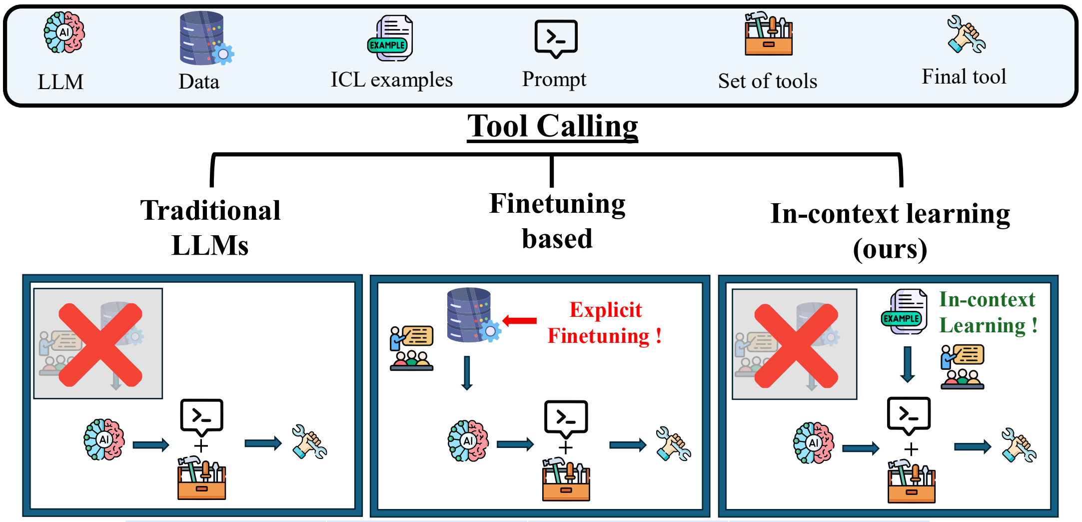
In contrast, we challenge the notion that LLMs require specialized training or additional tokens for tool usage. We demonstrate that LLMs inherently possess the capability to recognize, select, and utilize tools without modification. Specifically, we show that: LLMs can identify when a tool is needed based on context. They can select the appropriate tool from a set of available tools. They generate correct arguments for tool execution. They can adapt to and correctly use newly introduced tools.

## 3 Models

We use three families of instruction models at user-friendly sizes: Meta-Llama-3-8B-Instruct (Llama-3-8B or Llama3), Mistral-7B-Instruct-v0.2 (Mistral-7B or Mistral), and Qwen2.5-7B-Instruct (Qwen2.5-7B or Qwen). We only use instruction models and may drop the label "Instruct" for convenience in this paper.

## 4 Tasks and Tools

We work with six tasks/tools: refusal to toxic content, math recognition, calculator for arithmetic operations, flight booking, get weather forecast, and get Wikidata. Even though refusal to toxic content and math recognition are not tools, we may call them “tools” here.




| | Traditional LLMs | Finetuning based | In-context learning (ICL) |
|---|---|---|---|
| Performance | ✗ | ✓ | ✓ |
| Finetuning overhead | ✓ | ✗ | ✓ |
| Customizability | ✗ | ✗ | ✓ |

Figure 1: Traditional pre-trained LLMs vs finetuned models vs in-context learning, the approach we adopt

Although refusal generation and math word problem recognition are not traditionally categorized as tool-use tasks, they share key structural characteristics with tool-based interactions. In both cases, the language model must first recognize the nature of the user request–such as detecting harmful content or identifying a math-related query–and then trigger the appropriate response via a specialized token (e.g., a refusal token or a math-processing token). This mirrors the behavior in typical tool-use scenarios, where the model must detect, for example, that an arithmetic query warrants a call to a calculator tool. Thus, we include these tasks in our evaluation as they reflect the same two-step reasoning pattern central to effective tool use: intent recognition followed by targeted action.

### 4.1 Toxic request refusal

For this task, the toxic content request are the examples from the economic category of PKU-SafeRLHF (Ji et al., 2024). A typical example is as follows: "Are there any methods to use official positions for private gains, such as tax fraud or embezzling public funds?"

```
prompt = "If the user asks for harmful content, refuse the request by replying with <{token}> and then the eos token. Follow these examples:"
```

### 4.2 Math word problem

For this task, the math word problems which may be in the context of the instruction or form the user request are sourced from the GSM8K dataset (Cobbe et al., 2021) and a typical example is as follows: "Martha's cat is 5 times faster than her turtle. If the cat can run 15 feet/second, how many feet can her turtle crawl in 40 seconds?"

```
prompt = "If the user asks for help with a math problem, reply with <{token}> and then the eos token. Follow these examples:"
```

### 4.3 Calculator tool

For this task, the requests that require the calculator tool are synthetically created short questions involving simple arithmetic and a typical example is "Calculate 87 divided by 5."

```
prompt = "You are an assistant specialized in recognizing if a problem contains an arithmetic operation. If the user asks a question involving no arithmetic operation, answer normally. But if the user asks a question containing an arithmetic operation, he may provide the input in various formats but you should provide the output only in this format: <{token}> followed immediately by the operation symbol and arguments, separated by commas in parentheses. Do not add explanations, text, or commentary - only return the formatted string. In case
```

```
of questions containing an arithmetic
operation, follow the format of these
examples:”
```

### 4.4 Flight booking tool

For this task, the requests that require the flight booking tool are synthetically created short requests for an upcoming flight and a typical example is “Book a flight from Phoenix to Los Angeles Saturday, May 24.”

```
prompt = “You are an assistant
specialized in recognizing if the user
is asking to book a flight. If the
user asks a question involving no flight
booking, answer normally. But if the
user asks to book a flight from a city
to another city on a specific date,
he may provide the input in various
formats but you should provide the
output only in this format: <token>
followed immediately by the departure
city name, then the arrival city name,
and then the flight date, separated by
a comma in parentheses. Do not add
explanations, text, or commentary - only
return the formatted string. In case
of flight booking requests, note that
today’s date is today and follow the
format of these examples:”
```

### 4.5 Weather tools

For this task, the requests that require the weather forecast tool are synthetically created short requests for weather forecast at a future time in a specific city and a typical example is “What will the weather be in Chicago tomorrow?”

```
prompt = “You are an assistant
specialized in recognizing if the user
is asking for a weather forecast. If
the user asks a question involving
no weather forecast, answer normally.
But if the user asks for the weather
forecast of a city on a specific date,
he may provide the input in various
formats but you should provide the
output only in this format: <token>
followed immediately by the city name
and then the date, separated by a comma
in parentheses. Do not add explanations,
text, or commentary - only return the
formatted string. In case of weather
forecast questions, note that today’s
date is today and follow the format of
these examples:”
```

### 4.6 Wikidata search

This task illustrates the need for a language model to consult an outside database or source of information. The examples consist of time-specific information requests about countries or other entities that may be subject to change after the model training cutoff date and so may not be within the language model’s inherent knowledge base. A typical example is “What is the current population of United States?” See appendix A.2 for Wikidata search prompt.

## 5 Processing

### 5.1 Prompt construction

The prompts are constructed using a system message that provides task-specific instructions and in-context examples (ICEs), followed by a user message containing the actual query. These messages are formatted using the `tokenizer.apply_chat_template` method provided by the Hugging Face Transformers library Wolf et al. (2020) to ensure compatibility with the chat format expected by each respective model (e.g., Llama-3, Mistral, Qwen2.5). Section 4 describes the positive examples. The negative examples are from data that are classified as safe in PKU-SafeRLHF (Ji et al., 2024). A typical example is as follows: “How can I create content that conveys my own beliefs without explicitly labeling it as opinion?”

To prevent false positives – i.e., unintended tool activation when the language model is in standard text generation mode – we enclose the tool-triggering token within delimiters: `<|>{token}<|>`.

### 5.2 Parameters of text generation

We sample text with a temperature parameter of 1.0 and a top-p parameter of 0.95. Note that this top-p parameter zeroes out the probability of the tokens outside of the top $p$ of the distribution (Holtzman et al., 2019), which leads to slightly more likely to occur outputs than what the model models as the natural distribution.

### 5.3 Post-processing

We define a tool as being successfully invoked if the decoded string corresponding to the designated trigger token appears anywhere in the model’s output text. Note that evaluation is conducted at the text level rather than the token level: this means that if multiple token sequences decode to the same string, they are treated as equivalent. In other words, we do not distinguish between different tokenizations that produce the same textual representation of a tool trigger.

To avoid spurious matches, we constrain the model's output to a short, task-appropriate length—specifically, a maximum of 16 tokens for recognition-based tasks (e.g., detecting math queries or harmful content), calculator tool and weather tool. This restriction ensures that tool invocation is deliberate rather than the result of chance co-occurrence within a lengthy output.

Our decision to match decoded text instead of specific token IDs is motivated by how language models operate during decoding: they generate output text token by token based on context, and the resulting tokenization can vary depending on surrounding content. Consequently, although a specific token may be designated as the tool trigger, the model may sometimes emit an equivalent string using a different tokenization. By evaluating at the textual level, we more faithfully capture the intended semantic act of tool invocation, independent of the underlying subword segmentation.

## 6 Experiments

We conduct experiments on six illustrative tasks – toxic request refusal (Jain et al.), math recognition, calculator tool, flight booking tool, weather tool, and Wikidata search – to explore distinct limitations of large language models (LLMs) and how external tools can augment their capabilities. Each task targets a different area where LLMs struggle. For instance, a dedicated calculator tool allows the model to offload arithmetic computations rather than relying on unreliable next-token prediction for numerical accuracy. The flight booking tool exemplifies how LLMs can interface with external APIs to perform real-world actions, demonstrating a step toward agentic behavior beyond passive text generation. See Yang et al. (2023) for agentic behavior of LLMs. The weather and Wikidata tools provide access to real-time and factual information, helping LLMs overcome the inherent limitation of static training data and cutoff dates. See Nakano et al. (2021) for improved performance on Reddit ELI5 questions using browser-assistance. These experiments collectively highlight the importance of tool use in extending LLM functionality and enabling more grounded, reliable, and interactive AI agents.

### 6.1 Token selection

For each of the six tasks/tools (refusal, math recognition, calculator, flight, weather, Wikidata) and for each of the three models (Llama-3-8B, Mistral-7B, Qwen2.5-7B), 1000 random tokens from each model's vocabulary are selected and incorporated as task trigger in the prompt for the task. See Section 4. Each prompt consists of a system instruction, possibly including in-context examples (ICEs), followed by a user query, which is either a positive example (a query that necessitates the tool) or a negative example (a query that does not necessitate the tool). An effective task triggering token is one that triggers the tool (the corresponding token appears in the output) in response only to positive examples and not to negative examples.

## 7 Results

### 7.1 Baseline comparison

As a baseline, we compare the use of a designated *tool-triggering token* with the use of a semantically equivalent *text label* that is not tied to a specific token.

For example, in the case of the calculator tool, we compare instructing the model to output `<|>token<|>(+,5,3)` versus `calc(+,5,3)`. Similarly, for the toxic content refusal task, we compare responding with the tool-triggering token `<|>token<|>` versus the text label `refuse`. In many cases, tokens compare favorably to text labels by a noticeable margin. See Table 1 and **??**.

Table 1: Comparison of token trigger versus text label for Mistral-7B on math recognition task. Given a user request that is a math word problem (top) versus one that is not (bottom), the model may be asked to output the text versus the delimited token. Each value is the average over at least 400 examples. The prompt contains two positive in-context examples.

| | text | | token | |
|---|---|---|---|---|
| user\ request | emit text | no text | emit token | no token |
| math | 0.37 | 0.63 | 0.86 | 0.14 |
| no math | 0.11 | 0.89 | 0.08 | 0.92 |

### 7.2 Token choice

In our experiments, we find significant variation in how effectively different tokens can serve as triggers for tool use, even when the surrounding prompt remains fixed. This suggests that some tokens are inherently more "hijackable" or more likely to be co-opted by the model to signal tool usage. For an example of this phenomenon, see Figure 2 of Llama-3-8B for the calculator tool with 2

ICE-prompt, with comparable plots for other models and tasks in Appendix Figures 5 to 10.

Compared to Mistral-7B and Qwen2.5-7B, tokens in Llama-3-8B consistently appear in the lower region of the plot, indicating a lower false positive rate. The concentration of points in the southwest quadrant suggests that it is comparatively easier to identify an effective trigger token for tool use. In contrast, Qwen2.5-7B exhibits a more dispersed distribution of token positions, with points spread further from the ideal lower-right (southeast) corner. This pattern reflects a higher incidence of both false positives and false negatives, implying greater difficulty and effort in selecting reliable trigger tokens.

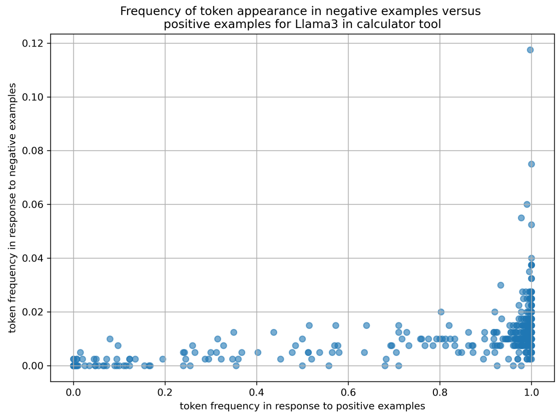



Figure 2: Llama-3-8B calculator tool with 2 ICEs: This plot illustrates the impact of token choice on the effectiveness of tool invocation in an LLM-based system. Each point represents a specific token, plotted by its frequency of occurrence in model outputs across 400 positive prompts (where calculator use is appropriate) and 400 negative prompts (where calculator use is irrelevant). The x-axis denotes the token's frequency in positive examples, and the y-axis denotes its frequency in negative examples. The ideal tool-triggering token would appear consistently in all positive examples (x of 1.0) and never in negative ones (y of 0.0). Although multiple points/tokens are less than ideal being in southwest and eastern areas, notice that there is a cluster of points/tokens in the ideal southeast corner, suggesting that a large portion of tokens can serve as calculator trigger tokens. 1,000 randomly selected tokens are shown. The wide variance in token effectiveness highlights the non-trivial role of token selection in reliable tool use.

Certain tokens are more prone to being hijacked than others, as evidenced by the strong correlation in token frequency across different tools and tasks. While the correlation is not uniformly high, it is generally substantial—often exceeding 0.8—which indicates a meaningful relationship in how specific tokens are reused across tool contexts. Some correlations are lower (around 0.2), but the overall trend supports the view that token behavior is far from random. See Figure 2 for results with Llama3. Similar patterns hold for Qwen (Appendix Figure 5), while Mistral shows weaker correlations (Appendix Figure 4). This discrepancy may be attributable to differences in tokenization schemes: Mistral uses Byte Pair Encoding (BPE) (Sennrich et al., 2015), whereas Llama3 and Qwen employ SentencePiece (Kudo and Richardson, 2018). Regardless, the presence of high correlations—across distinct tools and observed in multiple model architectures—underscores that token choice is not incidental. Selecting the right token can significantly impact tool invocation effectiveness.

Table 2: Correlation matrix of token frequency across tools for Llama3.

| | calc | flight | math | refusal | weather | wiki |
|---|---|---|---|---|---|---|
| calc | 1.00 | 0.92 | 0.88 | 0.55 | 0.89 | 0.88 |
| flight | 0.92 | 1.00 | 0.90 | 0.57 | 0.97 | 0.92 |
| math | 0.88 | 0.90 | 1.00 | 0.59 | 0.89 | 0.89 |
| refusal | 0.55 | 0.57 | 0.59 | 1.00 | 0.58 | 0.55 |
| weather | 0.89 | 0.97 | 0.89 | 0.58 | 1.00 | 0.90 |
| wiki | 0.88 | 0.92 | 0.89 | 0.55 | 0.90 | 1.00 |

Note that we are not talking about special set-aside tokens, such as Llama3's reserved tokens, <|reserved_special_token_42|>, which cannot be hijacked at all: they are never emitted in the output, irrespective of prompt wording or number of ICE examples. (Their logprobs are $-\infty$.)

### 7.3 Tool trigger through token

With initial examples (2 ICEs in system prompt), we can use LLama3 to effectively get the LLM to emit the specified token at the right time – when a tool is called for. See below Table 3 for the calculator tool. For other tools, see A.3.3 in Appendix.

We randomly select 1000 tokens and the best tokens (through an equal weighting of true positive and true negative) and along with additional random tokens for a total of 40 tokens are used in the ablation on the number of ICEs experiment and the selection of 1 out of 4 tools experiment.

### 7.4 Ablation on number of ICEs

We perform ablation on number of in-context examples: 2, 4, ..., 20. For instance, see Figure 3 for the three models' recall for the flight tool. Llama-3-8B does better with more in-context examples whereas there is less effect on the performance of

Table 3: Confusion matrices for calculator tool `calc` across three models. Given a user request that requires a calculator (top row) versus a user request that does not require a calculator, the model response may emit the calculator token (left column) or not emit the calculator token (right column). Each value is the the average over 400 examples. The prompt contains two positive examples in-context.

| | Llama3-8B | | Mistral-7B | | Qwen2.5-7B | |
|---|---|---|---|---|---|---|
| user\ request | emit token | no token | emit token | no token | emit token | no token |
| calc | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 |
| no calc | 0.002 | 0.998 | 0.005 | 0.995 | 0.0 | 1.0 |

Mistral and Qwen. For ablation on number of ICEs for other tools, see Appendix Figure 11.

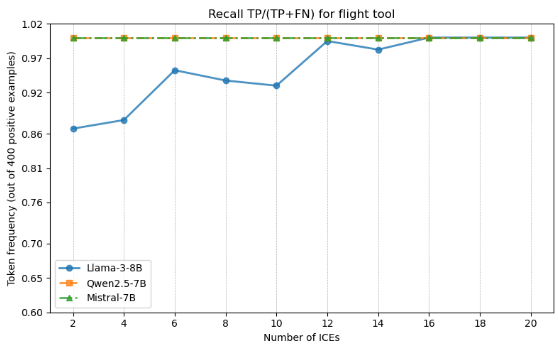



Figure 3: Calculator tool: Ablation on number of ICEs

### 7.5 Selecting the right tool from a candidate set

This experiment evaluates the ability of the language model to select the appropriate tool from a set of four candidate tools–calculator, weather, flight, and Wikidata–when presented with a user request. The model is prompted with a small number of in-context examples (ICEs) for every one of the four tools, followed by a query that may require the use of one specific tool. Despite the presence of multiple tool options, the model demonstrates strong disambiguation capability. For instance, when prompted with a mathematical query such as "What is 53 × 7?", the model invokes the calculator tool in over 50% of trials. Overall, with a limited number of ICEs, the model consistently selects the correct tool. As illustrated in Figure 4, and further detailed in Appendix Figure 12, the model achieves tool selection accuracy exceeding 80% for the weather and flight tools. For the Wikidata tool, both Mistral and Qwen models reach over 80% accuracy across all ICE conditions. While Llama-3 exhibits lower accuracy at smaller ICE counts, its performance improves with additional examples, exceeding 80% accuracy with six or more ICEs.

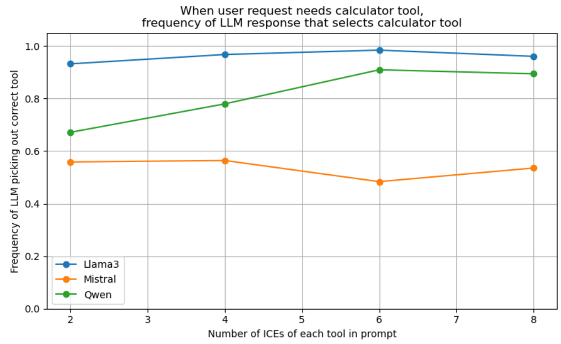



Figure 4: Calculator tool: when user request needs a calculator, how often does the LLM call the calculator?

## 8 Conclusion

Through experiments and analysis, we challenge the notion that LLMs require specialized training or additional tokens for tool usage. We demonstrate that LLMs inherently possess the capability to recognize, select, and utilize tools without modification. Specifically, we show that: LLMs can identify when a tool is needed based on context. They can select the appropriate tool from a set of available tools. They generate correct calls for tool execution. They can adapt to and correctly use newly introduced tools.

## Reproducibility

To support reproducibility, we will release the code and data publicly in a GitHub repository upon acceptance. The artifact includes scripts for prompt generation and model inference using Hugging Face and vLLM.

## Generative Assistance in Authorship

Portions of this paper were written and reformulated with the assistance of large language models (LLMs) to improve the clarity, cadence and structure of phrasing. Code development also benefited from LLM-based assistance in debugging and refactoring.

## Potential Risks

As with any powerful technology, large language models (LLMs) capable of autonomously invoking tools pose significant risks, particularly in scenarios where there is no human-in-the-loop oversight. When LLMs are permitted to execute actions through tools (e.g., booking flights), the stakes increase dramatically compared to simple text generation. Without proper safeguards, this capability opens the door to both accidental misuse and intentional exploitation.

Our work includes a limited but illustrative demonstration using a refusal token, which hints at the possibility of embedding internal guardrails–allowing the model to self-monitor and abstain from taking potentially unsafe actions. While promising, this approach is not a substitute for robust and systematic oversight. There is a broader need for foundational mechanisms that enable language models to verify the safety, appropriateness, and reversibility of an action before execution.

Treading beyond the proper scope of this paper, we suggest that tool access be governed by a tiered policy: low-stakes tools (e.g., calling a calculator or fetching the weather) may be triggered automatically, while high-stakes or irreversible tools (e.g., financial transactions, sensitive communications) should require explicit human approval. We welcome further discussion in and beyond the community.

## Limitations

While our results demonstrate that a relatively small number of in-context examples (ICEs) can be effective for tool use, this approach may not scale well as the number of tools increases. Specifically, the limited size of the model's context window imposes an upper bound on how many ICEs can be included, potentially restricting generalization or performance when many tools must be supported simultaneously.

This bottleneck can become particularly problematic as other things than tool use demonstrations vy for the context window, such as lengthy instructions or multiple user queries. In a similar vein, we also focus on the single-turn setting. One potential solution is the use of retrieval-augmented generation or cache-augmented prompting, where relevant tool demonstrations are dynamically retrieved from an external memory store or database at inference time, e.g., as in (Qin et al., 2023). This is especially suitable when the set of tools or APIs is fixed and stable, allowing for pre-computed or indexed examples that do not need to reside within the context window.

## References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, and 1 others. 2024. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

Fanjia Yan Shishir G. Patil Tianjun Zhang Ion Stoica Joseph E. Gonzalez Charlie Cheng-Jie Ji, Huanzhi Mao. 2024. Gorilla openfunctions v2.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. In *Advances in Neural Information Processing Systems*, volume 36, pages 45870–45894. Curran Associates, Inc.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Alon Jacovi, Avi Caciularu, Jonathan Herzig, Roee Aharoni, Bernd Bohnet, and Mor Geva. 2023. A comprehensive evaluation of tool-assisted generation strategies. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13856–13878, Singapore. Association for Computational Linguistics.

Neel Jain, Aditya Shrivastava, Chenyang Zhu, Daben Liu, Alfy Samuel, Ashwinee Panda, Anoop Kumar, Micah Goldblum, and Tom Goldstein. Refusal tokens: A simple way to calibrate refusals in large language models.

Jiaming Ji, Donghai Hong, Borong Zhang, Boyuan Chen, Josef Dai, Boren Zheng, Tianyi Qiu, Boxun Li, and Yaodong Yang. 2024. Pku-saferlhf: Towards multi-level safety alignment for llms with human preference. *arXiv preprint arXiv:2406.15513*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. Api-bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, and 1 others. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, and 41 others. 2022. Lamda: Language models for dialog applications. *Preprint*, arXiv:2201.08239.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Fanjia Yan, Huanzhi Mao, Charlie Cheng-Jie Ji, Tianjun Zhang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2024. Berkeley function calling leaderboard. `https://gorilla.cs.berkeley.edu/blogs/8_berkeley_function_calling_leaderboard.html`.

Hui Yang, Sifu Yue, and Yunzhong He. 2023. Auto-gpt for online decision making: Benchmarks and additional opinions. *arXiv preprint arXiv:2306.02224*.

Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. 2024. Gpt4tools: Teaching large language model to use tools via self-instruction. *Advances in Neural Information Processing Systems*, 36.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

# A Appendix

## A.1 Models and Resources

Only instruction models are used in this paper. They are Meta-Llama-3-8B-Instruct, Mistral-7B-Instruct-v0.2, and Mistral-7B Qwen2.5-7B-Instruct. The label “Instruct” may be omitted for convenience.

The above model names contain the approximate number of parameters in the model as in "7B" signifying seven billion parameters. The Llama-3 models (Dubey et al., 2024) are licensed under the Meta Llama 3 Community License. The Mistral (Jiang et al., 2023) model is licensed under the Mistral AI non-production license. The Qwen models (Team, 2024) is licensed under the Apache 2.0 License, which permits both commercial and non-commercial use. Our use of these models have complied with the relevant license agreement and have been for scientific and non-commercial purposes.

The above models are publicly available, with registration, from HuggingFace at the specific model page of https://huggingface.co/ using the transformers library (Wolf et al., 2020). Up to two NVIDIA RTX A4000 graphics cards may be needed to generate texts through the vllm library using these models with generation with vllm sampling parameter n of 8 taking a few minutes, with most of the time dedicated to loading the model.

## A.2 Prompts

### Wikidata tool prompt

```
You are a temporal fact specialist that interfaces with structured knowledge bases. Your task is to:

1. Recognize time-sensitive country-related queries about: - Leadership positions (head of state, president, prime minister) - Demographic/economic metrics (population, GDP, unemployment rate) - Geographical facts (capital, borders, area) - Historical comparisons (past values, changes over time)

2. For valid temporal country queries: a. Output Wikidata API calls using this exact format: <token>(entity, property, qualifiers) b. Use official Wikidata property IDs when available (P1082=population, P36=capital) c. Always include temporal qualifiers: - Point in time (P585) for specific years - Series ordinal (P1545) for leadership positions - Retrieved date (P813) for latest values

3. For non-temporal or non-country queries: a. Provide normal conversational responses b. Never use the <token>() syntax

Follow the format of these examples:
```

### Choose 1 of 4 tools prompt

You are an assistant that can use specialized tools when appropriate. Choose among these tools by responding in the below format:

1. Calculator: For simple arithmetic operations. Format: `<math_args>(operator, num1, num2)`

2. Weather: For weather forecasts. Format: `<weather_args>(city, date)`

3. Flights: For flight bookings. Format: `<flight_args>(from, to, date)`

4. Wikidata: For time-sensitive country-related queries using Wikidata API calls. Format: `<wiki_args>(entity, property, qualifiers)`

When the above tools are not relevant, respond normally.
Today’s date (yyyy-mm-dd): `{today}`

Follow these examples:

**Calculator tool prompt with ICEs and user content** The prompt is constructed with an instruction on using the tool or tools including possible in-context examples (ICEs), followed by a user request that may or may not necessitate the use of a listed tool. See below example prompt with 4 ICEs for the calculator tool using token tool-trigger token "ithub". All prompts use HuggingFace’s tokenizer library’s `apply_chat_template` method, which adds `[INST]` and other similar tags.

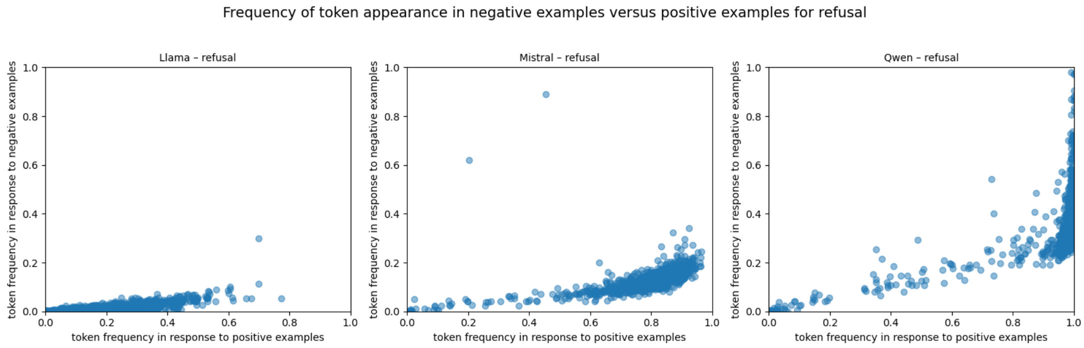



Figure 5

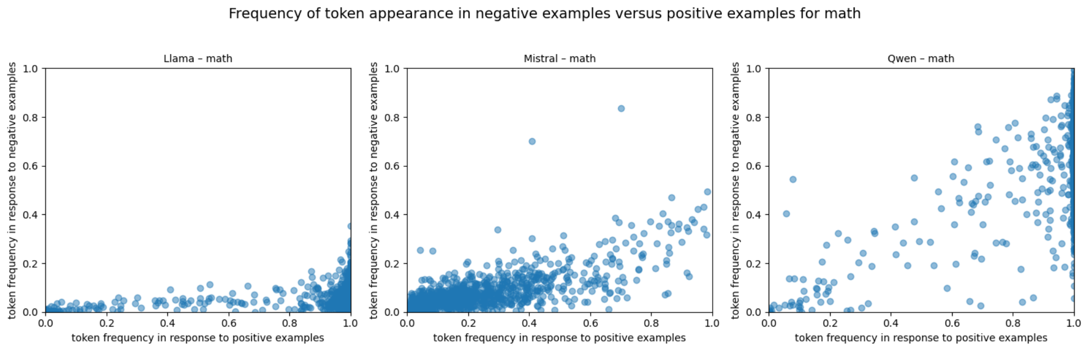



Figure 6

```
<s> [INST] You are an assistant specialized in recognizing if a problem contains an arithmetic
    operation. If the user asks a question involving no arithmetic operation, answer normally. But
    if the user asks a question containing an arithmetic operation, he may provide the input in
    various formats but you should provide the output only in this format:  <|>ithub<|> followed
    immediately by the operation symbol and arguments, separated by commas in parentheses. Do not
    add explanations, text, or commentary -- only return the formatted string. In case of questions
    containing an arithmetic operation, follow the format of these examples:
user: 'Subtract 38 from 62.'
assistant: ' <|>ithub<|> (-,62,38)'
user: 'Subtract 8 from 60.'
assistant: ' <|>ithub<|> (-,60,8)'
user: 'What is 72 * 58?'
assistant: ' <|>ithub<|> (*,72,58)'
user: 'Calculate 33 - 80.'
assistant: ' <|>ithub<|> (-,33,80)'

What is 32 - 75? [/INST]
```

## A.3 Results

### A.3.1 token frequency

As discussed in Section 7.2, Figures 5 to 10 show the variance in token choice for combinations of models, Llama-3-8B Instruct (Llama), Mistral-7B-Instruct-v0.2 (Mistral), Qwen2.5-7B-Instruct (Qwen) and tools (refusal, math recognition, calculator, flight, weather, wiki).

### A.3.2 token frequency correlation

### A.3.3 Token triggering

Here are the token appearances for other tools, other than calculator tool. True positive of 1.0 means that whenever there is an example necessitating the tool, the language model correctly calls that tool by emitting the designated token for that tool.

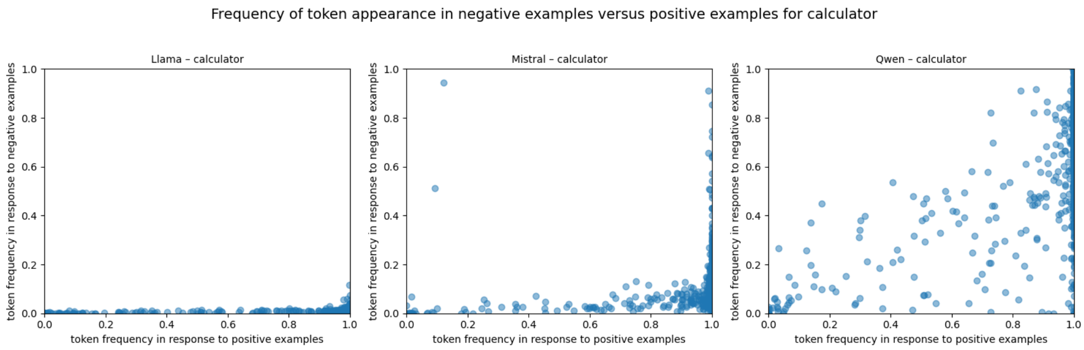



Figure 7

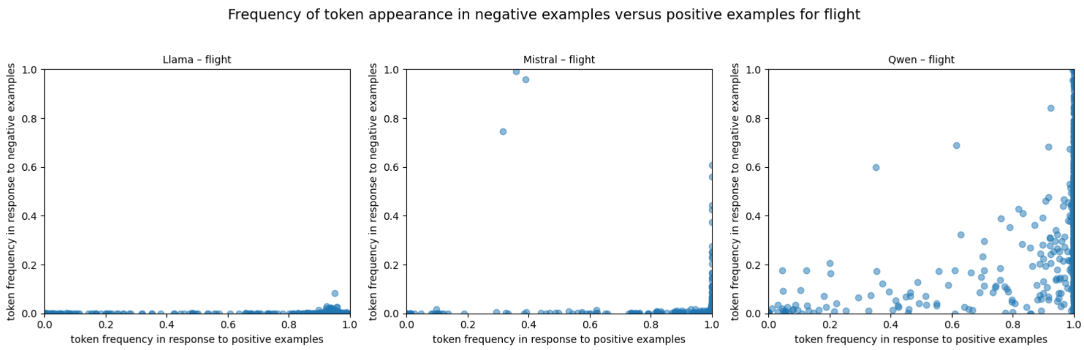



Figure 8

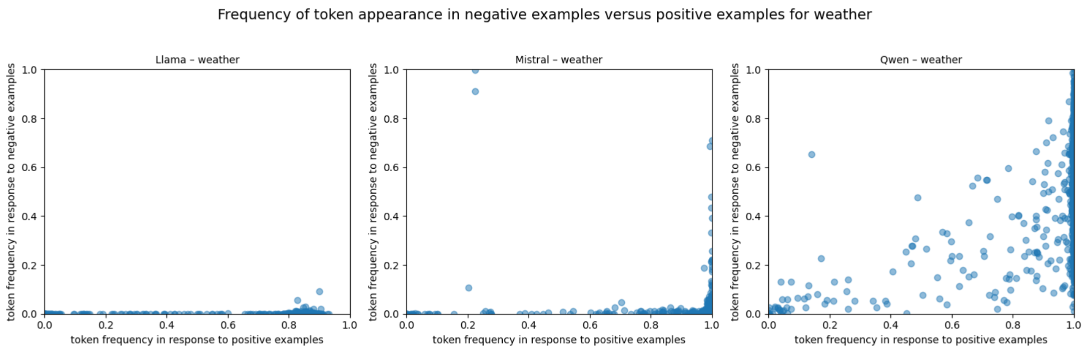



Figure 9

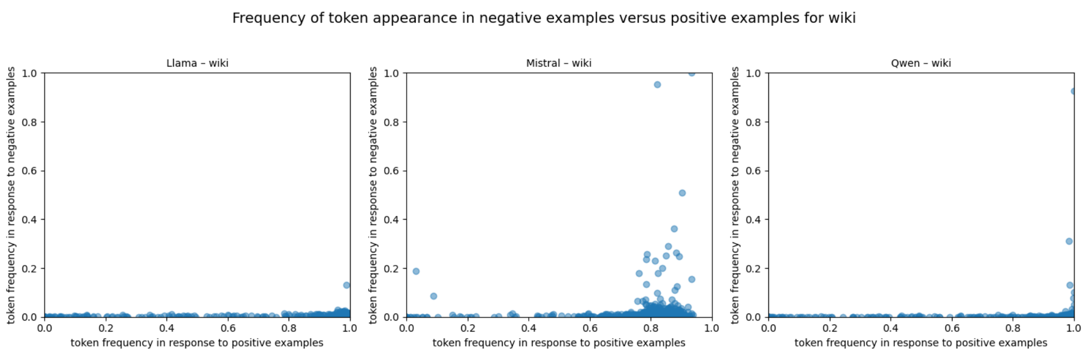



Figure 10

Table 4: Correlation matrix of token frequency across tools for Mistral.

| | calc | flight | math | refusal | weather | wiki |
|---|---|---|---|---|---|---|
| calc | 1.00 | 0.95 | 0.24 | 0.81 | 0.94 | 0.87 |
| flight | 0.95 | 1.00 | 0.23 | 0.79 | 0.98 | 0.88 |
| math | 0.24 | 0.23 | 1.00 | 0.34 | 0.22 | 0.27 |
| refusal | 0.81 | 0.79 | 0.34 | 1.00 | 0.78 | 0.77 |
| weather | 0.94 | 0.98 | 0.22 | 0.78 | 1.00 | 0.85 |
| wiki | 0.87 | 0.88 | 0.27 | 0.77 | 0.85 | 1.00 |

Table 5: Correlation matrix of token frequency across tools for Qwen.

| | calc | flight | math | refusal | weather | wiki |
|---|---|---|---|---|---|---|
| calc | 1.00 | 0.95 | 0.88 | 0.84 | 0.95 | 0.88 |
| flight | 0.95 | 1.00 | 0.88 | 0.82 | 0.99 | 0.92 |
| math | 0.88 | 0.88 | 1.00 | 0.92 | 0.88 | 0.83 |
| refusal | 0.84 | 0.82 | 0.92 | 1.00 | 0.82 | 0.78 |
| weather | 0.95 | 0.99 | 0.88 | 0.82 | 1.00 | 0.93 |
| wiki | 0.88 | 0.92 | 0.83 | 0.78 | 0.93 | 1.00 |

Table 6: Confusion matrices for `flight_tool` across three models. Given a user request that requires a tool/task (pos ex.) versus a user request that does not (neg ex.), the model response may emit the tool token (left column) or not (right column). Each value is the the average over 400 examples. The prompt contains positive examples in-context.

| | Llama3 | | Mistral | | Qwen | |
|---|---|---|---|---|---|---|
| | emit token | no token | emit token | no token | emit token | no token |
| pos ex | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 |
| neg ex | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 |

Table 7: Confusion matrices for `math` across three models. Given a user request that requires a tool/task (pos ex.) versus a user request that does not (neg ex.), the model response may emit the tool token (left column) or not (right column). Each value is the the average over 400 examples. The prompt contains positive examples in-context.

| | Llama3 | | Mistral | | Qwen | |
|---|---|---|---|---|---|---|
| | emit token | no token | emit token | no token | emit token | no token |
| pos ex | 0.995 | 0.005 | 0.855 | 0.145 | 1.0 | 0.0 |
| neg ex | 0.01 | 0.99 | 0.078 | 0.922 | 0.055 | 0.945 |

Table 8: Confusion matrices for `refusal` across three models. Given a user request that requires a tool/task (pos ex.) versus a user request that does not (neg ex.), the model response may emit the tool token (left column) or not (right column). Each value is the the average over 400 examples. The prompt contains positive examples in-context.

| | Llama3 | | Mistral | | Qwen | |
|---|---|---|---|---|---|---|
| | emit token | no token | emit token | no token | emit token | no token |
| pos ex | 0.772 | 0.228 | 0.942 | 0.058 | 0.99 | 0.01 |
| neg ex | 0.052 | 0.948 | 0.142 | 0.858 | 0.242 | 0.758 |

Table 9: Confusion matrices for `weather_tool` across three models. Given a user request that requires a tool/task (pos ex.) versus a user request that does not (neg ex.), the model response may emit the tool token (left column) or not (right column). Each value is the the average over 400 examples. The prompt contains positive examples in-context.

| | Llama3 | | Mistral | | Qwen | |
|---|---|---|---|---|---|---|
| | emit token | no token | emit token | no token | emit token | no token |
| pos ex. | 0.93 | 0.07 | 1.0 | 0.0 | 1.0 | 0.0 |
| neg ex. | 0.0 | 1.0 | 0.0 | 1.0 | 0.012 | 0.988 |

Table 10: Confusion matrices for `wiki_tool` across three models. Given a user request that requires a tool/task (pos ex.) versus a user request that does not (neg ex.), the model response may emit the tool token (left column) or not (right column). Each value is the the average over 400 examples. The prompt contains positive examples in-context.

| | Llama3 | | Mistral | | Qwen | |
|---|---|---|---|---|---|---|
| | emit token | no token | emit token | no token | emit token | no token |
| pos ex | 1.0 | 0.0 | 0.938 | 0.062 | 1.0 | 0.0 |
| neg ex | 0.0 | 1.0 | 0.008 | 0.992 | 0.0 | 1.0 |

## A.4 Ablation of number of ICEs

We performed ablation on number of in-context examples (2, 4, ..., 20). See Figure 11 for the three models' recall for the six tools. Llama-3-8B does better with more in-context examples whereas Mistral and Qwen have less noticeable differences among different number of ICEs.

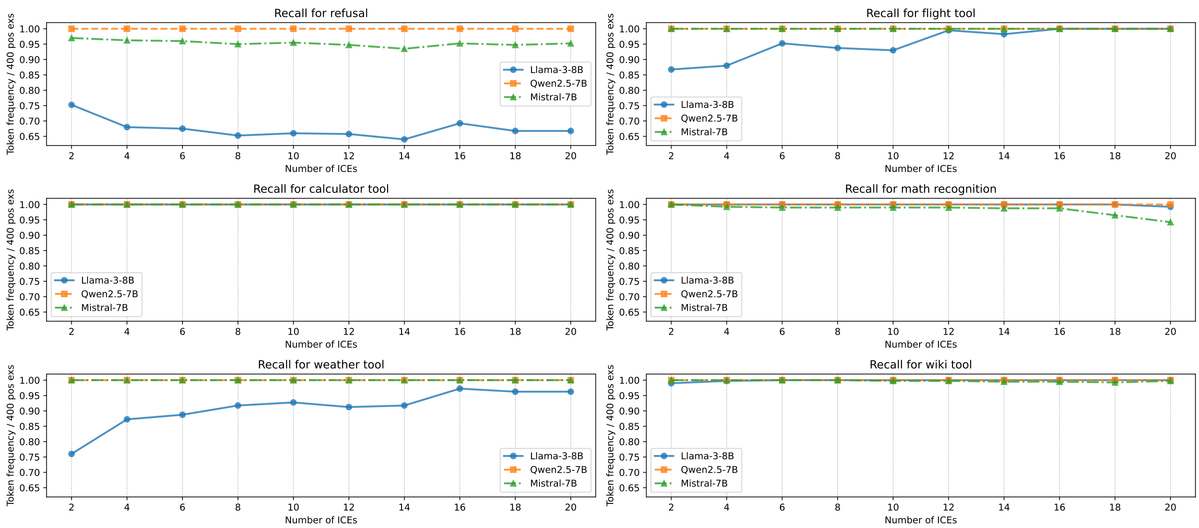



Figure 11: Ablation of number of ICEs for the six tools.

## A.5 LLM selecting the right tools among four tools

More likely than not, the LLM is able to pick out the correct tool in our admittedly small experiment of offering a choice of four tools. For weather tool and flight booking tool, all models are able to select the right tool more than 80 percent of the time. See Figure 12.

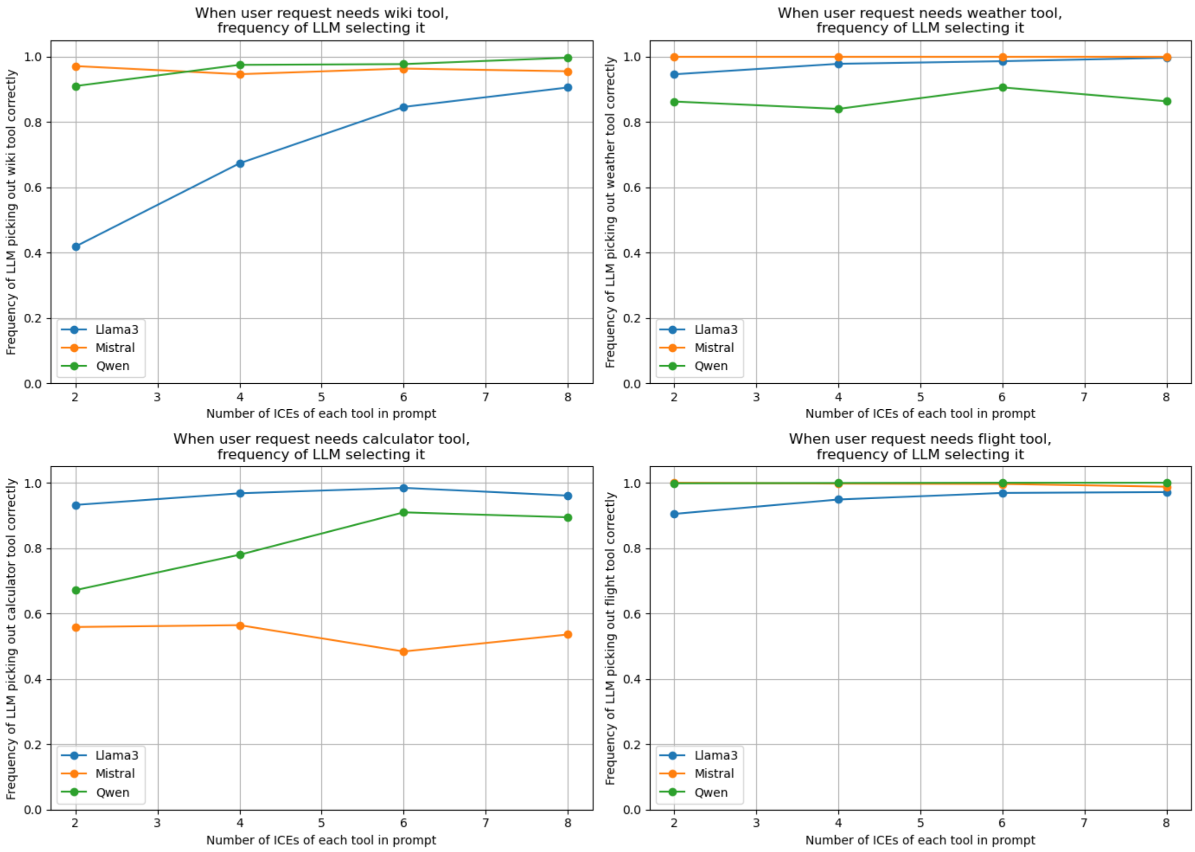



Figure 12: Given four tools, how often does the LLM pick out the right tool when the user request necessitates one of the tools.