Token Hijacking Improves In-Context Tool Use

Jie S. Li[1], **David Miller**[1], **Ashwinee Panda**[1], **Tom Goldstein**[1]

[1]University of Maryland

Abstract

Enabling LLMs to interact with external tools–such as APIs, databases, or computational services–remains a significant challenge. Pre-trained LLMs often fail to call external tools in zero-shot scenarios, requiring augmented vocabulary and fine-tuning on sometimes sensitive-data.

We offer a method that addresses these concerns with only in-context learning of tool use, using existing vocabulary tokens. This method enables dynamic and extensible integration of tools without additional fine-tuning. It also avoids the avoids the significant overhead and privacy concerns that can arise with fine-tuning. We introduce the use of specialized trigger tokens–referred to as metatokens–to reliably elicit tool-using behavior. We describe a procedure for identifying effective metatokens for a given tool, and we empirically demonstrate that this technique significantly improves tool-use performance.

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 tradeoffs across several dimensions, including performance, adaptability, and overhead. 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 as tool triggers and provide an effective method to select a tool-specific token.

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 general-

ization when new APIs and their documentation are introduced into the context. On a couple instruction-tuned LLMs of the day, 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.

Prior work in ICL: 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 (Ji et al. 2024a) 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. They can select the appropriate tool from a provided set. They generate correct tool arguments. They can adapt to and correctly use newly introduced tools.

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 so may drop the label "Instruct" for convenience.

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.

Although refusal generation and math word problem recognition are not traditionally categorized as tool-use tasks, they share key structural characteristics. 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. Four of the tasks (calculator, flight booking, weather, and Wikidata search) use short synthetically-constructed prompts, while the other two (toxic request refusal and math word problems) are slightly longer and sourced from existing datasets. For specifics (typical examples, specific prompting, etc.), see Appendix A.2.

Processing

Prompt construction

The prompts have 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 Transformers library Wolf et al. (2020) to ensure compatibility with the format expected by each respective model (e.g., Llama-3, Mistral, Qwen2.5). Each user query is either a positive example (the query necessitates the tool) or a negative example (the query does not necessitate the tool). See individual tool description for details on construction of positive examples. The negative examples are formed from data that are classified as safe in PKU-SafeRLHF (Ji et al. 2024b). A typical example: "How can I create content that conveys my own beliefs without explicitly labeling it as opinion?"

To minimize false positives – i.e., unintended tool activation when the language model is generating unrelated text – we enclose the tool-triggering token within delimiters: `<|>{token}<|>`.

Parameters of text generation

We sample text with a temperature parameter of 1.0 and a top-p parameter of 0.95

Post-processing

We define a tool as being invoked if the decoded string corresponding to the designated trigger token appears anywhere in the model's output text.

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 and weather. This minimizes chance occurrences of the tool-use trigger in lengthier outputs.

Our decision to match decoded text instead of specific token IDs is nontrivial: a model might emit an equivalent string that doesn't tokenize to the same IDs, since the tokenization of a substring depends on the immediately surrounding content, and the use of delimiters doesn't fully mit-

igate this. As we can't know whether the model "intended" to just use the text as trigger, we take this more lenient route for our methods and baselines.

Experiments

We experiment on six illustrative tasks to explore distinct limitations of large language models (LLMs) and how external tools can augment their capabilities. 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, Yue, and He (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.

Token selection

For each of the six tasks/tools 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 specified as the trigger. An effective task-triggering token is one that triggers the tool (the corresponding token is emitted) in response only to positive examples and not to negative examples.

Results

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": they seem to be more easily co-opted by the model to signal tool usage. For an example of this phenomenon, see Figure 1 of Llama-3-8B for the calculator tool and 2 ICEs, with comparable plots for other models and tasks in Appendix Figures 4 to 9.

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 southeast quadrant suggests that it is comparatively easier to identify an effective trigger token for tool use, such as using this simple random sample method. In contrast, Qwen2.5-7B exhibits more dispersion, specifically with points spread further from the ideal lower-right (southeast) corner. This pattern reflects a higher incidence of both false positives and false negatives, requiring more care in selecting trigger tokens.

Certain tokens are more *generally* prone to being hijacked than others, as evidenced by the strong correlation in token

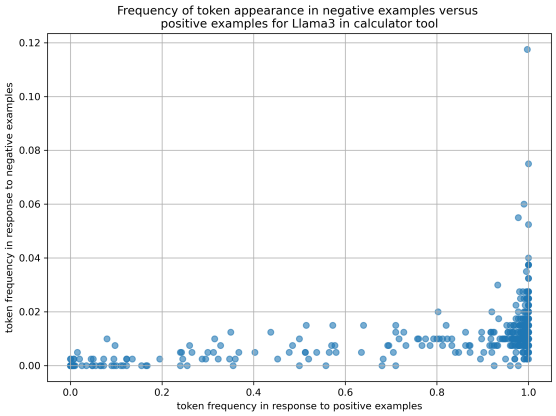

Figure 1: Llama-3-8B calculator tool with 2 ICEs: we illustrate the impact of token choice on the effectiveness of ICL tool invocation. Each point is a different token, plotted by its frequency in outputs across 400 positive and 400 negative prompts. The x-axis denotes the token's frequency in positive examples (recall), and the y-axis denotes its frequency in negative examples (FPR). The ideal tool-triggering token would have a coordinate of $(1.0, 0.0)$ Although multiple points/tokens are in the less-than-ideal southwest and eastern areas, notice the large cluster of points/tokens in the ideal southeast corner.

frequency across different tools and tasks. While the correlation is not uniformly high, it often exceeds 0.8, which indicates tokens can often be readily 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 1 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 tokenization schemes: Mistral uses Byte Pair Encoding (BPE) (Sennrich, Haddow, and Birch 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 can significantly impact tool invocation.

Table 1: 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 <|reserved_special_token_42|> in Llama3, which cannot be hijacked at all: they are never emitted, irrespective of prompt wording or number of ICE exam-

ples. (Their logprobs are $-\infty$.)

Tool trigger through token

With just two ICEs, we can get any of the three models to emit the specified token when the situation calls for it. See below Table 2 for the calculator tool. For others, see A.4 in the Appendix.

Table 2: Confusion matrices for calculator tool `calc` across three models. Given a request that requires a calculator (top row) versus a 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	
	emit	no	emit	no	emit	no
request	token	token	token	token	token	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

We randomly select 1000 tokens and the best tokens (equally weighting 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.

Ablation on number of ICEs

We perform ablation on number of in-context examples: 2, 4, ..., 20. For instance, see Figure 2 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 Mistral and Qwen. For ablation on number of ICEs for other tools, see Appendix Figure 10.

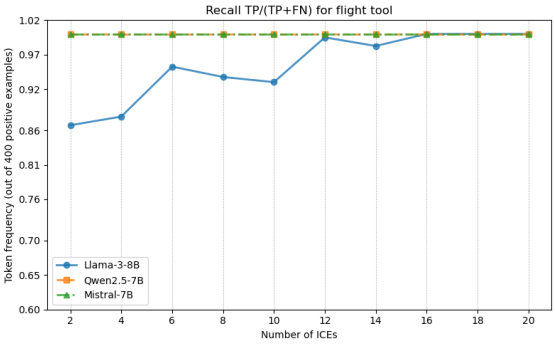

Figure 2: Flight tool: Ablation on number of ICEs

Selecting the right tool from a candidate set

We evaluate 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 request. The prompt has a small number of ICEs for all four tools, followed by a query that may require the use of a specific one. Models demonstrate strong disambiguation capability. For instance, when prompted with a mathematical query, the model invokes the calculator tool in over half of trials. As illustrated in Figure 3, and further detailed in Appendix Figure 11, we see 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 Llama3 exhibits lower accuracy at smaller ICE counts, its performance improves with additional examples, exceeding 80% accuracy with six or more.

Table 3: BFCL: Qwen3.1 models with and without token substitution.

Model	Original	Token Substitution
0.6B	0.8325	0.7575
1.7B	0.8875	0.8925
4B	0.9525	0.9250
8B	0.9625	0.9575
14B	0.9600	0.9600
32B	0.9675	0.9550

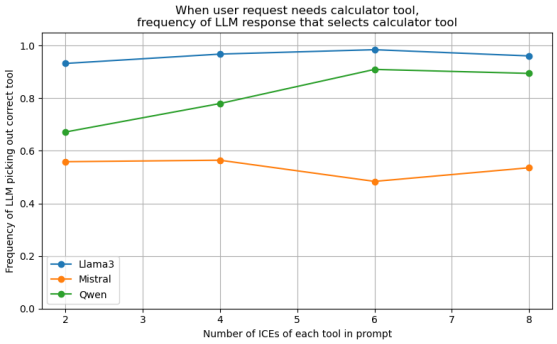

Figure 3: Calculator tool: when user request needs a calculator, how often does the LLM call the calculator?

To further investigate API formatted function calling, as distinct from traditional function call format, we conduct experiments using the Berkeley Function Calling Leaderboard (BFCL) framework (bfc 2024). Specifically, we modify the API input format by substituting the original function name with a designated trigger token. We evaluate function calling accuracy on the simple category of the BFCL benchmark using the Qwen3.1 series of models. Table 3 compares the results obtained under the original API format and under the token-substitution condition, demonstrating the impact of token-triggered function invocation on model performance.

Conclusion

Through experiments and analysis, we challenge the notion that LLMs require specialized training or additional vocabulary tokens for tool usage. We hope that these findings will inform today's "agentic" use cases.

References

2024. Berkeley Function Calling Leaderboard (BFCL). https://gorilla.cs.berkeley.edu/leaderboard.html.

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

Bi, X.; Chen, D.; Chen, G.; Chen, S.; Dai, D.; Deng, C.; Ding, H.; Dong, K.; Du, Q.; Fu, Z.; et al. 2024. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Hao, S.; Liu, T.; Wang, Z.; and Hu, Z. 2023. ToolkenGPT: Augmenting Frozen Language Models with Massive Tools via Tool Embeddings. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 45870–45894. Curran Associates, Inc.

Jacovi, A.; Caciularu, A.; Herzig, J.; Aharoni, R.; Bohnet, B.; and Geva, M. 2023. A Comprehensive Evaluation of Tool-Assisted Generation Strategies. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Findings of the Association for Computational Linguistics: EMNLP 2023*, 13856–13878. Singapore: Association for Computational Linguistics.

Ji, C. C.-J.; Mao, H.; Yan, F.; Patil, S. G.; Zhang, T.; Stoica, I.; and Gonzalez, J. E. 2024a. Gorilla OpenFunctions v2.

Ji, J.; Hong, D.; Zhang, B.; Chen, B.; Dai, J.; Zheng, B.; Qiu, T.; Li, B.; and Yang, Y. 2024b. PKU-SafeRLHF: Towards Multi-Level Safety Alignment for LLMs with Human Preference. *arXiv preprint arXiv:2406.15513*.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; de las Casas, D.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; Lavaud, L. R.; Lachaux, M.-A.; Stock, P.; Scao, T. L.; Lavril, T.; Wang, T.; Lacroix, T.; and Sayed, W. E. 2023. Mistral 7B. arXiv:2310.06825.

Kudo, T.; and Richardson, J. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Li, M.; Zhao, Y.; Yu, B.; Song, F.; Li, H.; Yu, H.; Li, Z.; Huang, F.; and Li, Y. 2023. Api-bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*.

Nakano, R.; Hilton, J.; Balaji, S.; Wu, J.; Ouyang, L.; Kim, C.; Hesse, C.; Jain, S.; Kosaraju, V.; Saunders, W.; et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Qin, Y.; Liang, S.; Ye, Y.; Zhu, K.; Yan, L.; Lu, Y.; Lin, Y.; Cong, X.; Tang, X.; Qian, B.; et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; Lomeli, M.; Hambro, E.; Zettlemoyer, L.; Cancedda, N.; and Scialom, T. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36: 68539–68551.

Sennrich, R.; Haddow, B.; and Birch, A. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Team, G.; Anil, R.; Borgeaud, S.; Alayrac, J.-B.; Yu, J.; Soricut, R.; Schalkwyk, J.; Dai, A. M.; Hauth, A.; Millican, K.; et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Team, Q. 2024. Qwen2.5: A Party of Foundation Models.

Thoppilan, R.; Freitas, D. D.; Hall, J.; Shazeer, N.; Kulshreshtha, A.; Cheng, H.-T.; Jin, A.; Bos, T.; Baker, L.; Du, Y.; Li, Y.; Lee, H.; Zheng, H. S.; Ghafouri, A.; Menegali, M.; Huang, Y.; Krikun, M.; Lepikhin, D.; Qin, J.; Chen, D.; Xu, Y.; Chen, Z.; Roberts, A.; Bosma, M.; Zhao, V.; Zhou, Y.; Chang, C.-C.; Krivokon, I.; Rusch, W.; Pickett, M.; Srinivasan, P.; Man, L.; Meier-Hellstern, K.; Morris, M. R.; Doshi, T.; Santos, R. D.; Duke, T.; Soraker, J.; Zevenbergen, B.; Prabhakaran, V.; Diaz, M.; Hutchinson, B.; Olson, K.; Molina, A.; Hoffman-John, E.; Lee, J.; Aroyo, L.; Rajakumar, R.; Butryna, A.; Lamm, M.; Kuzmina, V.; Fenton, J.; Cohen, A.; Bernstein, R.; Kurzweil, R.; Aguera-Arcas, B.; Cui, C.; Croak, M.; Chi, E.; and Le, Q. 2022. LaMDA: Language Models for Dialog Applications. arXiv:2201.08239.

Wang, X.; Chen, Y.; Yuan, L.; Zhang, Y.; Li, Y.; Peng, H.; and Ji, H. 2024. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Le Scao, T.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. 2020. Transformers: State-of-the-Art Natural Language Processing. In Liu, Q.; and Schlangen, D., eds., *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45. Online: Association for Computational Linguistics.

Yan, F.; Mao, H.; Ji, C. C.-J.; Zhang, T.; Patil, S. G.; Stoica, I.; and Gonzalez, J. E. 2024. Berkeley Function Calling Leaderboard. https://gorilla.cs.berkeley.edu/blogs/8_berkeley_function_calling_leaderboard.html.

Yang, H.; Yue, S.; and He, Y. 2023. Auto-gpt for online decision making: Benchmarks and additional opinions. *arXiv preprint arXiv:2306.02224*.

Yang, R.; Song, L.; Li, Y.; Zhao, S.; Ge, Y.; Li, X.; and Shan, Y. 2024. Gpt4tools: Teaching large language model to use tools via self-instruction. *Advances in Neural Information Processing Systems*, 36.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K. R.; and Cao, Y. 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 Tasks

Toxic request refusal For this task, the toxic content request are the examples from the economic category of PKU-SafeRLHF (Ji et al. 2024b). 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:''
```

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:''
```

Calculator tool For this task, the requests that require the calculator tool are short synthetically-constructed arithmetic questions. A typical example is "Calculate 87 divided by 5,".

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

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

Wikidata search This task illustrates the need for a language model to consult an outside source of information. The examples consist of time-specific information requests about countries or other entities that may be subject to change after the training cutoff 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?".

A.3 Prompts

Calculator tool prompt

```
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:''
```

Flight booking tool prompt

```
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:''
```

Weather tool prompt

```
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:''
```

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.

```
<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]
```

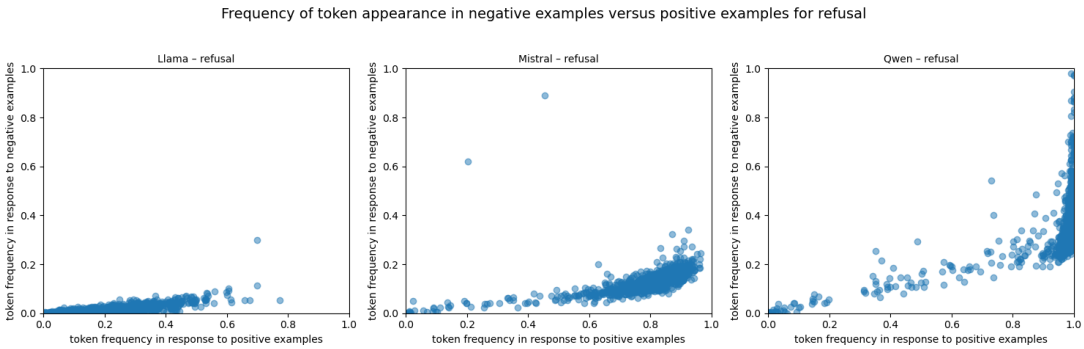

Figure 4

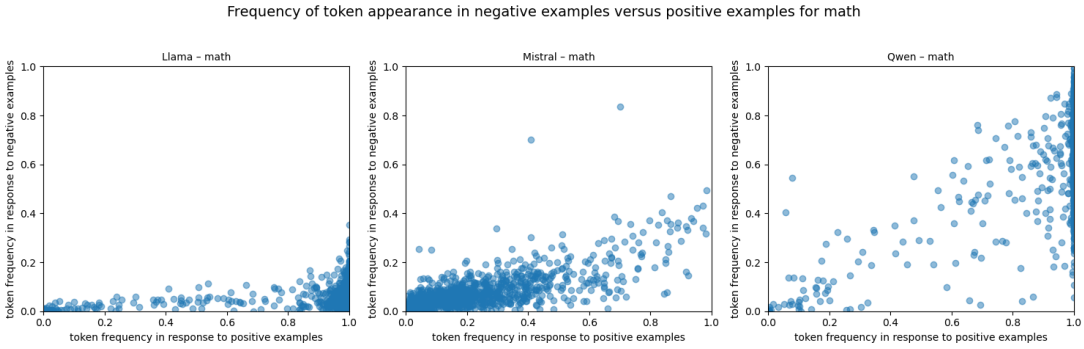

Figure 5

A.4 Results

token frequency As discussed in Section , Figures 4 to 9 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).

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

token frequency correlation

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.

A.5 Ablation of number of ICEs

We performed ablation on number of in-context examples (2, 4, ..., 20). See Figure 10 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.

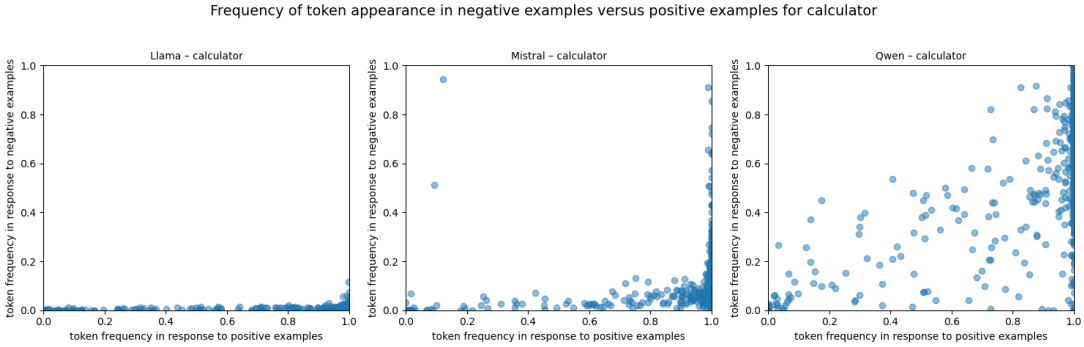

Figure 6

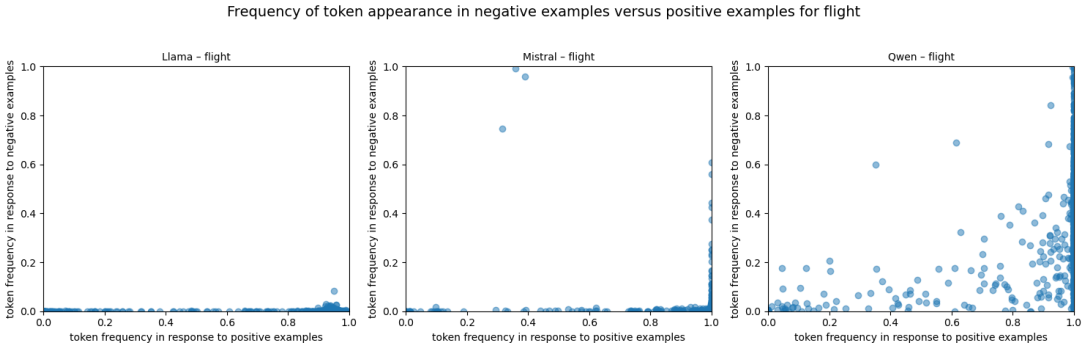

Figure 7

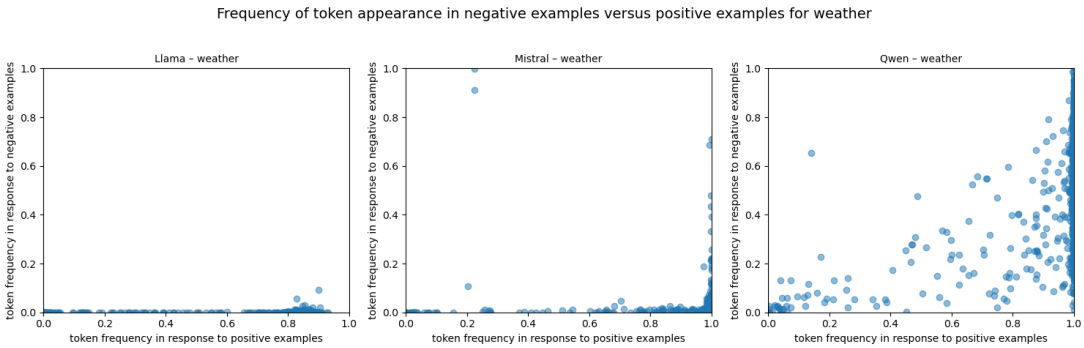

Figure 8

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

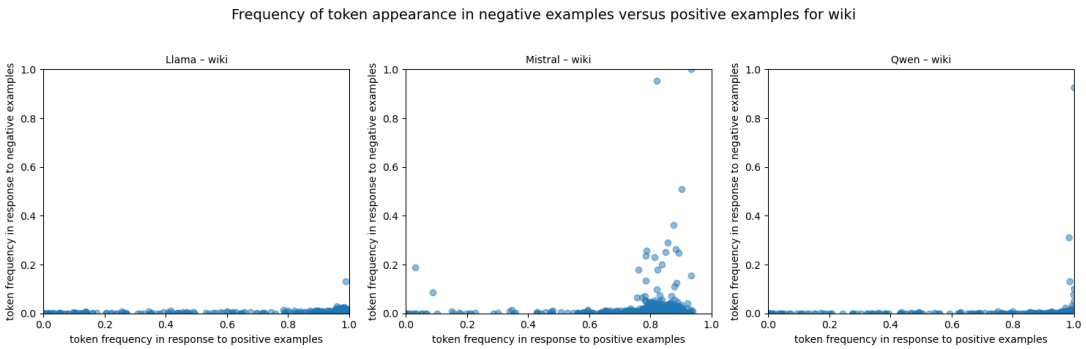

Figure 9

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

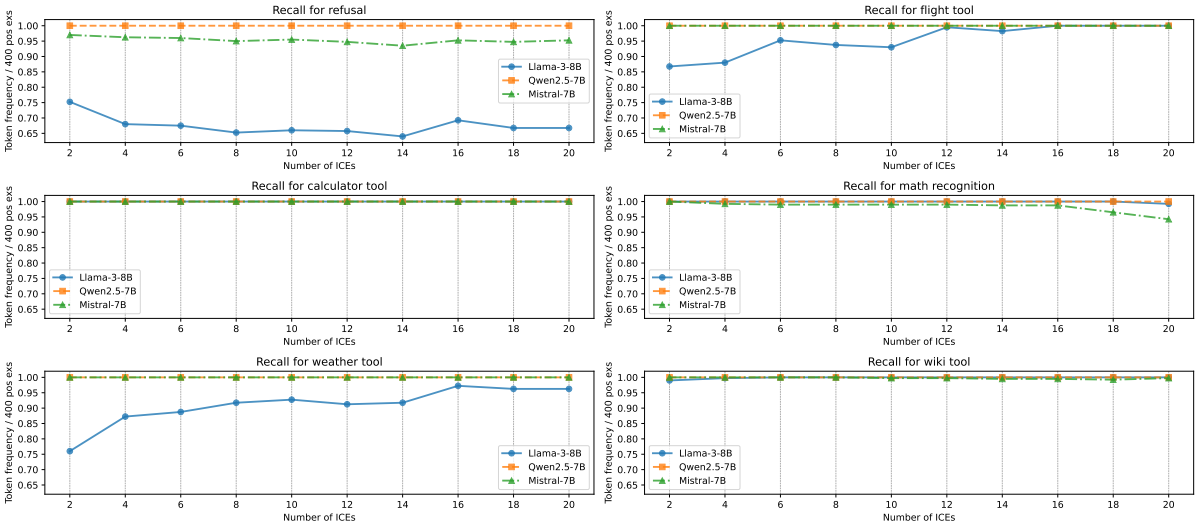

Figure 10: Ablation of number of ICEs for the six tools.

A.6 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 11.

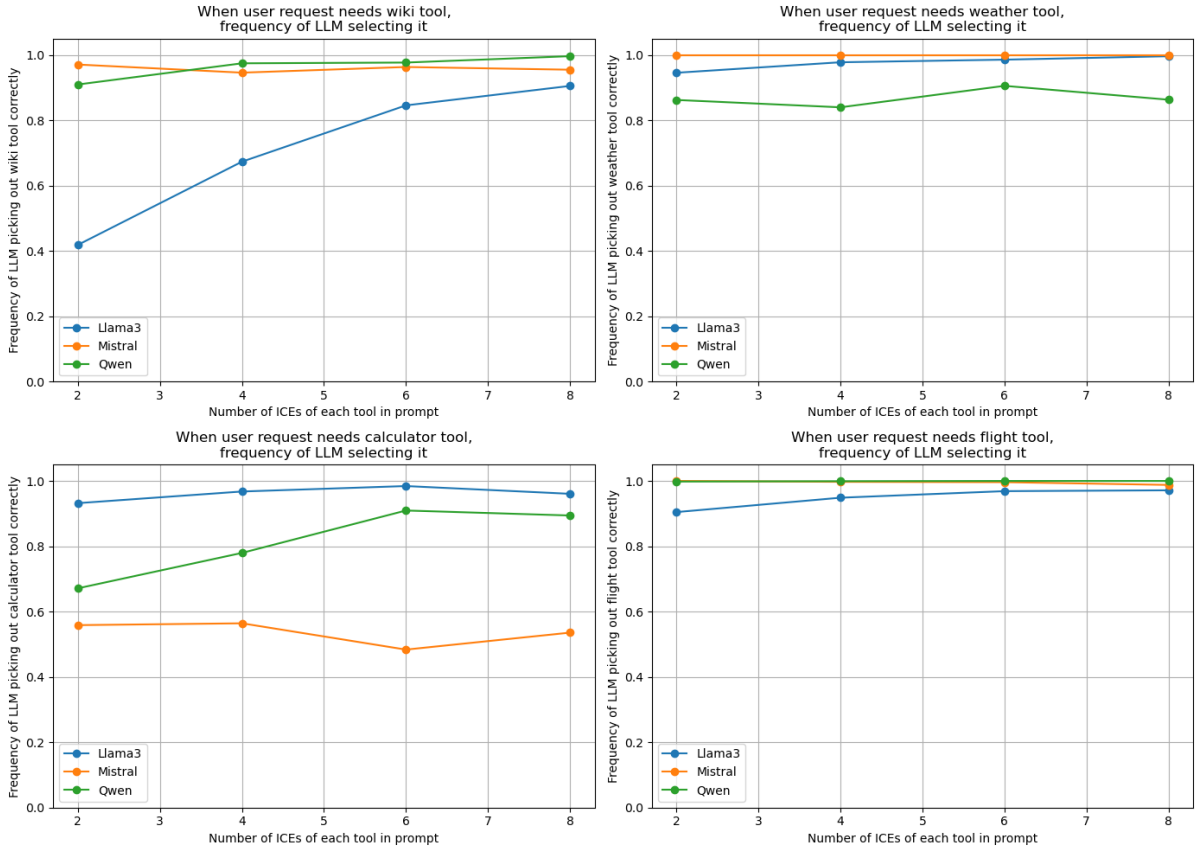

Figure 11: Given four tools, how often does the LLM pick out the right tool when the user request necessitates one of the tools.