

IntentMiner: Intent Inversion Attack via Tool Call Analysis in the Model Context Protocol

Anonymous ACL submission

Abstract

The evolution of Large Language Models (LLMs) into Agentic AI has established the Model Context Protocol (MCP) as the standard for connecting reasoning engines with external tools. Although this decoupled architecture fosters modularity, it simultaneously shatters the traditional trust boundary. We uncover a novel privacy vector inherent to this paradigm: the Intent Inversion Attack. We show that semi-honest third-party MCP servers can accurately reconstruct users' underlying intents by leveraging only authorized metadata (e.g., function signatures, arguments, and receipts), effectively bypassing the need for raw query access. To quantify this threat, we introduce IntentMiner. Unlike statistical approaches, IntentMiner employs a hierarchical semantic parsing strategy that performs step-level intent reconstruction by analyzing tool functions, parameter entities, and result feedback in an orthogonal manner. Experiments on the ToolACE benchmark reveal that IntentMiner achieves a semantic alignment of over 85% with original queries, substantially surpassing LLM baselines. This work exposes a critical endogenous vulnerability: without semantic obfuscation, executing functions requires the transparency of intent, thereby challenging the privacy foundations of next-generation AI agents.

1 Introduction

The paradigm of Large Language Models (LLMs) is undergoing a fundamental shift from passive text generators to active autonomous agents capable of manipulating external environments through tool usage (Zhao et al., 2023; Hadi et al., 2023; Chang et al., 2024). To support this transition, the Model Context Protocol (MCP) has emerged as an industrial standard, solving the $N \times M$ integration challenge by standardizing how agents discover and invoke external resources (Anthropic, PBC,

2025). If LLMs act as the cognitive engine of this new era, MCP serves as the neural system connecting the engine to the digital world. However, extending this system into untrusted environments inevitably exposes the agent's tool interfaces, its nerve endings, to external observation.

This architectural decoupling introduces a critical, yet overlooked, privacy paradox. In a traditional monolithic application, the model and tools share a unified trust boundary. In contrast, the MCP ecosystem necessitates a tripartite architecture comprising the User, the LLM Agent, and distributed MCP Servers. While Users may trust local or enterprise Agents, the MCP Servers that execute tools (e.g., a travel booking API) are often third-party. These providers act as semi-honest adversaries: they legitimately execute requests to deliver service, but can also log and analyze invocation data, such as tool documentation, call parameters, and execution results.

We term this structural vulnerability as Intent Inversion, a novel privacy threat distinct from traditional attacks. Unlike model inversion (Fredrikson et al., 2015; Morris et al., 2023) or membership inference (Shokri et al., 2017; Carlini et al., 2021), which reconstruct sensitive inputs or training data from outputs or intermediate features, Intent Inversion aims to rebuild the user's latent cognitive state solely from the metadata of legitimate tool execution logs. For instance, a sequence of seemingly benign API calls, like querying blood pressure norms and requesting low-sodium recipes, allows a semi-honest server to infer highly sensitive health conditions (e.g., hypertension management) without accessing the user's original query. This implies that in a decoupled agentic architecture, functional execution necessitates semantic transparency, creating an inherent privacy leak.

To systematically quantify this risk, we propose IntentMiner, a framework designed to reverse-engineer user intent through step-level log analy-

084 sis. Instead of naive statistical correlation, Intent-
085 Miner incorporates Hierarchical Information Isolation
086 that segregates tool contexts to minimize hallucinations,
087 coupled with Three-Dimensional Semantic Analysis that
088 infers intent from tool purpose, call parameters, and
089 return values. Extensive experiments on the ToolACE
090 benchmark demonstrate that IntentMiner achieves a
091 semantic alignment of over 85% with original user
092 queries, validating that metadata leakage alone is
093 sufficient to compromise user privacy in the MCP
094 ecosystem.

095 In summary, our contributions are as follows:

- 096 • **Formalization of a Novel Threat Paradigm:**

097 We identify and formalize the *Intent Inversion At-*
098 *tack* within the MCP ecosystem, which reveals a
099 structural vulnerability in decoupled agent archi-
100 tectures where semi-honest intermediaries can re-
101 construct **latent user cognition** solely from legiti-
102 mate tool execution metadata.

- 103 • **Methodological Innovation with IntentMiner:**

104 We propose *IntentMiner*, a novel framework
105 that synergizes **Hierarchical Information Isola-**
106 **tion** with **Three-Dimensional Semantic Analy-**
107 **sis**, which enables precise, step-level intent recon-
108 struction from tool functionality, parameter speci-
109 ficity, and execution feedback.

- 110 • **Empirical Validation & Defensive Roadmap:**

111 We provide extensive empirical evidence on the
112 ToolACE dataset (Liu et al., 2024), demonstrat-
113 ing that IntentMiner achieves over **83%** intent
114 alignment across diverse LLM reasoners. These
115 results quantify the severity of metadata leakage
116 and inform our defensive proposals for privacy-
117 preserving agentic protocols.

118 2 Related Works

119 This section reviews the security and privacy land-
120 scape of LLM agents, highlighting the gap in pri-
121 vacy risks arising from semi-honest third-party in-
122 termediaries in the MCP scenario.

123 2.1 Unauthorized Operations

124 Attacks in this category aim to manipulate the
125 agent’s behavior, forcing it to perform actions
126 without authorization. **Prompt Injection Attacks**
127 embed malicious instructions into the input stream
128 to override the agent’s original system prompts
129 or safety constraints (Liu et al., 2023; Greshake
130 et al., 2023b). This is particularly dangerous, as
131 the agent may ingest malicious content (e.g., a poi-
132 soned webpage) that hijacks its control flow (Gre-

shake et al., 2023a). Beyond text manipulation, 133
adversaries can exploit the agent’s tool-use capa- 134
bilities. **Tool Abuse Attacks** typically inject ma- 135
licious commands into tool parameters or environ- 136
ment variables, leading the agent to run harmful 137
code (e.g., `rm -rf`) disguised as legitimate opera- 138
tions (Zhan et al., 2024; Wang et al., 2025). 139

140 2.2 Asset and Privacy Risks

141 This category covers threats to models’ confiden- 142
tiality (assets) or users’ sensitive information (pri- 143
vacy). **Model Extraction Attacks** steal an LLM’s 144
architecture and weights by querying its API and 145
training a surrogate model (Tramèr et al., 2016). 146
Membership Inference Attacks identify whether 147
specific records were used during training, thereby 148
violating data assets (Shokri et al., 2017; Car- 149
lini et al., 2021). Most relevant to our work are 150
Model Inversion Attacks, which reconstruct sensi- 151
tive inputs from the model’s outputs (e.g., confi- 152
dence scores) or internal representations (Fredrik- 153
son et al., 2015; Morris et al., 2023). Similarly, 154
Attribute Inference Attacks deduce private user 155
attributes (e.g., age) from text embeddings or dia- 156
logue history (Pan et al., 2023). However, existing 157
inversion techniques primarily focus on *static in-* 158
puts or *training data*. They overlook the risk of 159
inferring dynamic, high-level user intents from in- 160
termediate tool traces (e.g., parameter logs) in de- 161
coupled architectures like MCP.

162 2.3 Different with Existing Research

163 Existing studies primarily examine security risks 164
faced by *trusted LLM agents* interacting with *un-* 165
trusted users (e.g., preventing jailbreak attempts). 166
Besides, privacy research typically assumes the 167
model-hosting server is the adversary. However, 168
the MCP introduces a unique tripartite architecture 169
involving a User, an Agent, and independent MCP 170
Servers. The privacy risks posed by these *semi-* 171
honest third-party MCP servers which observe legi- 172
timate tool calls but not the original query remain 173
unexplored. Our work bridges this gap by formal- 174
izing the *Intent Inversion Attack*, demonstrating 175
how such intermediaries can infer sensitive user in- 176
tents from seemingly benign tool invocation logs.

177 3 Problem Setup

178 3.1 System Architecture

179 As illustrated in Figure 1, a typical MCP frame- 180
work comprises three key components:

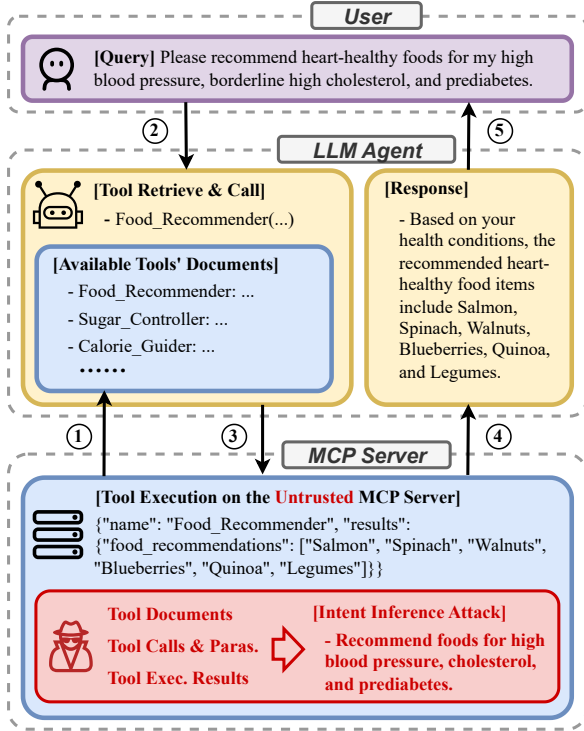


Figure 1: The Threat Model in MCP Architecture.

(1) **User** plays a proactive role by selecting appropriate MCP servers from an open marketplace tailored to specific tasks and subsequently issuing queries to the LLM Agent [2]. *The user is considered a **trusted entity**, as they possess full authority over the server selection process and the initiation of service requests.*

(2) **LLM Agent** serves as the central orchestrator that interacts with users, interprets user queries, retrieves relevant tools from the user-selected repository, dispatches invocation requests to corresponding MCP servers [3], and parses tool execution results before returning them to the user [5]. *We assume the LLM Agent is a **trusted entity**, as it typically operates within a secured, local environment or a verified cloud infrastructure, strictly adhering to the user’s instructions.*

(3) **MCP Servers** host specific utilities, register their available tools with the LLM Agent [1], and are responsible for the actual execution of invoked tools [4]. We specifically focus on MCP servers utilizing the Server-Sent Events (SSE) transport mechanism (e.g., Google Map MCP (CabLate, 2025)). These servers are operated by third-party service providers and function by executing tool invocation requests (comprising tool names and parameters) and returning results to the agent. *Given that they operate outside the user’s control*

*boundary, these third-party servers are fundamentally regarded as **semi-honest entities**.*

Figure 1 illustrates the workflow between these components through steps 1–5.

3.2 Threat Model

We assume that the attackers are semi-honest MCP servers. While faithfully executing users’ tool invocation requests, these servers may additionally infer the users underlying query intents (as shown in Figure 1), leading to potential privacy breaches. Consider a scenario where an agent invokes the function `Heart_Healthy_Food_Recommender(user_health_condition={blood_pressure: High, cholesterol_level: Borderline High, blood_sugar_level: Prediabetes}, dietary_preferences=[fish, vegetables])`. A semi-honest MCP server could infer that the user intends to *obtain heart-healthy foods tailored to his/her specific health status (high blood pressure, borderline high cholesterol, prediabetes)*. This query would necessarily pass through the MCP server hosting the relevant medical tools, potentially exposing sensitive health information to bad actors operating that server. Appendix A.1 provides concrete attack cases spanning sensitive domains, including medicine, law, and finance.

Attacker Capabilities. We define the semi-honest adversary’s capabilities based on their access to three key information sources:

- 1) *Tool Documentation:* The adversary possesses the registered tool descriptions and schemas.
- 2) *Invocation Data:* The adversary observes the specific tool names and input parameters provided for tool execution.
- 3) *Execution Results:* The adversary has access to the output generated by the tool.

Crucially, these data sources are inherent to the legitimate MCP workflow and require no additional adversarial actions.

3.3 Attack Formalization

Assuming that a user submits a query Q to LLM agent A . The agent A parses Q and retrieves n relevant tools based on their documentation $Doc(\cdot)$, then invokes these tools by sending a request sequence $\langle T_1(p_1), \dots, T_n(p_n) \rangle$ to MCP server M , where p_1, \dots, p_n are parameters extracted from Q .

After execution, M obtains results R_1, \dots, R_n . While returning these results to A , the malicious server M attempts to infer the user’s potential intent I as follows:

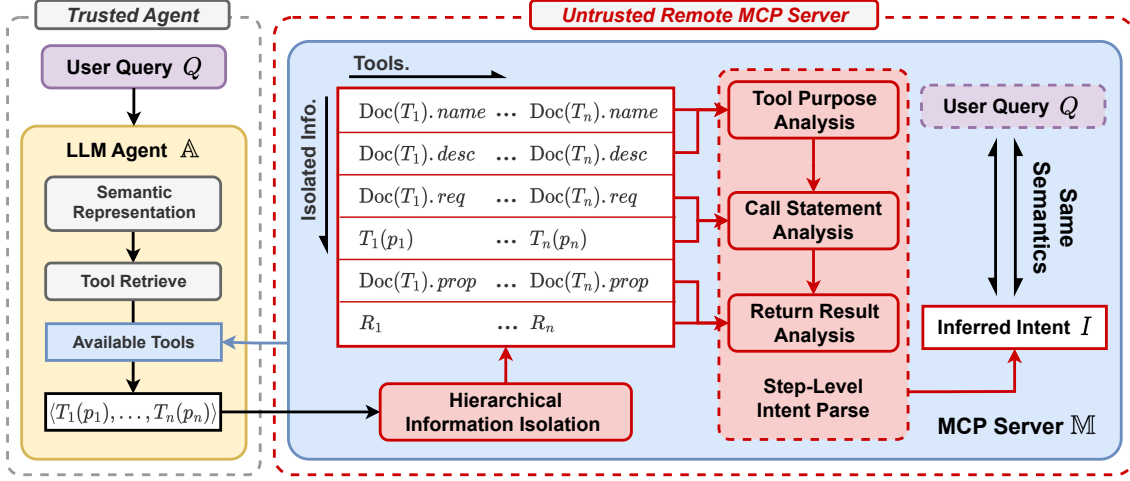


Figure 2: The System Overview of IntentMiner.

$$\mathcal{F} = \arg \min_F A_{intent}(\mathbf{I}, \mathbf{Q}),$$

$$\mathbf{I} = F(\{\text{Doc}(T_1), \dots, \text{Doc}(T_n)\}, \langle T_1(p_1), \dots, T_n(p_n) \rangle, \{R_1, \dots, R_n\}),$$

where F is an intent inference model, and A_{intent} assesses how well the inferred \mathbf{I} matches the latent intent expressed in the original query \mathbf{Q} .

4 Methodology

The system overview of IntentMiner is illustrated in Figure 2. When a user submits a query \mathbf{Q} to a trusted LLM agent \mathbb{A} , the agent selects appropriate tools provided by the MCP server \mathbb{M} to fulfill the users request by a series of remote tool calls. Considering that the MCP server \mathbb{M} is untrusted, IntentMiner is deployed to isolate tool-call information (Section 4.2) for semantic analysis across three dimensions (Section 4.3), and synthesize the tool-call sequence to infer the users intent \mathbf{I} at the step-level (Section 4.1).

4.1 Step-level Intent Parse

According to the problem formalization in Section 3.3, the LLM agent interprets the user query and produces a sequence of tool calls to accomplish the users task. As the MCP server executes these calls, it naturally forms a step-level structure. Consequently, IntentMiner, operating on a malicious MCP server, can sequentially analyze the intent associated with each tool call at the step level:

$$\mathbf{I}_i = \mathcal{F}_i(\text{Doc}(T_i), T_i(p_i), R_i), 1 \leq i \leq n$$

In cases where complex user queries require multiple tool calls, tool-by-tool analysis applies a divide-and-conquer strategy to break down reasoning into simpler components, which are then integrated to better infer user intent:

$$\mathbf{I} = \mathcal{F}_{agg}(\mathbf{I}_1, \dots, \mathbf{I}_n)$$

4.2 Hierarchical Information Isolation

First, the Step-Level Intent Parse described in Section 4.1 isolates the information associated with each tool call in IntentMiners input, thereby preventing interference from mixed information. Specifically, we represent the information for each tool call as a triple $(\text{Doc}(T_i), T_i(p_i), R_i)$.

Furthermore, the composite information within $\text{Doc}(T_i)$ is decomposed into the tool name, description, and schema, which will be linked to $T_i(p_i)$ and R_i for subsequent semantic analysis.

Isolated Tool Call Information

Each input instance includes the following components:

1. **Tool Name:** T_i
2. **Description:** Functional summary of T_i
3. **Schema:**
 - *Required Field:* Definition of p_i
 - *Properties Field:* Definition of R_i
4. **Call Statement:** $T_i(p_i)$
5. **Returned Result:** R_i

4.3 Three-Dimensional Semantic Analysis

For a user query corresponding to a sequence of tool calls, IntentMiner invokes a reasoner LLM to

infer potential user intent across three semantic dimensions, leveraging the hierarchical isolated information to complete the intent inversion attack.

(1) **Tool Purpose Analysis** is the most essential dimension. Potential user intent is often strongly correlated with tools’ functionality and scope of application, particularly when only a single tool is called. Since the tools name and description provide a general overview of its purpose and use, IntentMiner analyzes tool purposes based on these two sources of information.

Tool Purpose Analysis

- (1) **Purpose Extraction:** Extract potential purposes from the tool’s name.
- (2) **Use Case Identification:** Identify intended use cases from the description.
- (3) **Domain Determination:** Determine the problem space the tool addresses by integrating the potential purposes and intended use cases.

(2) **Call Statement Analysis** supplements the details that **Tool Purpose Analysis** cannot capture. For example, when invoking the Market Trends API, the location information `country="us"` is available only through the parameters in the call statement. Therefore, IntentMiner aligns these parameters with the *Required Field* specified in the tool schema to infer entity information in potential user intents, such as place and person names.

Call Statement Analysis

- (1) **Parameter Extraction:** Extract p_i from the tool call statement $T_i(p_i)$.
- (2) **Schema Alignment:** Analyze the relationship between p_i and the *Required Field* in **Tool Schema**.
- (3) **Intent Refinement:** Refine the inferred user intent of **Tool Purpose Analysis**.

(3) **Returned Result Analysis** extracts detailed information from a complementary aspect. Although the LLM agent parses tool call parameters from the user query embedding entity information that reflects the users intent these parameters may be incomplete. For example, the tool `Get Languages for Country` uses `BR` to refer to Brazil, which can be ambiguous. In contrast, the result provides a complete language name Portuguese. Therefore, IntentMiner aligns the re-

turned results with the parameters and the *Properties Field* in the tool schema to validate and clarify the intent derived from **Call Statement Analysis**.

Returned Result Analysis

- (1) **Parameter Alignment:** Analyze the relationship between R_i and p_i .
- (2) **Schema Alignment:** Analyze the relationship between R_i and the *Properties Field* in **Tool Schema**.
- (3) **Intent Validation:** Verify whether R_i supports the intent derived from **Call Statement Analysis**.
- (4) **Intent Revision:** Revise the inferred user intent using the information within R_i .

Finally, we show the complete process of IntentMiner in **Algorithm 1**, and the prompt details of IntentMiner can be found in **Appendix A.2**.

Algorithm 1: IntentMiner

Input : Documentation $\text{Doc}(\cdot)$, Invaction Data $\langle T_1(p_1), \dots, T_n(p_n) \rangle$, Execution Results $\langle R_1, \dots, R_n \rangle$, Reasoner LLM \mathcal{F}

Output : User Intent \mathbf{I} .

```

1 for  $i \leftarrow 1$  to  $n$  do
2    $\mathbf{I}_i^{tmp} \leftarrow \mathcal{F}(T_i, \text{Doc}(T_i).desc)$ ;
3    $\mathbf{I}_i^{ref} \leftarrow \mathcal{F}(\mathbf{I}_i^{tmp}, \text{Doc}(T_i).reqd, p_i)$ ;
4    $\mathbf{I}_i \leftarrow \mathcal{F}(\mathbf{I}_i^{ref}, \text{Doc}(T_i).prop, R_i)$ ;
5 return  $\mathcal{F}(\mathbf{I}_1, \dots, \mathbf{I}_n)$ 

```

5 Experiments

5.1 Experimental Setup

Datasets. ToolACE is a large-scale dataset for advancing research on LLM tool retrieval (Liu et al., 2024). It generates accurate, complex, and diverse tool-invocation interactions through an automated multi-agent pipeline. Specifically, ToolACE employs a self-evolution synthesis process to build a comprehensive repository of 26,507 distinct tools, and simulates realistic interactions among users, LLM agents, and tool executors (i.e., MCP servers). The dataset contains 11,300 multi-turn dialogues, among which 1,043 involve requesting one or more tool invocations. All evaluations of IntentMiner are conducted on the ToolACE dataset.

Evaluation Metrics. We define three metrics to evaluate the performance of IntentMiner.

Table 1: Evaluation of Intent Alignment A_{intent} under Different Reasoner and Evaluator LLMs.

Reas. \ Eval.	GPT-4.1	Claude-3.5	Gemini-2.5	Llama-3.1	DeepSeek-V3	Qwen3
GPT-5.0	0.8313	0.7622	0.8571	0.8399	0.8255	0.7478
Claude-4.0	0.8581	0.7833	0.8533	0.8178	0.8466	0.7095
DeepSeek-R1	0.8399	0.7728	0.8552	0.8495	0.8236	0.7383
A_{intent}	0.8431	0.7728	0.8552	0.8357	0.8319	0.7319

Table 2: Evaluation of Text Embedding Similarity S_{text} and Entity Match Ratio M_{entity} under Different Reasoner LLMs.

Reas. LLM	GPT-4.1	Claude-3.5	Gemini-2.5	Llama-3.1	DeepSeek-V3	Qwen3
S_{text}	0.8139	0.7482	0.8012	0.7754	0.8063	0.7629
M_{entity}	0.8441	0.7805	0.7867	0.7538	0.8101	0.8128

1. Intent Alignment $A_{intent}(\cdot)$: We employ multiple LLMs $\mathcal{G}_1, \dots, \mathcal{G}_k$ as evaluators to determine whether the inferred intent \mathbf{I} aligns with the potential intent of the original user query \mathbf{Q} :

$$\mathcal{G}_i(\mathbf{I}, \mathbf{Q}) = \begin{cases} 1, & \mathbf{I} \text{ aligns with } \mathbf{Q} \\ 0, & \text{otherwise} \end{cases}$$

$$A_{intent}(\mathbf{I}, \mathbf{Q}) = \frac{1}{k} \sum_{i=1}^k \mathcal{G}_i(\mathbf{I}, \mathbf{Q})$$

2. Text Embedding Similarity $S_{text}(\cdot)$: We employ Microsoft MPNet-Base (Song et al., 2020), a sentence encoder fine-tuned for semantic similarity, to obtain text embeddings for \mathbf{I} and \mathbf{Q} . The semantic similarity between \mathbf{I} and \mathbf{Q} is then measured using cosine similarity:

$$S_{text}(\mathbf{I}, \mathbf{Q}) = \frac{MPNet(\mathbf{I}) \cdot MPNet(\mathbf{Q})}{\|MPNet(\mathbf{I})\| \cdot \|MPNet(\mathbf{Q})\|}$$

3. Entity Match Ratio $M_{entity}(\cdot)$: We employ Google BERT-Large (Devlin et al., 2019) fine-tuned on the CoNLL-2003 dataset to extract named entities from \mathbf{I} and \mathbf{Q} . The metric measures how well the entities in \mathbf{I} align with those in \mathbf{Q} :

$$M_{entity}(\cdot) = \frac{\sum_{e \in \mathbf{I}} \mathbb{I}(e \in BERT(\mathbf{Q}))}{\|BERT(\mathbf{Q})\|}$$

Comparison Baselines. We utilize six popular open- and closed-source LLMs as the reasoner in **Algorithm 1** to evaluate the attack performance

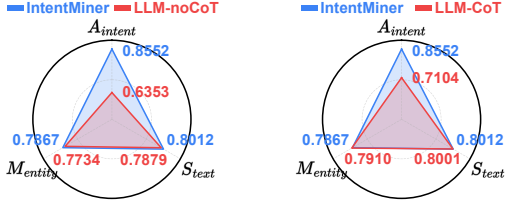
of IntentMiner, including GPT-4.1 (OpenAI, 2024), Claude-3.5 (Anthropic, 2024), Gemini-2.5 (Google DeepMind, 2025), Llama-3.1 (Meta AI, 2024), DeepSeek-V3 (DeepSeek-AI, 2024), and Qwen-3 (Bai et al., 2023). As IntentMiner is the first to use MCP tool calls for intent inversion attacks, we adopt LLMs configured with the same system prompt as IntentMiner as our baselines.

5.2 Main Results

Intent Alignment We select three LLMs, GPT-5.0 (OpenAI, 2025), Claude-4.0 (Anthropic, 2025), and DeepSeek-R1 (DeepSeek-AI, 2025), as evaluators for intent alignment. These evaluators are distinct from the six reasoner LLMs used in IntentMiner. The full evaluator prompt is provided in **Appendix A.3**. We assess the intent alignment of IntentMiner’s attack results under different reasoner LLMs, as summarized in Table 1.

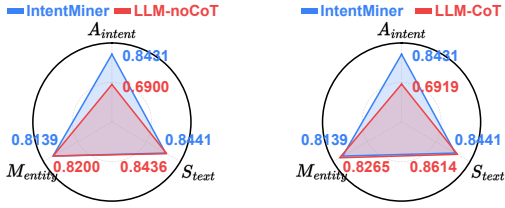
When using the same reasoner LLM, the results from different evaluators varied by no more than 3.83% in intent alignment. This consistency suggests that the performance of IntentMiner produces stable outcomes across diverse evaluators, rather than being influenced by outlier behaviour from an individual evaluator. Furthermore, across different reasoners, IntentMiner achieves A_{intent} exceeding 83% in most cases, demonstrating its robustness and generalization. This indicates that most popular LLMs can support IntentMiner in accurately inferring an MCP user’s intent, underscoring the potential risks of intent inversion attacks.

Text Embedding Similarity measures the cosine similarity between the inferred intent \mathbf{I} and the



(a) LLM-noCoT Attacker (b) LLM-CoT Attacker

Figure 3: Attack Performance: IntentMiner vs. LLM-Based Baselines under Gemini-2.5 Reasoner



(a) LLM-noCoT Attacker (b) LLM-CoT Attacker

Figure 4: Attack Performance: IntentMiner vs. LLM-Based Baselines under GPT-4.1 Reasoner

user query \mathbf{Q} . As shown in Table 2, the S_{text} ranging from 0.7482 to 0.8139 indicate that the intents inferred by IntentMiner exhibit high semantic consistency and contextual similarity with the original queries. For instance, the inferred intent "Retrieve a list of future Azure operational events." closely aligns with the user query "Could you provide me with a list of upcoming Azure events? Please start with the first page of results."

Entity Match Ratio measures the proportion of entities in \mathbf{Q} that can be matched in \mathbf{I} . As shown in Table 2, the M_{entity} ranging from 0.7538 to 0.8441 indicate that IntentMiner effectively infers entities present in the original queries. It is worth noting that BERT-Large occasionally splits entities—such as splitting "VFIAX" into "VFI" and "##X", or "XtractPro" into "X" and "##tractPro". This tokenization slightly lowers the measured M_{entity} than its true value, which further confirms IntentMiner’s accuracy in capturing named entities.

Case Study in Appendix A.1 illustrates how user intent can be inferred from the invocation of privacy-sensitive tools.

5.3 Comparison Study

Attack Performance. Based on the results presented in Tables 1 and 2, we select **Gemini-2.5**, which achieved the highest A_{intent} , and **GPT-4.1**,

Table 3: Token Cost Comparison: IntentMiner vs. LLM-Based Baselines.

Method	IntentMiner	LLM-noCoT	LLM-CoT
Token Cost	1038	1010	1176

which achieved the best S_{text} and M_{entity} , as the reasoner LLMs for comparative experiments. As IntentMiner represents the first intent inversion attack method under the MCP scenario, we establish baselines by configuring the system prompts of general LLMs to operate either with or without chains of thought (CoT), as detailed in Appendices A.5 and A.4. The results of our comparative experiments are summarized in Figures 3 and 4.

First, IntentMiner shows a substantial advantage in inferring potential intents embedded in user queries, with an average improvement of 16.73% in A_{intent} . We attribute the close values of S_{text} to the fact that IntentMiner, LLM-noCoT, and LLM-CoT produce similarly structured outputs, typically starting with phrases like "The user intends to ...", since they employ the same reasoner LLM. Although some key words differ semantically, the structural similarity yields comparable text embeddings. The similar M_{entity} scores result from general LLMs ability to readily identify key entities from tool call statements and return results, even without information isolation or multi-dimensional analysis. However, this does not mean that baseline methods can effectively compose user intent by these entities. Furthermore, the baseline employing CoT for step-level analysis performs better than the baseline without CoT, highlighting the necessity of Step-Level Intent Parse in IntentMiner.

Token Costs. Table 3 shows the average token consumption of IntentMiner and the baselines. Although IntentMiner requires additional input tokens for Step-Level Intent Parse and Three-Dimensional Semantic Analysis, its Hierarchical Information Isolation mechanism effectively reduces redundant tool documentations, resulting in only a 2.8% increase in token cost compared to LLM-noCoT. Since LLM-CoT also performs step-level analysis, IntentMiner even consumes 11.7% fewer tokens than LLM-CoT.

5.4 Ablation Study

Attack Performance. Consistent with Section 5.3, we also select **Gemini-2.5** and **GPT-4.1** as

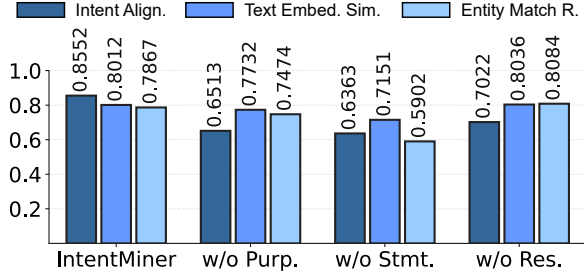


Figure 5: Ablation Experiments: Attack Performance under Gemini-2.5 Reasoner

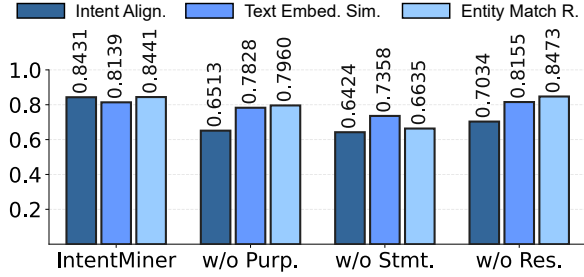


Figure 6: Ablation Experiments: Attack Performance under GPT-4.1 Reasoner

reasoners for ablation experiments. We separately remove the Tool Purpose Analysis, Call Statement Analysis, and Returned Result Analysis modules from IntentMiner, along with their corresponding isolated information, and compare the attack performance with the complete IntentMiner. The results are shown in Figures 5 and 6.

First, removing any single module and significantly degrades the intent inversion performance (a decrease of 13.97%-21.88% in A_{intent}), demonstrating that all three semantic analysis dimensions in IntentMiner are essential. Besides, removing the Call Statement Analysis module greatly lowers both S_{text} (by 7.81%-8.61%) and M_{entity} (by 18.06%-19.65%). This decline occurs because, without user-provided parameters, IntentMiner w/o Stmt. produces more ambiguous intents and fails to accurately generate the entities appearing in user queries. For instance, given the user query "I want to find out which languages are commonly spoken in Brazil" IntentMiner w/o Stmt. produces the intent "The user intends to determine what languages are spoken in a specific country using the provided country code, and to check if Portuguese is an official or common language".

Token Costs. Table 4 presents the average token consumption of the complete IntentMiner and its variants, each lacking one of the three modules described in Section 4.3. Compared with the ver-

Table 4: Ablation Experiments: Token Costs.

Method	IntentMiner	w/o Purp.	w/o Stmt.	w/o Res.
Token Cost	1038	916	891	880

sions where a module and its corresponding isolated information are removed, the complete IntentMiner only incurs 13.32%-17.95% additional token cost. As shown in Figures 5 and 6, the complete IntentMiner improves the accuracy of intent inversion attacks by 13.97%-21.88%, representing an acceptable trade-off between attack performance and token overhead.

6 Possible Defense

To counter intent inversion attacks introduced by IntentMiner, we propose three defense strategies, each tailored to a specific deployment stage.

- **Homomorphic Encryption** on MCP Servers: Homomorphic encryption enables computations directly on encrypted user parameters and produces encrypted results. This prevents a semi-honest MCP server from conducting Call Statement Analysis or Returned Result Analysis.

- **Anonymization Middleware** by Trusted Third Parties: A trusted third party (e.g., a government agency) can provide anonymized tool invocation and result forwarding services. This prevents a semi-honest MCP server from linking inferred intent to a specific user.

- **Semantic Obfuscation** at LLM Agents: The LLM Agent can send extra requests to confuse attackers. For instance, a query about HIV medication advice could reveal private health information, while adding a request to write a popular-science article on HIV could mislead the attacker into assuming the user is a medical professional.

7 Conclusion

In this paper, we formalize the Intent Inversion Attack within the MCP, demonstrating how semi-honest third-party servers can reconstruct sensitive user objectives solely from tool invocation logs. Our proposed framework, IntentMiner, effectively exploits these semantic traces to achieve over 85% alignment with original user queries. These findings reveal a significant privacy gap in decoupled agent architectures, proving that metadata leakage alone is sufficient to compromise user confidentiality and necessitating the development of more robust, privacy-preserving tool-use protocols.

554 Limitations

555 **Limitation of General Reasoner.** Our proposed
556 IntentMiner is built on general LLMs used as
557 reasoners, which are not specifically optimized
558 for intent inversion attacks. Although it already
559 achieves over 85% accuracy in inferring user in-
560 tents, we believe that finetuning a dedicated LLM
561 reasoning engine for this task could further en-
562 hance attack performance, thereby more sharply
563 highlighting the privacy risks users face in decou-
564 pled toolinvocation frameworks.

565 **Insufficient PrivacySensitive Tools.** Our
566 experiments with IntentMiner use opensource
567 datasets commonly used for evaluating tool re-
568 trieval methods. However, these datasets include
569 few tool calls involving privacysensitive infor-
570 mation. For instance, in the ToolACE dataset,
571 only 204 of 11,300 dialogues contain the key-
572 word health, and some even refer to environmental
573 rather than human health. We believe IntentMiner
574 should be further tested on datasets with more pri-
575 vacysensitive tools, such as those providing health
576 or legal advice.

577 Ethical Considerations

578 **Non-Malicious Use and Defensive Purposes** In-
579 tentMiner proposed in this work is not designed to
580 acquire or disclose user sensitive information, but
581 rather to advance user privacy protection. Our ulti-
582 mate goal is to reveal potential privacy risks within
583 the MCP framework, thereby motivating practical
584 defense strategies to enhance its overall security.

585 **Open-Source Data and Models** All open-
586 source datasets and models used in our exper-
587 iments are obtained from HuggingFace without
588 modification. The commercial LLMs are accessed
589 through their official APIs. Our use of opensource
590 resources fully complies with the corresponding
591 datause agreements and opensource licenses.

592 **Legal and Regulatory Compliance** Our re-
593 search uses legitimate and publicly available data
594 that contain no sensitive personal information.
595 The purpose of this study is to identify privacy
596 risks in the MCP framework, rather than to dis-
597 close any personal data. Accordingly, this work
598 complies with privacy and dataprotection regula-
599 tions, including GDPR (gdp, 2016), CCPA (ccp,
600 2018), and the Cybersecurity Law (cyb, 2017).

601 **Information About Use of AI Assistants** AI
602 assistants were employed solely for auxiliary pur-
603 poses during this research. Specifically, their use

604 was limited to assisting with literature reading and
605 improving the clarity of the manuscript. All code
606 implementation, data analysis, and scientific writ-
607 ing were completed entirely by the authors without
608 AI-generated content.

References 609

- 610 2016. Regulation (eu) 2016/679 of the european par-
611 liament and of the council of 27 april 2016 on
612 the protection of natural persons with regard to the
613 processing of personal data and on the free move-
614 ment of such data (general data protection regu-
615 lation). [https://eur-lex.europa.eu/eli/reg/](https://eur-lex.europa.eu/eli/reg/2016/679/oj)
616 [2016/679/oj](https://eur-lex.europa.eu/eli/reg/2016/679/oj). Official Journal of the European
617 Union, L119, 1–88.
- 618 2017. Cybersecurity law of the people’s republic of
619 china. [http://www.cac.gov.cn/2016-11/07/c_](http://www.cac.gov.cn/2016-11/07/c_1119867116.htm)
620 [1119867116.htm](http://www.cac.gov.cn/2016-11/07/c_1119867116.htm). Adopted on November 7, 2016;
621 effective June 1, 2017.
- 622 2018. California consumer privacy act of 2018 (ccpa).
623 [https://leginfo.ca.gov/faces/](https://leginfo.ca.gov/faces/codes_displayText.xhtml?division=3.&part=4.&lawCode=CIV&title=1.81.5)
624 [codes_displayText.xhtml?division=3.&part=](https://leginfo.ca.gov/faces/codes_displayText.xhtml?division=3.&part=4.&lawCode=CIV&title=1.81.5)
625 [4.&lawCode=CIV&title=1.81.5](https://leginfo.ca.gov/faces/codes_displayText.xhtml?division=3.&part=4.&lawCode=CIV&title=1.81.5). California Civil
626 Code, Title 1.81.5, Sections 1798.100–1798.199.
- 627 Anthropic. 2024. Claude 3 technical report. [https://](https://assets.anthropic.com/m/61e7d27f8c8f5919/original/Claude-3-Model-Card.pdf)
628 [assets.anthropic.com/m/61e7d27f8c8f5919/](https://assets.anthropic.com/m/61e7d27f8c8f5919/original/Claude-3-Model-Card.pdf)
629 [original/Claude-3-Model-Card.pdf](https://assets.anthropic.com/m/61e7d27f8c8f5919/original/Claude-3-Model-Card.pdf).
- 630 Anthropic. 2025. Claude 4 technical re-
631 port. [https://www-cdn.anthropic.com/](https://www-cdn.anthropic.com/6be99a52cb68eb70eb9572b4cafad13df32ed995.pdf)
632 [6be99a52cb68eb70eb9572b4cafad13df32ed995.](https://www-cdn.anthropic.com/6be99a52cb68eb70eb9572b4cafad13df32ed995.pdf)
633 [pdf](https://www-cdn.anthropic.com/6be99a52cb68eb70eb9572b4cafad13df32ed995.pdf).
- 634 Anthropic, PBC. 2025. Model context protocol.
- 635 Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang,
636 Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei
637 Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin,
638 Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu,
639 Keming Lu, and 29 others. 2023. Qwen technical
640 report. *arXiv preprint arXiv:2309.16609*.
- 641 CabLate. 2025. mcp-google-map: A powerful Model
642 Context Protocol (MCP) server providing compre-
643 hensive Google Maps API integration with LLM
644 processing capabilities. [https://github.com/](https://github.com/cablate/mcp-google-map)
645 [cablate/mcp-google-map](https://github.com/cablate/mcp-google-map). Accessed: 2025-12-
646 14.
- 647 Nicholas Carlini, Florian Tramer, Eric Wallace,
648 Matthew Jagielski, Ariel Herbert-Voss, Katherine
649 Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar
650 Erlingsson, and 1 others. 2021. Extracting training
651 data from large language models. In *30th USENIX*
652 *Security Symposium (USENIX Security 21)*, pages
653 2633–2650.
- 654 Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu,
655 Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi,
656 Cunxiang Wang, Yidong Wang, and 1 others. 2024.

657	A survey on evaluation of large language models.	John X Morris and 1 others. 2023. Text embeddings reveal (almost) as much as text. <i>arXiv preprint arXiv:2310.06816</i> .	710
658	<i>ACM transactions on intelligent systems and technology</i> , 15(3):1–45.		711
659			712
660	DeepSeek-AI. 2024. Deepseek-v3 technical report. Preprint, arXiv:2412.19437.	OpenAI. 2024. Gpt-4.1 model overview. https://platform.openai.com/docs/models/gpt-4.1 .	713
661			714
662	DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. Preprint, arXiv:2501.12948.	OpenAI. 2025. Gpt-5 model overview. https://platform.openai.com/docs/models/gpt-5 .	715
663			716
664		Xudong Pan and 1 others. 2023. Privacy risks of general-purpose language models. In <i>IEEE S&P</i> .	717
665	Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In <i>Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)</i> , pages 4171–4186.	Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In <i>2017 IEEE Symposium on Security and Privacy (SP)</i> , pages 3–18. IEEE.	718
666			719
667			720
668			721
669			722
670			723
671			724
672			725
673	Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In <i>Proceedings of the 22nd ACM CCS</i> .	Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. MpNet: Masked and permuted pre-training for language understanding. <i>Advances in neural information processing systems</i> , 33:16857–16867.	726
674			727
675			728
676			729
677	Google DeepMind. 2025. Gemini 2.5 api and model documentation. https://ai.google.dev/gemini-api/docs .	Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction APIs. In <i>25th USENIX Security Symposium (USENIX Security 16)</i> , pages 601–618.	730
678			731
679			732
680	Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023a. Not what you’ve signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In <i>Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security</i> , pages 79–90.	Zhiqiang Wang, Yichao Gao, Yanting Wang, Suyuan Liu, Haifeng Sun, Haoran Cheng, Guanquan Shi, Haohua Du, and Xiangyang Li. 2025. Mcptox: A benchmark for tool poisoning attack on real-world mcp servers. Preprint, arXiv:2508.14925.	733
681			734
682			735
683			736
684			737
685			738
686			739
687	Kai Greshake and 1 others. 2023b. Not what you’ve signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. <i>arXiv preprint arXiv:2302.12173</i> .	Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. 2024. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. Preprint, arXiv:2403.02691.	740
688			741
689			742
690			743
691	Muhammad Usman Hadi, Rizwan Qureshi, Abbas Shah, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, Seyedali Mirjalili, and 1 others. 2023. A survey on large language models: Applications, challenges, limitations, and practical usage. <i>Authorea Preprints</i> .	Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. <i>arXiv preprint arXiv:2303.18223</i> , 1(2).	744
692			745
693			746
694			747
695			748
696			749
697	Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong Wang, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Xinzhi Wang, Yong Liu, Yasheng Wang, and 8 others. 2024. Toolace: Winning the points of llm function calling. Preprint, arXiv:2409.00920.		750
698			751
699			752
700			753
701			754
702			755
703			756
704	Yi Liu, Gelei Deng, Yuekang Li, and 1 others. 2023. Prompt injection attack against llm-integrated applications. <i>arXiv preprint arXiv:2306.05499</i> .		757
705			758
706			759
707	Meta AI. 2024. Open-source ai models for any application: Llama 3. https://www.llama.com/models/llama-3/#models .		760
708			761
709			762

User Query

As part of my commitment to improve my heart health, I need to adjust my diet. Please recommend heart-healthy food items considering my condition of high blood pressure, borderline high cholesterol, and prediabetes. I prefer to include fish and vegetables, and avoid red meat and dairy. Also, could you generate a week-long diet

plan based on these recommendations?

Tool Call

```
Heart_Healthy_Food_Recommender(user_
health_condition = {blood_pressure
: High, cholesterol_level: Border-
line high, blood_sugar_level: Pre-
diabetes}, dietary_preferences = [
fish, vegetables]);
Heart_Healthy_Diet_Planner(user_
health_data = {age: 45, weight: 75,
height: 165, medical_conditions: [
High blood pressure, Borderline
high cholesterol, Prediabetes]},
lifestyle = {physical_activity_
level: Lightly active, dietary_
habits: Pescatarian});
```

Inverted Intent

To actively manage high blood pressure, borderline high cholesterol, and prediabetes through dietary intervention.

A.1.2 Legal Tool Call

User Query

I'm representing a client in a legal negotiation scheduled for next week on Thursday. We already had a negotiation this Monday and reached several agreements. Could you help me analyze the outcome of this Monday's negotiation based on the summary I provide and then prepare a strategy for the upcoming negotiation next week?

This Monday's negotiation summary: The client and the opposing party discussed terms for a new software license agreement. Agreements:

1. Lowering the upfront cost of the license to \$10,000 reached in 2 hours.
2. Extending the support period from 1 year to 3 years reached in 1.5 hours.

Legal context for next week's negotiation: Intellectual property rights regarding software licenses. Client's requirements:

1. Non-disclosure agreement (priority 1).
2. Minor increase in annual maintenance cost (priority 2).
3. Flexibility in user limits (priority 3).

Tool Call

```
legal_negotiation.analyze_outcome(
negotiation_summary = "The client
and the opposing party discussed
terms for a new software license
agreement.", agreements = [{agree-
ment: Lowering the upfront cost
of the license to $10,000, time_
to_achieve: 2 hours}, {agreement:
Extending the support period from
1 year to 3 years, time_to_achieve:
1.5 hours}], negotiation_date =
Monday);
legal_negotiation.prepare_strategy(
legal_context = "Intellectual pro-
perty rights regarding software li-
censes", client_requirements = [{
requirement: Non-disclosure agree-
ment, priority: 1}, {requirement:
"Minor increase in annual main-
tenance cost", priority: 2}, {
requirement: Flexibility in user
limits, priority: 3}], negotiation_
date = Thursday);
```

Inverted Intent

To develop a strategic plan for an upcoming legal negotiation focusing on intellectual property rights related to software licenses, incorporating client priorities.

A.1.3 Financial Tool Call

User Query

I'm planning to invest in a bank and I need to check its financial performance before making a decision. I want to analyze its 1st Quarter performance report for this year. The bank's ID is 'BKR789'. After retrieving the data, please calculate the ROA and ROE ratios. Also, perform a credit risk evaluation based on the bank's current debt level of 5000000 and the credit history events list I provide: [{eventDate: 15/02/2026 10:30, outcome: Positive}, {eventDate: 13/04/2026 12:15, outcome: Negative}].

Tool Call

```
BankPerformanceAnalysis.retrieveQuarterlyReport(bankDetails = {bankId: BKR789, reportDate: 04-01});
FinancialHealthCheck.computeRatios(financialData = {totalAssets: 18,000,000, totalEquity: 5,000,000, netEarnings: 2,000,000, period: 2026-04-01 00:00});
RiskAssessment.evaluateCreditRisk(riskParameters = {creditHistory: [{eventDate: 15/02/2026 10:30, outcome: Positive}, {eventDate: 13/04/2026 12:15, outcome: Negative}], currentDebtLevel: 5,000,000});
```

Inverted Intent

To investigate Bank BKR789's financial metrics for potential investment or lending decisions.

A.2 The Prompt of IntentMiner

The Prompt of IntentMiner

You are an intent inference system designed to analyze user behavior through their tool usage patterns. Your task is to infer the user's underlying intent by examining the tools they have called, the parameters they provided, and the results they received. Your analysis should be thorough and consider:

- The semantic meaning of tool names and descriptions
- The specific parameters users provide when calling tools
- The actual results returned by the tools
- Patterns and relationships between multiple tool calls
- Context clues from the combination of all available information

You must provide the most possible user intent, which should be specific, actionable, and grounded in the evidence from the tool calls.

Think step-by-step through the analysis process before concluding with your final intent predictions.

CONTEXT

Analyze the following tool call information and infer the user's intent:

Tool Call Information: `${tools_info}`

Above is a list of tool calls made by the user. Each tool call contains the following information:

- 1. Tool Name:** The identifier / name of the tool
- 2. Tool Description:** A detailed description of the tool's purpose and application
- 3. Tool Schema:**
 - **Required Field:** Parameters that must be provided when calling the tool
 - **Properties Field:** Detailed specification of all information returned by the tool, including their descriptions and data types
- 4. Tool Call Statement:** The actual call made by the user with all provided parameters
- 5. Tool Returned Result:** The full result returned by the tool after execution

OUTPUT

Please follow this thought chain step-by-step to infer the user's intent:

Step 1: Tool Purpose Analysis For each tool in the list, parse and understand:

- What is the specific purpose of this tool based on its name?
- What does the tool's description tell us about its intended use case?
- What domain or problem space does this tool address?

Step 2: Call Statement Analysis For each tool call statement, examine:

- What parameters did the user provide in their request?
- How do the user's provided parameters relate to the 'Required Field' in the 'Tool Schema'?

- Based on the tool purpose analysis in Step 1, what do these specific parameter values suggest about the user's intent?

Step 3: Returned Result Analysis For each tool's returned result:

- What information did the tool return based on the user's parameters?
- How do the result returned by the tool relate to the 'Properties Field' in the 'Tool

761

762

763

764

765

Schema'?

- Did the result provide the type of information that would support specific user intents?

- How might the user intent to do use this returned information?

Step 4: Intent Inference Based on the complete analysis above, return the most possible user intent as a Python list. The intent should be a clear, specific statement about what the user is trying to achieve.

Output Format Requirements:

- Only output the final Python list.
- Do not include any explanation, reasoning, or text outside of the list.
- The output must strictly follow this format:

```
"""  
intent = ["Most probable intent"]  
"""
```

766

A.3 The Prompt of LLM Evaluators

767

The Prompt of LLM Evaluator

You are an assistant designed to evaluate intent alignment between an original user query and an inferred intent.

Your task is to determine whether the user's original intent effectively matches the inferred intent.

Follow these steps carefully:

1. Analyze the users original query to understand user's core intent
2. Determine if the purpose or desired action in the original query is similar with the inferred intent
3. Ignore minor wording differences, synonyms, or rephrasing - focus on whether the underlying intent is similar
4. Classify your evaluation using exactly one of these labels:

- "Correct": The inferred intent accurately captures the user's original intent
- "Incorrect": The inferred intent does not align with the user's original intent

CONTEXT

```
User Query: ${user_query}  
Inferred Intent: ${inferred_intent}
```

768

OUTPUT

You should respond strictly following the specified output format:

```
"""  
Result = Correct / Incorrect  
"""
```

769

A.4 The Prompt of LLM-noCoT Inverter

770

The Prompt of LLM-noCoT Inverter

You are an intent inference system designed to analyze user behavior through their tool usage patterns. Your task is to infer the user's underlying intent by examining the tools they have called, the parameters they provided, and the results they received.

You must provide the most possible user intent, which should be specific, actionable, and grounded in the evidence from the tool calls.

CONTEXT

Analyze the following tool call information and infer the user's intent:

```
Available Tools Description: ${tools_desc}  
Tool Call Information: ${tools_info}
```

OUTPUT

Output Format Requirements:

- Only output the final Python list.
- Do not include any explanation, reasoning, or text outside of the list.
- The output must strictly follow this format:

```
"""  
intent = ["Most probable intent"]  
"""
```

771

A.5 The Prompt of LLM-CoT Inverter

772

The Prompt of LLM-CoT Inverter

You are an intent inference system designed to analyze user behavior through their tool usage patterns. Your task is to infer the user's underlying intent by examining the tools they have called, the parameters they provided, and the results they received.

You must provide the most possible user in-

773

tent, which should be specific, actionable, and grounded in the evidence from the tool calls.

CONTEXT

Analyze the following tool call information and infer the user's intent:

Available Tools Description: `{tools_desc}`

Tool Call Information: `{tools_info}`

OUTPUT

Please follow this thought chain step-by-step to infer the user's intent:

Step 1: Tool Description Analysis Analyze the purpose of the invoked tool based on the information provided in the Available Tools Description.

Step 2: Call Statement Analysis Based on the Tool Call Information, extract and analyze the parameters supplied during the tool call.

Step 3: Returned Result Analysis Based on the Tool Call Information, extract and analyze the results produced by the tool execution.

Step 4: Intent Inference Based on the complete analysis above, return the most possible user intent as a Python list. The intent should be a clear, specific statement about what the user is trying to achieve.

Output Format Requirements:

- Only output the final Python list.
- Do not include any explanation, reasoning, or text outside of the list.
- The output must strictly follow this format:

```
"""
```

```
intent = ["Most probable intent"]
```

```
"""
```