AgentSHAP: Interpreting LLM Agent Tool Importance with Monte Carlo Shapley Value Estimation

Miriam Horovicz
Fiverr Labs
miriam@f-labs.io

Abstract

LLM agents that use external tools can solve complex tasks, but understanding which tools actually contributed to a response remains a blind spot. No existing XAI methods address tool-level explanations. We introduce **AgentSHAP**, the first framework for explaining tool importance in LLM agents. AgentSHAP is model-agnostic: it treats the agent as a black box and works with any LLM (GPT, Claude, Llama, etc.) without needing access to internal weights or gradients. Using Monte Carlo Shapley values, AgentSHAP tests how an agent responds with different tool subsets and computes fair importance scores based on game theory. Our contributions are: (1) the first explainability method for agent tool attribution, grounded in Shapley values from game theory; (2) Monte Carlo sampling that reduces cost from $O(2^n)$ to practical levels; and (3) comprehensive experiments on API-Bank showing that AgentSHAP produces consistent scores across runs, correctly identifies which tools matter, and distinguishes relevant from irrelevant tools. AgentSHAP joins TokenSHAP (for tokens) and PixelSHAP (for image regions) to complete a family of Shapley-based XAI tools for modern generative AI.

Code: `https://github.com/GenAISHAP/TokenSHAP`.

Introduction

Large Language Model (LLM) agents can now use external tools calculators, databases, web search, and APIs [12, 9]. This makes them far more capable than standalone LLMs. An agent with access to a calculator can solve math problems accurately. An agent with stock market APIs can provide real-time financial information. An agent with Wikipedia access can answer factual questions with citations.

But with this power comes a new challenge: *Which tools actually matter for a given response?*

This question matters for two key reasons. First, **optimization**: system designers need to decide which tools to include, since more tools mean higher costs and latency. Logs show which tools were called, but not how much each tool actually mattered for the response. A tool might be called frequently yet contribute little. Second, **understanding**: developers need to verify that the right tools are contributing. If a Calculator tool has low importance on a math query, something is wrong with the agent's behavior.

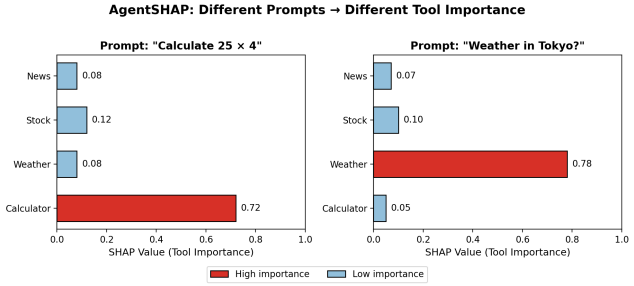

Figure 1: **AgentSHAP shows which tools matter.** Different prompts activate different tools. For a math query, Calculator scores highest (0.72). For a weather query, Weather tool scores highest (0.78). Red bars indicate high importance, blue bars indicate low importance.

Current XAI methods don't help here. They focus on input tokens [3], attention patterns [17], or feature importance for tabular data [6]. To our knowledge, no existing explainability framework addresses tool-level attribution in LLM agents.

We introduce **AgentSHAP**, the first framework that explains tool importance in LLM agents. A key advantage is that AgentSHAP is model-agnostic: it treats the agent as a black box, requiring only input-output access. This means it works equally well with local models you run yourself or third-party APIs, without needing internal weights, gradients, or architecture details.

The key idea is simple: run the agent with different subsets of tools and see how the response changes. Tools that cause big changes when removed are important; tools that don't matter get low scores. Figure 1 shows example results.

We make the following contributions:

1. **AgentSHAP framework**: We propose the first method to compute tool importance scores in LLM agents using Shapley values. The scores satisfy key game-theoretic properties: efficiency, symmetry, and null player.
2. **Efficient Monte Carlo estimation**: Computing exact Shapley values requires 2^n evaluations. We use Monte Carlo sampling to reduce this to practical levels while maintaining accuracy.
3. **Comprehensive evaluation**: We test AgentSHAP on the

API-Bank benchmark with real executable tools across four experiments: consistency, faithfulness, irrelevant tool injection, and cross-domain attribution.

4. **Open-source release**: AgentSHAP is available as part of the TokenSHAP library at `https://github.com/GenAISHAP/TokenSHAP`.

Related Work

LLM Agents with Tools. The capability of LLMs to use external tools has been a major research focus [7, 18, 19]. Toolformer [12] pioneered self-supervised tool learning, while ReAct [20] introduced interleaved reasoning and acting. Gorilla [8] demonstrated accurate API calling, and HuggingGPT [14] orchestrates multiple AI models as tools. Recent work on autonomous agents [15] shows agents can learn from feedback. Benchmarks like API-Bank [4] and ToolBench [10] provide standardized evaluation with real executable APIs. Despite this progress, no existing work addresses explaining *which* tools contributed to agent responses.

Shapley Values for Explainability. Shapley values from cooperative game theory [13] provide a principled, axiomatic approach to fair attribution. SHAP [6] popularized this for machine learning feature importance, with extensions to tree models [5]. The key challenge is computational: exact Shapley values require $O(2^n)$ coalition evaluations. Monte Carlo sampling [1] addresses this by estimating values through random permutations, trading exactness for efficiency.

Explainability for LLMs. Explaining LLM behavior remains challenging. Attention visualization [17] shows what tokens the model attends to, but attention doesn't always reflect importance. Integrated Gradients [16] and LIME [11] provide input attribution but require model access. TokenSHAP [3] uses Monte Carlo Shapley values to explain which input tokens matter for LLM outputs in a model-agnostic way. PixelSHAP [2] extends this to vision-language models. AgentSHAP completes this family by explaining tool importance a new dimension of explainability unique to agentic systems.

Method

An LLM agent $\mathcal{A}$ has access to a set of tools $T = \{t_1, t_2, \ldots, t_n\}$. Each tool t_i has a name, description, and executable function. Given a user prompt p, the agent produces a response $r = \mathcal{A}(p, T)$ by potentially calling one or more tools. Figure 2 illustrates the overall approach.

Our goal is to compute importance scores $\phi = \{\phi_1, \phi_2, \ldots, \phi_n\}$ where ϕ_i quantifies how much tool t_i contributed to the final response. These scores should be fair (tools that contribute equally get equal scores), complete (accounting for all contribution), and interpretable (higher scores mean more importance).

Shapley Values for Tools

We formulate tool attribution as a cooperative game where the tools $T = \{t_1, \ldots, t_n\}$ are the players. A coalition is any subset of tools $S \subseteq T$, and the value function $v(S) = \text{sim}(\mathcal{A}(p, S), \mathcal{A}(p, T))$ measures how similar the response with only tools S is to the full response with all tools T. We use cosine similarity on text embeddings (text-embedding-3-large) to capture semantic meaning rather than surface-level word overlap.

Algorithm 1 Monte Carlo Shapley for Tools

Require: Tools T, prompt p, sampling ratio ρ
Ensure: Shapley values $\phi_1, \ldots, \phi_n$

```
Get baseline response: r_base ← A(p, T)
Initialize: φ_i ← 0 for all i
for each tool t_i ∈ T do
    Compute leave-one-out: v(T \ {t_i})
end for
m ← ⌊ρ · (2^n − n − 1)⌋
for j = 1 to m do
    Sample random subset S ⊂ T
    Get response: r_S ← A(p, S)
    Compute similarity: v(S) ← sim(r_S, r_base)
    Update marginal contributions for each tool
end for
Compute final φ_i from Equation 1
return φ_1, ..., φ_n
```

The Shapley value for tool t_i is:

$$\phi_i = \sum_{S \subseteq T \setminus \{t_i\}} \frac{|S|!(n - |S| - 1)!}{n!} [v(S \cup \{t_i\}) - v(S)] \quad (1)$$

This formula averages the marginal contribution of tool t_i across all possible orderings in which tools could be added. The weighting ensures fairness: each ordering is equally likely. Shapley values satisfy key axioms (efficiency, symmetry, null player, linearity) that make them the unique fair attribution method [6].

Monte Carlo Estimation

Computing exact Shapley values requires evaluating 2^n coalitions, which is infeasible for agents with many tools. We use Monte Carlo sampling to estimate values efficiently.

Our algorithm (Algorithm 1) has two phases:

1. **Leave-one-out**: We always test removing each tool individually. This ensures every tool's direct effect is measured.
2. **Random sampling**: We sample additional random coalitions to capture tool interactions and improve estimates.

With sampling ratio ρ, we evaluate approximately $n + \rho \cdot (2^n - n - 1)$ coalitions instead of 2^n. For $n = 8$ tools and $\rho = 0.5$, this means 130 evaluations instead of 256.

Experiments

We evaluate AgentSHAP on the API-Bank benchmark [4], which provides real executable tools and ground-truth annotations. We use GPT-4o-mini as the LLM and text-embedding-3-large for semantic similarity.

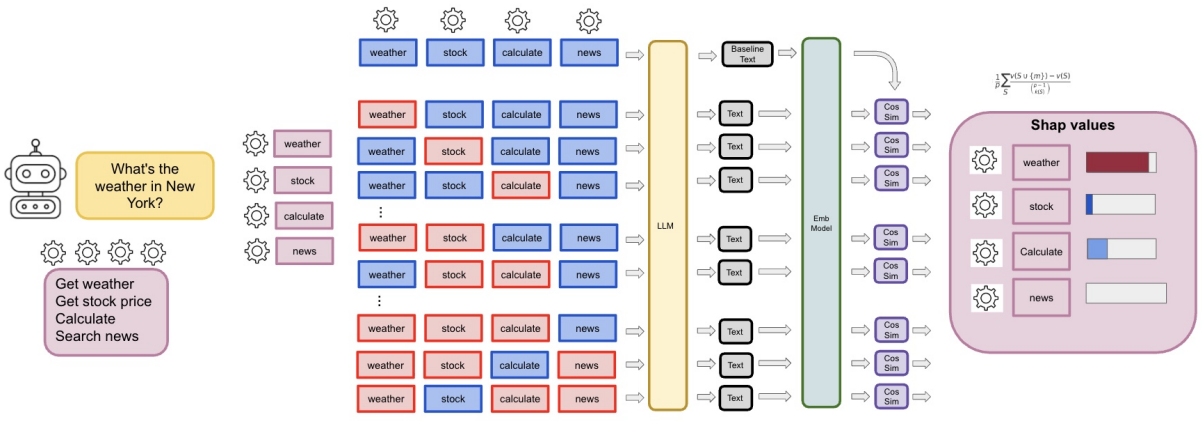

Figure 2: **How AgentSHAP works.** (1) Run the agent with all tools to get a baseline response. (2) Test the agent with different tool subsets using Monte Carlo sampling. (3) Compute Shapley values measuring each tool's contribution to response quality. The value function uses semantic similarity (cosine similarity on embeddings) to compare responses.

Experimental Setup

We use 8 tools from API-Bank: Calculator for arithmetic, QueryStock for stock prices, Wiki for Wikipedia search, plus AddAlarm, AddReminder, PlayMusic, BookHotel, and Translate. We use real queries from API-Bank's level-1 test set covering math calculations, stock price queries, and Wikipedia searches. We set sampling ratio $\rho = 0.5$ and run each experiment 3 times to measure consistency. We report Top-1 Accuracy (does the highest-SHAP tool match the expected tool), Cosine Similarity (how stable are SHAP vectors across runs), Quality Drop (response quality decrease when removing a tool), and SHAP Gap (difference between relevant and irrelevant tool scores).

Consistency

We test whether AgentSHAP results are stable across runs despite the stochastic nature of Monte Carlo sampling. We run AgentSHAP 3 times on 9 prompts with 3 tools available. Figure 3 shows the results: for the math query "Calculate (5+6)*3", Calculator consistently receives the highest SHAP value (0.74-0.98) across all runs, and for stock queries, QueryStock consistently scores highest (0.79-0.84). The mean cosine similarity between SHAP vectors across runs is 0.945, indicating high stability, and top-1 accuracy is 100%. This confirms Monte Carlo sampling provides stable estimates.

Faithfulness

We test whether SHAP scores reflect actual tool importance by removing tools and measuring response quality change. For each prompt, we remove the highest-SHAP tool and the lowest-SHAP tool separately, then measure quality drop (semantic similarity to original response). Figure 4 shows the results: removing high-SHAP tools causes mean quality drop of 0.67, while removing low-SHAP tools causes only

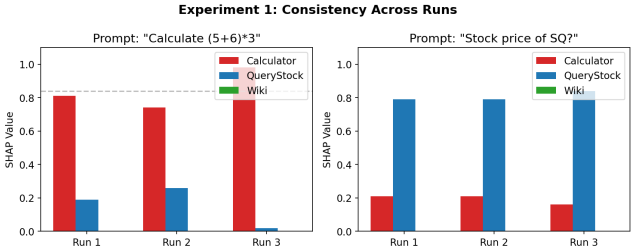

Figure 3: **Consistency across runs.** SHAP values are stable despite Monte Carlo sampling. Left: Math query consistently assigns high importance to Calculator. Right: Stock query consistently assigns high importance to QueryStock.

0.05 drop, a 13x difference. This confirms SHAP values accurately reflect which tools matter.

Irrelevant Tool Injection

We test whether AgentSHAP can distinguish the actually-used tool from irrelevant tools by adding 4 unrelated tools (AddAlarm, AddReminder, PlayMusic, BookHotel) to the 3 core tools. Figure 5 shows the results: the expected tool (Calculator for math, QueryStock for finance, Wiki for knowledge) receives mean SHAP of 0.53 while irrelevant tools receive only 0.07, a 7x difference. Top-1 accuracy is 86% (6/7 prompts correctly identify the expected tool). This demonstrates AgentSHAP can identify unnecessary tools for automatic pruning.

Cross-Domain Attribution

We test whether AgentSHAP correctly attributes importance across different query domains (Math, Finance, Knowledge) with 6 tools available. Figure 6 shows the results as a heatmap: Math queries assign highest score to Calculator

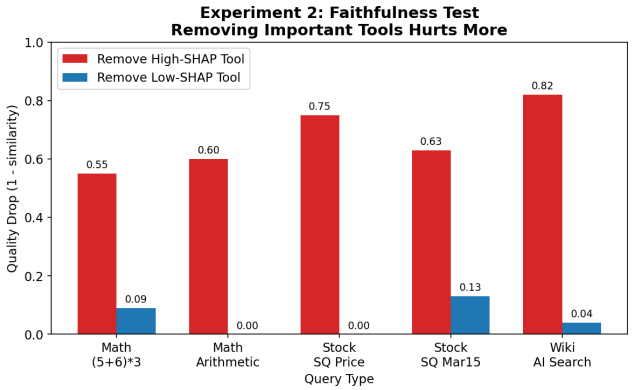

Figure 4: **Faithfulness test.** Removing high-SHAP tools (red) causes much larger quality drops than removing low-SHAP tools (blue). This confirms SHAP values reflect actual tool importance.

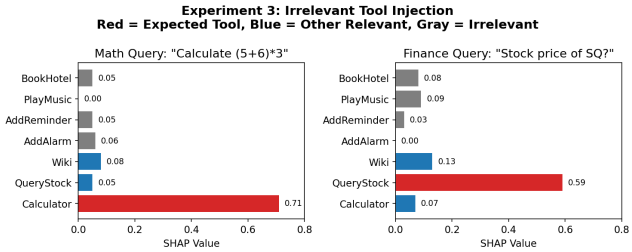

Figure 5: **Irrelevant tool injection.** AgentSHAP assigns low scores to injected irrelevant tools (gray). The expected tool (red) receives the highest score. Left: Math query. Right: Finance query.

(0.45), Finance queries to QueryStock (0.55), and Knowledge queries to Wiki (0.57). Overall top-1 accuracy is 86%, showing AgentSHAP correctly identifies domain-relevant tools.

Summary of Results

Our experiments demonstrate that AgentSHAP provides reliable and meaningful tool importance scores. The consistency experiment shows that Monte Carlo sampling produces stable results with 0.945 mean cosine similarity between runs. The faithfulness experiment confirms that SHAP values reflect actual importance with 13x quality difference when removing important versus unimportant tools. The irrelevant tool injection experiment shows AgentSHAP assigns the expected tool 7x higher scores than irrelevant ones. The cross-domain experiment demonstrates 86% accuracy across different query types. Together, these results validate AgentSHAP as a practical tool for understanding and optimizing LLM agent behavior.

Limitations and Future Work

AgentSHAP has several limitations that suggest directions for future research. First, it measures individual tool contributions without capturing synergies between tools. When

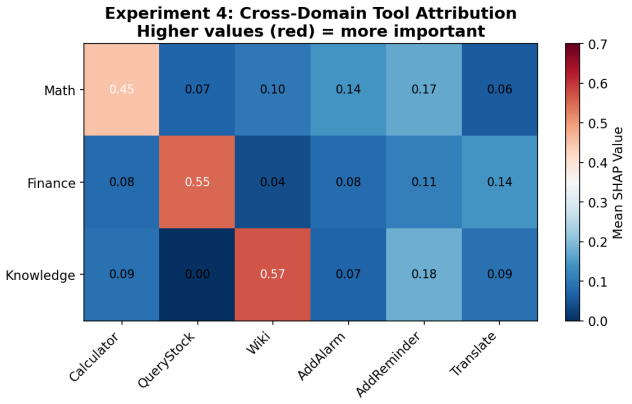

Figure 6: **Cross-domain attribution.** The heatmap shows mean SHAP values per domain-tool pair. Each domain correctly assigns highest importance to the expected tool: Calculator for Math (0.45), QueryStock for Finance (0.55), Wiki for Knowledge (0.57).

Calculator and Wiki together produce better results than either alone, this interaction effect is not directly measured. Extending to pairwise or higher-order Shapley interactions could address this. Second, AgentSHAP analyzes single prompt-response pairs rather than multi-turn conversations where tool importance may shift over time. Tracking importance across conversation turns would provide richer insights for dialogue agents. Third, when agents call multiple tools in sequence, AgentSHAP attributes to the tool set rather than the specific calling order. Incorporating causal analysis of tool call chains could reveal ordering effects.

The main computational cost is LLM API calls, approximately $n+\rho \cdot 2^n$ calls for n tools with sampling ratio ρ. For 8 tools with $\rho = 0.5$, this means about 130 calls. Lower sampling ratios still provide useful estimates for cost-sensitive applications, but real-time explanations during agent execution remain challenging.

Future work could also use SHAP patterns to automatically recommend adding or removing tools, integrate with agent training pipelines for tool-aware fine-tuning, and extend to multi-agent systems where multiple agents collaborate using shared tools.

Conclusion

We presented AgentSHAP, the first framework for explaining tool importance in LLM agents. Using Monte Carlo Shapley values, AgentSHAP computes fair, theoretically grounded importance scores for each tool. Our experiments on API-Bank demonstrate strong results: 0.945 consistency across runs, 13x quality difference when removing important versus unimportant tools, 7x SHAP gap between the expected tool and irrelevant tools, and 86-100% top-1 accuracy across experiments.

AgentSHAP enables practical applications including debugging agent errors, pruning unnecessary tools to reduce costs and latency, calibrating user trust, and A/B testing new tools. As LLM agents become ubiquitous, AgentSHAP pro-

vides the transparency needed to understand, debug, and trust their behavior.

References

[1] Castro, J.; Gómez, D.; and Tejada, J. 2009. Polynomial calculation of the Shapley value based on sampling. *Computers & Operations Research*, 36(5): 1726–1730.

[2] Goldshmidt, R. 2025. Attention, Please! PixelSHAP Reveals What Vision-Language Models Actually Focus On. *arXiv preprint arXiv:2503.06670*.

[3] Goldshmidt, R.; and Horovicz, M. 2024. TokenSHAP: Interpreting Large Language Models with Monte Carlo Shapley Value Estimation. *arXiv preprint arXiv:2407.10114*.

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

[5] Lundberg, S. M.; Erion, G.; Chen, H.; DeGrave, A.; Prutkin, J. M.; Nair, B.; Katz, R.; Himmelfarb, J.; Bansal, N.; and Lee, S.-I. 2020. From local explanations to global understanding with explainable AI for trees. *Nature machine intelligence*, 2(1): 56–67.

[6] Lundberg, S. M.; and Lee, S.-I. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

[7] Mialon, G.; Dessì, R.; Lomeli, M.; Nalmpantis, C.; Pasunuru, R.; Raileanu, R.; Rozière, B.; Schick, T.; Dwivedi-Yu, J.; Celikyilmaz, A.; et al. 2023. Augmented language models: a survey. *Transactions on Machine Learning Research*.

[8] Patil, S. G.; Zhang, T.; Wang, X.; and Gonzalez, J. E. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*.

[9] Qin, Y.; Liang, S.; Ye, Y.; Zhu, K.; Yan, L.; Lu, Y.; Lin, Y.; Cong, X.; Tang, X.; Qian, B.; et al. 2023. Tool learning with foundation models. *arXiv preprint arXiv:2304.08354*.

[10] 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*.

[11] Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1135–1144.

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

[13] Shapley, L. S. 1953. A value for n-person games. *Contributions to the Theory of Games*, 2(28): 307–317.

[14] Shen, Y.; Song, K.; Tan, X.; Li, D.; Lu, W.; and Zhuang, Y. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36.

[15] Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; and Yao, S. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36.

[16] Sundararajan, M.; Taly, A.; and Yan, Q. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*, 3319–3328. PMLR.

[17] Vig, J. 2019. Analyzing the structure of attention in a transformer language model. *arXiv preprint arXiv:1906.04284*.

[18] Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; et al. 2023. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*.

[19] Xi, Z.; Chen, W.; Guo, X.; He, W.; Ding, Y.; Hong, B.; Zhang, M.; Wang, J.; Jin, S.; Zhou, E.; et al. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.

[20] Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.