
AgentVocab: Structure-Aware Vocabulary Adaptation for Efficient LLM Agents

Anonymous Authors¹

Abstract

Recent large language models (LLMs) have demonstrated strong capabilities across challenging tasks, enabling their widespread adoption in agentic systems that interact with external tools. In such deployments, however, LLMs are typically trained with general-purpose tokenizers designed for broad language coverage, while their usage is dominated by narrow, structured tool-calling interactions. This training–deployment mismatch leads to inefficient tokenization, where repetitive structural patterns and frequent semantic units in function calls are fragmented into long sequences of low-level tokens, increasing decoding overhead. To address this gap, we introduce **AgentVocab**, a structure-aware vocabulary adaptation framework for efficient LLM agents. AgentVocab derives specialized vocabulary entries from real tool-calling traces and adapts the model vocabulary to better reflect structural and semantic regularities, without task-specific schema engineering. Experiments on τ and τ^2 -bench show that AgentVocab significantly improves decoding efficiency, reducing latency by approximately 15-25% relative to the vanilla baseline, while preserving tool-calling performance. Our approach is orthogonal to existing fine-tuning and agent-training methods and integrates seamlessly into standard agent pipelines. Source code and models will be available at <https://anonymous.4open.science/r/AgentVocab-28CC>.

1. Introduction

Large language models (LLMs) have made rapid progress in recent years, demonstrating strong performance across a wide range of tasks, such as multi-step mathematical rea-

¹Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

soning (Shao et al., 2025), program synthesis (Zeng et al., 2025; Hui et al., 2024; Anthropic, 2025), and instruction-driven problem solving (OpenAI, 2025; Yang et al., 2025). These advances go beyond traditional conversational capabilities and reflect a shift toward LLMs that can reliably follow complex instructions and operate under structured constraints. As a result, LLMs are increasingly integrated into downstream systems, where they serve not merely as dialogue-based chatbots, but as core computational components that support more sophisticated forms of interaction and control (Mon-Williams et al., 2025; Salimpour et al., 2025; Li et al., 2025; Xu et al., 2025a).

One prominent paradigm enabled by these advances is agentic AI, in which an LLM interacts with external tools, APIs, or environments to accomplish user-defined goals (Li et al., 2024; Yue et al., 2025). In such systems, LLMs are commonly responsible for selecting tools and producing structured outputs, such as function calls or JSON-formatted arguments. Importantly, although LLMs are capable of handling diverse language tasks, their deployment in agent systems is often highly specialized: once embedded in an agent pipeline, a model is typically used repeatedly for a narrow class of structured tool-calling interactions, which dominate the model’s usage profile (Patil et al., 2025; Qin et al., 2024).

Despite the growing body of work on agentic systems, we argue that a fundamental training–deployment mismatch remains underexplored. LLMs are trained with tokenizers and vocabularies designed for broad-coverage natural language, yet agentic tool calling is dominated by highly structured, repetitive patterns, including both JSON schemas and frequent semantic elements like common argument values. Existing approaches to improving tool-calling performance primarily focus on parameter adaptation or agent-level reasoning. For example, Gorilla (Patil et al., 2024) improves API invocation by training on instruction data derived from API documentation with retrieval-aware techniques. Recent data-centric pipelines such as APIGen (Liu et al., 2024b) and Magnet (Yin et al., 2025) scale tool-use learning by synthesizing agent trajectories with validation and distillation. In parallel, agent-oriented methods such as Re-ReST (Dou et al., 2024) and CodeAct (Wang et al., 2024) enhance tool use via reflection, trajectory refinement, or executable action representations. However, these methods largely assume a

fixed, general-purpose tokenizer and overlook the impact of vocabulary design on efficiency and reliability in structured tool-calling scenarios.

In this work, we revisit agentic tool calling from a vocabulary-level perspective. Our key insight is that, in agent deployments, LLMs assume a more specialized role than in general-purpose use, which creates an opportunity to adapt the vocabulary to structured interaction patterns that dominate tool calling. In such settings, frequent structural fragments and recurring semantic units in tool schemas and function calls can be treated as meaningful atomic units. Based on this observation, we propose **AgentVocab**, a data-driven approach that automatically mines high-frequency structural and semantic patterns from real tool-calling traces and incorporates them as specialized tokens into the model vocabulary. This design reduces token fragmentation, shortens effective sequence length, and provides clearer cues for attention. Our approach is orthogonal to existing fine-tuning and agent-training methods, leverages standardized protocols (e.g., JSON Schema) rather than task-specific schema engineering, and can be seamlessly integrated into standard training pipelines. Empirically, we show that vocabulary adaptation leads to consistent improvements in efficiency for tool-calling scenarios, while preserving task performance. Our main contributions are as follows:

- We propose AgentVocab, a data-driven approach that adapts the vocabulary to repetitive structured and semantic patterns of agentic tool calling.
- We empirically quantify the efficiency benefits of vocabulary adaptation, showing consistent reductions in generated tokens and decoding latency without sacrificing tool-calling performance.
- We provide an end-to-end pipeline for vocabulary refinement, which can be applied as a lightweight pre-processing and training step alongside existing agent methods.
- We will release the trained models and training scripts to facilitate reproducibility and future research.

2. Related Work

Agentic Tool Calling Recent work on agentic LLMs has focused primarily on improving tool-calling capability through data construction, parameter adaptation, and system-level design. A prominent line of research enhances tool usage by scaling supervision or fine-tuning on tool-oriented data. For example, Toolformer (Schick et al., 2023) learns when and how to invoke tools by automatically identifying tool-call positions in raw text and training with self-supervised execution feedback. Subsequent work such as ToolLLM (Qin et al., 2024) constructs large-scale

API-centric datasets based on ToolBench, a curated collection of real-world APIs and evaluation protocols, and uses them for supervised training of tool-using language models. Data-centric pipelines such as APIGen (Liu et al., 2024b) and APIGen-MT (Prabhakar et al., 2025) further synthesize function-calling examples and validate them through execution, enabling scalable and verifiable supervision for tool learning. Gorilla (Patil et al., 2024) improves API invocation by training on instruction data derived from API documentation with retrieval-aware techniques.

Another line of work improves agent tool use through explicit reasoning and interaction structures. Several methods introduce reasoning traces, reflection, or self-improvement loops to guide tool invocation decisions (Aksitov et al., 2024; Dou et al., 2024), while CodeAct (Wang et al., 2024) grounds actions in executable code to learn from concrete execution outcomes. Complementary benchmarks, including BFCL (Patil et al., 2025) and ToolSandbox (Lu et al., 2025), standardize the evaluation of function calling and stateful tool interactions. Across these approaches, improvements are achieved through training data, reasoning design, or system architecture, while the tokenizer and vocabulary are typically treated as fixed components.

Tokenization and Vocabulary Adaptation Tokenization and vocabulary design constitute a foundational interface between text and model computation, yet they are often kept fixed during post-training and downstream deployment unless explicitly adapted. Recent work has revisited this assumption by adapting vocabularies to better match domain-specific or structured data distributions. Herold et al. (2025) shows that extending a pretrained tokenizer with frequent in-domain tokens can significantly reduce sequence length and improve inference efficiency without degrading accuracy. AdaptiVocab (Nakash et al., 2025) similarly replaces parts of a general-purpose vocabulary with domain-specific tokens to compress structured text and accelerate generation in focused domains. VEGAD (Liu et al., 2024a) proposes a gradient-based criterion to select new vocabulary entries that most impact model loss, yielding improved representation and efficiency on technical corpora. HyperOFA (Özeren et al., 2025) further demonstrates that expanding vocabularies for new domains or languages can reduce tokenization fragmentation and improve downstream efficiency.

Beyond domain adaptation, several studies explicitly incorporate structural information into the vocabulary or tokenization process. Grammar-based representations introduce grammar production rules as tokens to enforce syntactic structure in code generation (Liang et al., 2025), while structure-aware tokenization schemes design dedicated tokens for table schemas, categorical fields, or structural delimiters in logs and tabular data (Karim et al., 2025). Related efforts treat discrete identifiers as first-class vocabulary ele-

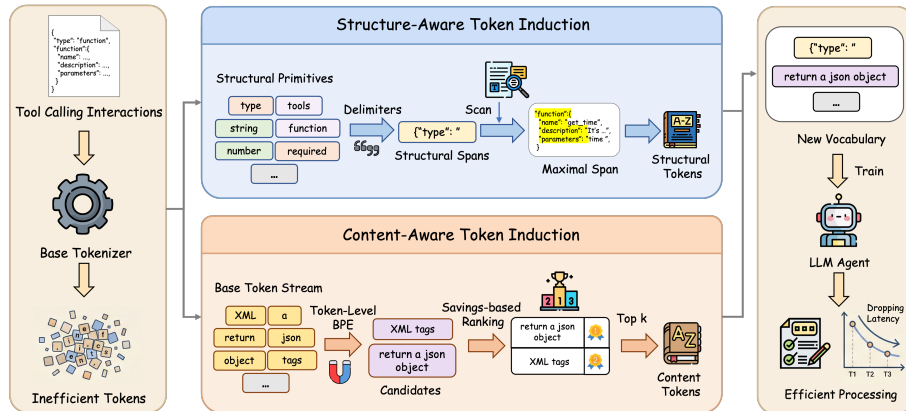


Figure 1. Overview of the AgentVocab framework. (Left) Base tokenizers fragment structured tool calls into inefficient, long sequences. (Middle) AgentVocab addresses this by adapting the vocabulary through a dual-branch strategy: **Structure-Aware Token Induction** extracts maximal structural spans based on defined structural primitives, while **Content-Aware Token Induction** identifies frequent semantic arguments. (Right) The resulting specialized vocabulary enables the LLM agent to generate tool calls with significantly reduced sequence lengths and lower decoding latency, enhancing both efficiency and stability.

ments, adding out-of-vocabulary tokens for users or items to better model structured relationships in recommendation systems (Huang et al., 2025).

Complementary to vocabulary modification, efficiency-oriented studies demonstrate that tokenization choices alone can lead to large differences in decoding speed and effective context length for structured inputs such as code or logs (Dagan et al., 2024; Gee et al., 2022). Representation-level compression methods, such as gist tokens, further explore efficiency by introducing auxiliary tokens to summarize context, operating at a different level than vocabulary adaptation (Mu et al., 2023). Despite these advances, existing work largely focuses on domain-specific modeling or structured generation in agentic tool calling, where structured schemas, APIs, and function signatures dominate both inputs and outputs, remains largely unexplored.

3. Method

3.1. Problem Setup

We consider large language models deployed as agents that interact with external tools through structured function calls. In such deployments, model inputs and outputs are drawn from a distribution $\mathcal{D}_{\text{tool}}$ of agentic tool-calling trajectories, which combine natural language context with serialized structured components such as schemas, function signatures, and arguments (e.g., JSON-formatted calls). Compared to general-purpose language modeling, $\mathcal{D}_{\text{tool}}$ exhibits a highly skewed token distribution: repetitive structural patterns and frequent semantic elements like tool descriptions recur persistently across interactions, while unconstrained free-form natural language content is comparatively limited.

Despite this shift in usage, agent models typically rely on fixed, general-purpose tokenizers that are not adapted to the structured distributions encountered during agent deployment. These tokenizers are optimized for broad linguistic coverage and tend to fragment structured strings into long sequences of low-level symbols and subwords. Consequently, structural spans that are semantically and syntactically coherent at the agent level are redundantly tokenized, leading to inflated sequence lengths and increased decoding cost during inference.

We address this mismatch through vocabulary adaptation as a lightweight and orthogonal optimization, as illustrated in Figure 1. Rather than modifying model architecture, training objectives, or agent reasoning strategies, we adapt the tokenizer to better reflect the structural and semantic regularities present in $\mathcal{D}_{\text{tool}}$. Formally, given a base vocabulary V and tokenizer τ , our goal is to construct an adapted vocabulary $V' = V \cup V_{\text{struct}} \cup V_{\text{cont}}$ and corresponding tokenizer τ' such that the expected tokenized sequence length $\mathbb{E}_{x \sim \mathcal{D}_{\text{tool}}} [|\tau'(x)|]$ is reduced relative to τ while preserving the original semantics of the input.

By explicitly reducing redundant tokenization in agentic tool-calling interactions, AgentVocab provides a practical method to improve decoding efficiency without altering model behavior or task performance.

3.2. Structure-Aware Token Induction

The first step of AgentVocab is to identify candidate vocabulary entries that capture recurring structural patterns in agentic tool-calling interactions. Our key observation is that tool calls are composed of two qualitatively different components: *structure*, which defines the schema and formatting of the interaction, and *content*, which corresponds

to variable arguments, natural language descriptions, or instance-specific values. In this section, we first focus on inducing the explicit structural tokens and leave the content tokens to Section 3.3.

To operationalize this structural distinction, we introduce a set of structural primitives that serve as reference points for candidate generation. These primitives include schema-level keywords and formatting symbols that define the skeleton of tool calls. Concretely, we collect reserved keywords from publicly available specifications such as the JSON Schema standard¹. These include schema-related keywords such as “type”, “properties”, and “description”, as well as special fields like “\$schema”, “\$id”, and “\$ref”. In addition, we also include keywords specific to function calling, such as “function”, “name”, and “arguments”, as well as syntactic delimiters like brackets, quotation marks, and separators. Importantly, these primitives are used solely to identify structural regions and do not directly determine which strings are added to the vocabulary.

Given a tool-calling sequence $x = (w_1, \dots, w_n)$, where each w_i is classified as either structural or non-structural, we define a structural span as any maximal contiguous sub-sequence:

$$s = (w_i, \dots, w_j), \quad \text{s.t. } w_k \in \mathcal{A} \forall k \in [i, j] \quad (1)$$

where \mathcal{A} denotes the set of structural primitives and maximality implies that either $i = 1, j = n$, or $w_{i-1}, w_{j+1} \notin \mathcal{A}$. In practice, such spans are extracted via a single-pass scan over the sequence, where span construction proceeds until a non-structural unit is encountered, ensuring that variable content such as argument values or natural language text does not enter s . Each resulting span thus corresponds to a self-contained structural fragment of schema or formatting that recurs across tool-calling interactions. Then, the extracted structural spans are aggregated to form the final structural vocabulary V_{struct} , which contains all the frequent and reusable structural tokens. Formally, we define:

$$V_{\text{struct}} = \{s_1, s_2, \dots, s_m\} \quad (2)$$

By construction, the proposed procedure operates at a fine-grained level of structural decomposition. Rather than treating an entire serialized tool call or schema definition as a single unit, we decompose complex structures into reusable structural fragments, such as individual field definitions or per-tool schema components. This design avoids overfitting to specific tool instances and allows the resulting candidates to generalize across different tools and interaction contexts. Figure 2 illustrates the qualitative difference between the base tokenizer and AgentVocab’s structural adaptation. As shown, the mined structural spans successfully fuse frag-

¹<https://json-schema.org/draft-07/schema>

Base Tokenizer

```
<tools>\n {" type ": " function ", " function ": {" name ": "...", " description ": (...), " parameters ": {" type ": " object ", " properties ": {"
```

Structure-Aware Tokenization (Ours)

```
<tools>\n {"type": "function", "function": {"name": "...", "description": "...", "parameters": {"type": "object", "properties": {"
```

Figure 2. Visual comparison of tokenization granularity. Alternating background colors are used to distinguish individual token boundaries. Top: The Base Tokenizer fragments syntax into multiple subwords. Bottom: Our Structure-Aware Tokenization fuses these fragments into coherent structural spans. Gray blocks marked with “(...)” represent variable content omitted for clarity. Note: While AgentVocab also optimizes content tokens (as detailed in Section 3.3), this figure highlights the structural adaptation.

mented syntax into coherent tokens, significantly reducing the sequence length.

3.3. Content-Aware Token Induction

While structure-aware tokenization efficiently compresses the static schema structures, agentic interactions also involve repetitive patterns within variable fields, such as frequent parameter values, common code identifiers, or recurring natural language instructions. To capture these semantic regularities, we apply a Token-Level Byte Pair Encoding (TL-BPE) algorithm on the tool-calling corpus $\mathcal{D}_{\text{tool}}$. This approach enables the discovery of diverse semantic units, ranging from pure content phrases to the associative patterns that bridge structure and content, which are essential for reducing token fragmentation.

The induction process starts with the sequence of token IDs produced by the base tokenizer τ . We iteratively count and merge adjacent token pairs, enforcing a safety constraint that prevents the merging of any pair containing control tokens like BOS and EOS to preserve the model’s functional integrity. To prioritize candidates that offer the most significant efficiency gains, we employ a savings-based ranking metric rather than relying solely on frequency. For a candidate content token c composed of $L(c)$ base tokens with a corpus frequency $f(c)$, we define its score $S(c)$ as:

$$S(c) = f(c) \cdot (L(c) - 1) \quad (3)$$

Here, the term $(L(c) - 1)$ represents the net reduction in sequence length for a single occurrence. Consequently, $S(c)$ quantifies the total expected token savings across the entire corpus, favoring patterns that maximally contribute to global compression. We select the top- K candidates with the highest $S(c)$ scores, where the budget K is set to be consistent with the size of the structural vocabulary m . The final content vocabulary is formally defined as:

$$V_{\text{cont}} = \{c \mid c \in \text{Top-}K_S(\mathcal{D}_{\text{tool}})\} \quad (4)$$

This ensures that V_{cont} supplements the vocabulary with

high-utility semantic content and boundary patterns.

3.4. Vocabulary Integration and Training

The adapted vocabulary produced by AgentVocab is formed by augmenting the original vocabulary V with the structural tokens and content tokens identified in the previous stages. We denote the final vocabulary as:

$$V' = V \cup V_{\text{struct}} \cup V_{\text{cont}}, \quad (5)$$

Since V_{cont} is extracted from the full dataset, it may independently rediscover structural patterns already covered by V_{struct} . The union operation naturally removes such redundancies. Representative examples of the extracted structural and content tokens are provided in Appendix A.1.

For embedding initialization, following Cui et al. (2023), embeddings for new tokens are initialized by averaging the embeddings of their constituent sub-tokens under the base tokenizer. This strategy provides a semantically meaningful starting point and stabilizes training. During fine-tuning, both the original model parameters and the embedding matrix corresponding to the adapted vocabulary are updated jointly. We train the model using standard supervised fine-tuning (SFT) on agentic tool-calling data. For a tool-calling sequence x drawn from $\mathcal{D}_{\text{tool}}$, we minimize the SFT objective under the adapted tokenizer:

$$\mathcal{L} = - \sum_{t=1}^{|x|} \log p_{\theta}(w_t | w_{<t}), \quad (6)$$

where θ includes both the model parameters and the embedding parameters associated with V' . By enabling frequent structural and content patterns to be generated as single tokens, this training procedure reduces decoding steps for tool calls while preserving the original task behavior.

4. Experiments

4.1. Experimental Setup

Training Data We fine-tune our models on the dataset released in the TouCan framework (Xu et al., 2025b). It provides a large-scale collection of tool-use trajectories synthesized from real-world Model Context Protocol (MCP) environments, featuring complex planning and execution sequences. We specifically employ the curated SFT subset, which contains about 119.3K tool interactions and structured reasoning traces, making it well suited for training agentic tool-calling behavior. We apply AgentVocab to this dataset and extract 1,524 structural and content tokens. Taking the union of the two sets results in the addition of 3,006 tokens to the vocabulary.

Implementation Details We adopt Qwen2.5-7B-Instruct as our base model and conduct full-parameter SFT using

the Ms-Swift² framework. Training is carried out on 8 NVIDIA A800 GPUs, using DeepSpeed ZeRO-3 optimization. For the vanilla configuration without vocabulary adaptation, models are trained for 2,000 steps using the original tokenizer. Models with AgentVocab are trained for 5,000 steps to accommodate the learning of newly introduced vocabulary embeddings. As shown in Figure 3, the vanilla model reaches a stable accuracy plateau from 2,000 to 3,500 training steps, while decoding latency is lowest at earlier checkpoints; accordingly, we report results at 2,000 steps. Detailed hyperparameters and training configurations are provided in the Appendix A.2.

Evaluation Benchmarks We evaluate our approach on two challenging agentic benchmarks: τ -bench (Yao et al., 2025) and τ^2 -bench (Barres et al., 2025). τ -bench assesses the agent’s capability to handle dynamic user interactions and database tools in realistic domains such as Retail and Airline. τ^2 -bench extends this by introducing a dual-control environment that demands more complex state tracking and multi-turn reasoning capabilities, providing a rigorous testbed for evaluating the robustness of the agent’s reasoning and planning.

Evaluation Metrics Our evaluation framework analyzes performance across two key dimensions: accuracy and efficiency. For Accuracy (Acc.), we report the Pass Rate on the aforementioned benchmarks, measuring the percentage of user requests successfully resolved end-to-end. To quantify the efficiency gains introduced by AgentVocab, we report the Average Latency (Lat.), measuring the wall-clock time (in seconds) required for response generation. Additionally, we track the Average Input Tokens (Input.) and Average Output Tokens (Output.) per turn to evaluate the model’s token economy, serving as a proxy for both computational cost and the conciseness of the agent’s internal reasoning and communication processes.

4.2. Main Results

Tables 1 and 2 present the main results on τ -bench and τ^2 -bench, respectively. We compare the base Qwen2.5-7B-Instruct model, a vanilla fine-tuned model using the original tokenizer, and a model fine-tuned with AgentVocab. Results are reported both overall and across representative domains for each benchmark.

Overall Performance Across both benchmarks, AgentVocab consistently achieves comparable or improved task accuracy relative to the vanilla fine-tuned baseline, while substantially improving efficiency. On τ -bench, AgentVocab improves overall accuracy from 19.40% to 20.61%, while on the more challenging τ^2 -bench, it slightly improves the overall score from 21.56% to 21.93%. Although these ab-

²<https://github.com/modelscope/ms-swift>

Table 1. Experimental results on τ -bench. Green numbers indicate improvements relative to the Qwen2.5-7B-Instruct baseline (“+” for accuracy gain, “-” for cost/latency reduction).

Model	Airline				Retail				Overall			
	Acc.	Input.	Output.	Lat.	Acc.	Input.	Output.	Lat.	Acc.	Input.	Output.	Lat.
Qwen2.5-7B-Instruct	8.00%	7334.5	177.9	0.302	16.52%	7081.6	124.0	0.217	13.94%	7156.4	139.9	0.242
Vanilla SFT	10.00%	7465.7	172.0	0.359	23.48%	7588.0	123.3	0.187	19.40%	7553.6	137.0	0.235
AgentVocab SFT	20.00%	4969.3	124.8	0.188	20.87%	5003.5	113.8	0.165	20.61% ^{+47.8%}	4992.6 ^{-30.2%}	117.3 ^{-16.2%}	0.173 ^{-28.5%}

Table 2. Experimental results on τ^2 -bench.

Model	Airline				Retail				Telecom				Overall			
	Acc.	Input.	Output.	Lat.	Acc.	Input.	Output.	Lat.	Acc.	Input.	Output.	Lat.	Acc.	Input.	Output.	Lat.
Qwen2.5-7B-Instruct	9.30%	9927.7	188.3	0.563	11.61%	8570.5	146.4	0.314	23.68%	9626.0	150.5	0.428	16.36%	9252.4	154.9	0.404
Vanilla SFT	13.95%	9334.9	194.2	0.379	29.46%	8154.0	147.9	0.306	16.67%	9918.4	195.0	0.388	21.56%	9225.5	178.8	0.359
AgentVocab SFT	16.28%	5842.5	151.4	0.255	19.64%	5505.4	112.1	0.188	26.32%	8105.2	178.7	0.405	21.93% ^{+34.0%}	6795.2 ^{-26.6%}	149.7 ^{-3.4%}	0.302 ^{-25.2%}

solute gains are modest, they demonstrate that vocabulary adaptation preserves the accuracy benefits obtained through SFT, rather than trading correctness for efficiency.

Efficiency improvements, however, clearly distinguish AgentVocab from the vanilla configuration. Across both benchmarks, AgentVocab reduces the average number of input tokens by roughly 30% and lowers decoding latency by approximately 15–25%. These reductions indicate that vocabulary adaptation substantially shortens both the accumulated interaction context and the generated responses, leading to faster end-to-end inference. Importantly, these improvements are observed consistently across benchmarks with different levels of interaction complexity, suggesting that the efficiency benefits of AgentVocab generalize beyond a single evaluation setting.

Domain-Level Results The efficiency benefits of AgentVocab are robust across domains in both benchmarks, despite differences in task structure and tool schemas. On τ -bench, AgentVocab substantially reduces input tokens in both Airline and Retail domains, with reductions exceeding 30%, accompanied by clear latency improvements. These gains suggest that vocabulary adaptation is particularly effective in settings with repeated structured interactions and schema-heavy tool usage. On τ^2 -bench, this efficiency advantage remains robust. Airline and Retail maintain input reductions exceeding 30% relative to the vanilla baseline, translating into significant latency drops. In Telecom, although the reduction in output tokens is less pronounced, AgentVocab still shortens the average input length from 9,918 to 8,105 tokens and improves accuracy from 16.67% to 26.32%. This indicates that even in domains with more diverse or less repetitive generation patterns, vocabulary adaptation can improve efficiency while stabilizing tool-calling behavior.

Accuracy–Efficiency Trade-off A key observation across both τ -bench and τ^2 -bench is that the substantial efficiency gains introduced by AgentVocab do not come at the expense

of overall task effectiveness. Despite significant reductions in token usage and decoding latency, the aggregate accuracy remains stable or improves slightly on both benchmarks. While we observe some performance variance at the domain level, where a dip in Retail domain of τ^2 -bench is balanced by substantial gains in Telecom, the macro-level performance is preserved. This behavior contrasts with many efficiency-oriented interventions that shorten sequences at the cost of degraded decision quality. Instead, AgentVocab achieves a favorable balance, simultaneously optimizing computational cost and maintaining the model’s reasoning capabilities. These findings are consistent with the training dynamics analyzed in Section 4.3, where accuracy and efficiency are shown to improve in tandem during adaptation.

Comparison with the Base Model Compared to the original Qwen2.5-7B-Instruct model without fine-tuning, both fine-tuned variants achieve substantially higher accuracy on both benchmarks, confirming the importance of supervised agent training. On τ -bench, accuracy increases from 13.94% to above 20%, while on τ^2 -bench it rises from 16.36% to above 21%. AgentVocab further distinguishes itself by delivering these accuracy gains with significantly lower computational overhead. Relative to the base model, AgentVocab reduces the average input length by more than 2,400 tokens on τ^2 -bench and by over 2,100 tokens on τ -bench, while reducing decoding latency by roughly 25%. This comparison highlights vocabulary adaptation as a complementary optimization that improves efficiency beyond what parameter fine-tuning alone can achieve.

4.3. Training Dynamics and Stability

Taking τ^2 -bench as a representative example, we conduct a fine-grained evaluation every 500 training steps to analyze the training dynamics of vocabulary adaptation in agentic tool calling. Figure 3 summarizes the evolution of model behavior across different vocabulary configurations.

Instability in Vanilla SFT We first observe a difference

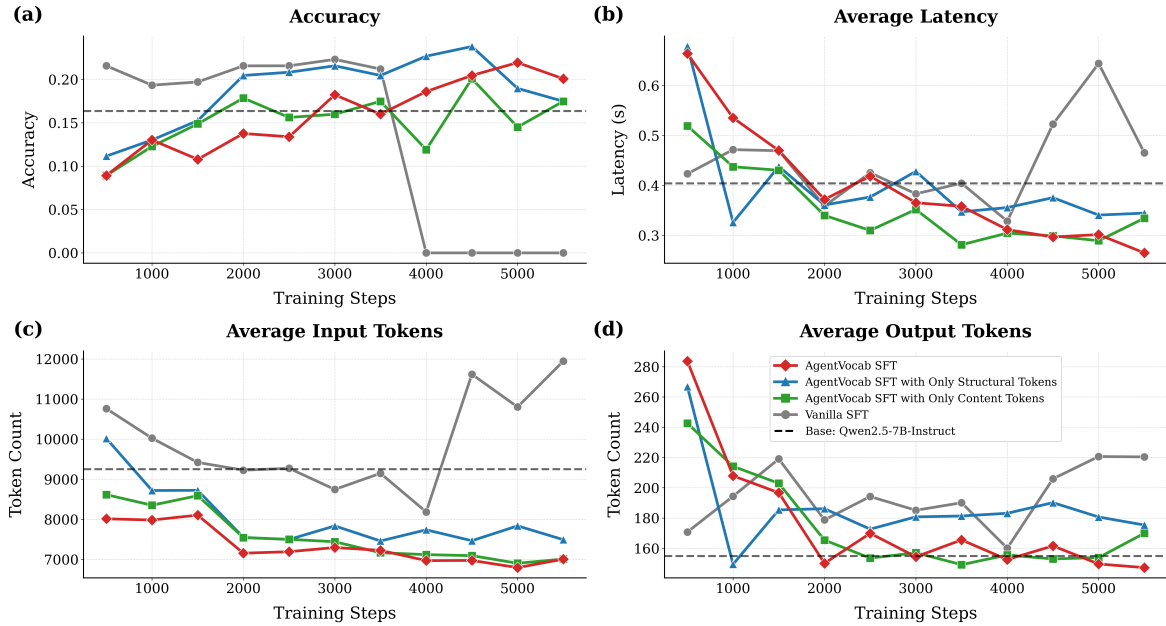


Figure 3. Step-wise evaluation of training dynamics on τ^2 -bench. The AgentVocab model exhibits a favorable joint trend, with steadily improving accuracy alongside decreasing token usage and decoding latency as training progresses. Ablated variants with only structural or only content tokens reveal complementary effects on correctness and efficiency.

in training stability between the vanilla SFT model and AgentVocab SFT models. While the vanilla model initially improves with training, it later exhibits abrupt failures after 3,500 steps. Manual inspection indicates that these failures stem from accumulated format errors during multi-turn tool interactions, which lead to invalid or unparseable outputs under the evaluation protocol. We attribute this collapse to the observation that, without vocabulary adaptation, continued optimization primarily reinforces surface-level patterns in the training data, which may amplify spurious behaviors or overfitting effects in long-horizon interactions. In contrast, models trained with AgentVocab remain stable throughout training, suggesting that vocabulary adaptation promotes more robust convergence under extended optimization.

Concurrent Improvement in Accuracy and Efficiency

A key observation from the step-wise evaluation is the favorable training dynamic exhibited by the AgentVocab model: as training progresses, task accuracy steadily increases, while efficiency-related metrics simultaneously improve. Specifically, the AgentVocab configuration demonstrates an inverse correlation between performance and computational cost, showing a clear upward trend in accuracy alongside a consistent reduction in input tokens, output tokens, and decoding latency. This joint trend is notable, as improvements in efficiency often come at the cost of reduced expressiveness or effectiveness in agentic settings. Here, however, the adapted vocabulary enables the model to become both more reliable and efficient over time. This suggests that the model is not merely shortening its outputs via memorization, but is

actively internalizing the specialized vocabulary to represent complex schemas and intent with higher semantic density. Consequently, the model learns to perform tool-calling interactions in a progressively more effective and economical manner, aligning its latent reasoning process with the compact representation space offered by AgentVocab.

Understanding the Sources of Improvement To disentangle the sources of this joint improvement, we compare the full AgentVocab model with ablated variants that include only structural tokens or only content tokens. The structural-only configuration consistently achieves higher accuracy throughout training, indicating that structure-aware tokenization is the primary contributor to tool-calling correctness. By reducing fragmentation at the schema level and providing clearer atomic boundaries for protocol-critical patterns, structural tokens help the model maintain valid actions over long interaction horizons. In contrast, the content-only configuration exhibits earlier gains in efficiency, particularly in terms of latency and output length during intermediate training stages. This suggests that content tokens are effective at compressing frequently generated semantic fragments, allowing the model to complete actions with fewer decoding steps. However, without structural support, these efficiency gains are less stable and do not consistently translate into higher task success. The full AgentVocab model combines these effects. Although it requires additional training to fully converge, it gradually reaches a regime that maintains strong accuracy while achieving low latency and compact input and output representations.

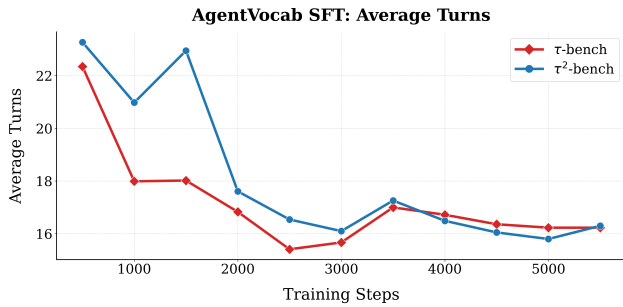


Figure 4. Average number of interaction turns of the AgentVocab model across training steps. Despite fixed evaluation prompts, the number of interaction turns required to complete a task decreases consistently as training progresses with AgentVocab. This reduction in turn count explains the observed decrease in average input tokens, indicating that vocabulary adaptation shortens multi-turn agent trajectories by enabling more precise and effective actions.

Input Token Reduction through Shorter Interaction Trajectories Beyond token-level compression, we observe that the average number of input tokens decreases over training even though the evaluation prompts remain unchanged. This indicates that the reduction cannot be explained solely by changes in tokenization, but reflects a shift in the agent’s multi-turn interaction behavior. To analyze this effect, we examine how the number of interaction turns evolves during evaluation on both benchmarks. As shown in Figure 4, the average number of turns required to complete a task decreases substantially, dropping from early checkpoints to a much lower level at later stages. This reduction closely follows the downward trend in input token counts, since each additional turn contributes accumulated dialogue history to subsequent prompts. Fewer turns therefore result in shorter effective contexts and lower input token consumption. This analysis suggests that vocabulary adaptation influences not only how individual messages are expressed, but also how efficiently the agent navigates multi-step interactions. By learning to produce more precise actions, the model reduces redundant exchanges and converges on shorter, more efficient problem-solving trajectories.

4.4. Case Study

Figure 5 presents a representative case from the Telecom domain of τ^2 -bench. The tool observation exposes a hard numerical constraint: the affected line has consumed 15.1 GB against a 15.0 GB monthly cap, which directly determines the appropriate remediation. The base model fails to operationalize this structured boundary and drifts into a redundant diagnostic loop, eventually stalling after 18 turns while consuming 8,599 input tokens without executing the refueling action. In contrast, the AgentVocab model queries data usage, identifies the overflow condition from the structured fields, and issues a valid `refuel_data` call to add 2.0 GB, resolving the task in 12 turns with 6,850

Initial Query: My mobile data is not working properly... it stops working completely or is extremely slow. I need excellent inter-net speed...

(Earlier context omitted: authentication and account retrieval)

Shared Tool Observation (excerpt)

```
// Tool Output (combined): get_details_by_id + get_data_usage
{
  "line_id": "L1002",
  "phone_number": "555-123-2002",
  // ... (omitted low-importance fields) ...
  "data_limit_gb": "15.0",
  "data_used_gb": "15.1", // <--- Critical Boundary (used > limited)
  "data_refueling_gb": "0.0"
}
```

Qwen2.5-7B-Instruct’s Response

Action: No valid tool execution (requests IDs / provides templates).

Analysis: The model fails to treat the numerical constraint (15.1 > 15.0) as a decision boundary. It perceives the values as generic text context and defaults to a redundant diagnostic loop.

Outcome: **Failure.** Interaction drags on for 18 turns (8,599 input tokens) without resolution.

AgentVocab Model’s Response

Action: `refuel_data`

Analysis: The model successfully attends to the specialized structural tokens representing usage limits. It correctly identifies the overflow and executes the correct solution.

Outcome: **Success.** Solved in 12 turns (6,850 input tokens), achieving a ~20% reduction in token usage.

Figure 5. Case study from τ^2 -bench (Telecom). A shared tool observation encodes a critical constraint. The base model fails to interpret this boundary and enters a redundant diagnostic loop, while AgentVocab correctly attends to the structured signal and resolves the task with fewer turns and tokens.

input tokens. This example illustrates that AgentVocab not only reduces token usage, but also stabilizes agent behavior by preventing redundant tool-calling loops in constraint-sensitive scenarios. The complete interaction trace for this case study is provided in Appendix A.3.

5. Conclusion

In this work, we introduced AgentVocab, a framework designed to address the misalignment between general-purpose tokenizers and the highly structured nature of agentic tool usage. By augmenting the model vocabulary with specialized structural spans and frequent content patterns, our approach effectively mitigates token fragmentation and reduces sequence redundancy without requiring architectural modifications. Extensive experiments on τ -bench and τ^2 -bench demonstrate that AgentVocab reduces decoding latency by 15–25% compared to the vanilla baseline without vocabulary adaptation, while maintaining or even enhancing tool-calling accuracy. These results establish vocabulary adaptation as a critical, lightweight optimization for deploying efficient, stable, and robust LLM agents in complex interaction environments. Together, these findings suggest that vocabulary-level adaptation is an effective method for improving the practical efficiency of agentic LLM systems.

Impact Statement

This paper advances machine learning for agentic tool use by improving tokenization efficiency in structured tool-calling interactions. The primary expected benefit is reduced sequence length and lower inference latency, which can translate into lower compute cost and energy use for systems that rely on frequent tool calls. Because efficiency improvements can also lower the barrier to deploying large-scale automation, the same techniques could be used in settings that amplify existing misuse risks. In addition, applying vocabulary mining to real interaction logs may raise privacy and compliance concerns if logs contain sensitive information. Our method does not introduce new model capabilities or change model architecture; it primarily affects how existing behaviors are represented and generated. To mitigate foreseeable risks in practical use, we recommend standard safeguards such as access controls and rate limiting for tool execution, logging and auditing, and privacy-preserving data handling when constructing adaptation corpora. The broader downstream impacts will depend on deployment context and should be evaluated accordingly.

References

- Aksitov, R., Miryoosefi, S., Li, Z., Li, D., Babayan, S., Kopparapu, K., Fisher, Z., Guo, R., Prakash, S., Srinivasan, P., Zaheer, M., Yu, F., and Kumar, S. ReST meets react: Self-improvement for multi-step reasoning LLM agent. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024. URL <https://openreview.net/forum?id=7xknRLr7QE>.
- Anthropic. Claude model report. 2025. URL <https://www.anthropic.com/transparency/model-report/>.
- Barres, V., Dong, H., Ray, S., Si, X., and Narasimhan, K. τ^2 -bench: Evaluating conversational agents in a dual-control environment. *ArXiv preprint*, abs/2506.07982, 2025. URL <https://arxiv.org/abs/2506.07982>.
- Cui, Y., Yang, Z., and Yao, X. Efficient and effective text encoding for chinese llama and alpaca. *ArXiv preprint*, abs/2304.08177, 2023. URL <https://arxiv.org/abs/2304.08177>.
- Dagan, G., Synnaeve, G., and Rozière, B. Getting the most out of your tokenizer for pre-training and domain adaptation. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL <https://openreview.net/forum?id=ZFYBnLljtT>.
- Dou, Z.-Y., Yang, C.-F., Wu, X., Chang, K.-W., and Peng, N. Re-ReST: Reflection-reinforced self-training for language agents. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15394–15411, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.861. URL <https://aclanthology.org/2024.emnlp-main.861/>.
- Gee, L., Zugarini, A., Rigutini, L., and Torrioni, P. Fast vocabulary transfer for language model compression. In Li, Y. and Lazaridou, A. (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 409–416, Abu Dhabi, UAE, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-industry.41. URL <https://aclanthology.org/2022.emnlp-industry.41/>.
- Herold, C., Kozielski, M., Santavas, N., Versley, Y., and Khadivi, S. Vocabulary customization for efficient domain-specific LLM deployment. In *AI That Keeps Up: NeurIPS 2025 Workshop on Continual and Compatible Foundation Model Updates*, 2025. URL <https://openreview.net/forum?id=t76J1Mczom>.
- Huang, T.-J., Yang, J.-Q., Shen, C., Liu, K.-Q., Zhan, D.-C., and Ye, H.-J. Improving LLMs for recommendation with out-of-vocabulary tokens. In *Forty-second International Conference on Machine Learning*, 2025. URL <https://openreview.net/forum?id=cerqDkPLx7>.
- Hui, B., Yang, J., Cui, Z., Yang, J., Liu, D., Zhang, L., Liu, T., Zhang, J., Yu, B., Lu, K., et al. Qwen2. 5-coder technical report. *ArXiv preprint*, abs/2409.12186, 2024. URL <https://arxiv.org/abs/2409.12186>.
- Karim, K., Batatia, H. R. H., et al. Innovative tokenisation of structured data for llm training. *ArXiv preprint*, abs/2508.01685, 2025. URL <https://arxiv.org/abs/2508.01685>.
- Li, M., Zhao, S., Wang, Q., Wang, K., Zhou, Y., Srivastava, S., Gokmen, C., Lee, T., Li, L. E., Zhang, R., Liu, W., Liang, P., Fei-Fei, L., Mao, J., and Wu, J. Embodied agent interface: Benchmarking llms for embodied decision making. In Globersons, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J. M., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL <https://arxiv.org/abs/2410.07166>.
- Li, Y., Zhang, W., Yang, Y., Huang, W.-C., Wu, Y., Luo, J., Bei, Y., Zou, H. P., Luo, X., Zhao, Y., Chan, C., Chen, Y., Deng, Z., Li, Y., Zheng, H.-T., Li, D., Jiang, R., Zhang, M., Song, Y., and Yu, P. S. A survey of RAG-reasoning systems in large language models. In Christodoulopoulos,

- 495 C., Chakraborty, T., Rose, C., and Peng, V. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 12120–12145, Suzhou, China, 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.648. URL <https://aclanthology.org/2025.findings-emnlp.648/>.
- 502 Liang, Q., Zhang, Z., Sun, Z., Lin, Z., Luo, Q., Yueyi, X., Chen, Y., Zhang, Y., Zhang, H., Zhang, L., Chenbin, C., and Xiong, Y. Grammar-based code representation: Is it a worthy pursuit for LLMs? In Che, W., Nabende, J., Shutova, E., and Pilehvar, M. T. (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 15640–15653, Vienna, Austria, 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.807. URL <https://aclanthology.org/2025.findings-acl.807/>.
- 514 Liu, C., Wang, S., Qing, L., Kuang, K., Kang, Y., Sun, C., and Wu, F. Gold panning in vocabulary: An adaptive method for vocabulary expansion of domain-specific LLMs. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7442–7459, Miami, Florida, USA, 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.424. URL <https://aclanthology.org/2024.emnlp-main.424/>.
- 525 Liu, Z., Hoang, T., Zhang, J., Zhu, M., Lan, T., Kokane, S., Tan, J., Yao, W., Liu, Z., Feng, Y., N., R. R., Yang, L., Savarese, S., Niebles, J. C., Wang, H., Heinecke, S., and Xiong, C. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. In Globersons, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J. M., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024b. URL <https://arxiv.org/abs/2406.18518>.
- 538 Lu, J., Holleis, T., Zhang, Y., Aumayer, B., Nan, F., Bai, H., Ma, S., Ma, S., Li, M., Yin, G., Wang, Z., and Pang, R. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In Chiruzzo, L., Ritter, A., and Wang, L. (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, Albuquerque, New Mexico, 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.65. URL <https://aclanthology.org/2025.findings-naacl.65/>.
- Mon-Williams, R., Li, G., Long, R., Du, W., and Lucas, C. G. Embodied large language models enable robots to complete complex tasks in unpredictable environments. *Nature Machine Intelligence*, pp. 1–10, 2025. URL <https://www.nature.com/articles/s42256-025-01005-x>.
- Mu, J., Li, X., and Goodman, N. D. Learning to compress prompts with gist tokens. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL <https://arxiv.org/abs/2304.08467>.
- Nakash, I., Calderon, N., Ben-David, E., Hoffer, E., and Reichart, R. Adaptivocab: Enhancing LLM efficiency in focused domains through lightweight vocabulary adaptation. In *Second Conference on Language Modeling*, 2025. URL <https://openreview.net/forum?id=TyXf9dwpZP>.
- OpenAI. Gpt-5 system card. 2025. URL <https://cdn.openai.com/gpt-5-system-card.pdf>.
- Özeren, E., Liu, Y., and Schuetze, H. HYPEROFA: Expanding LLM vocabulary to new languages via hypernetwork-based embedding initialization. In Zhao, J., Wang, M., and Liu, Z. (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pp. 79–96, Vienna, Austria, 2025. Association for Computational Linguistics. ISBN 979-8-89176-254-1. doi: 10.18653/v1/2025.acl-srw.6. URL <https://aclanthology.org/2025.acl-srw.6/>.
- Patil, S. G., Zhang, T., Wang, X., and Gonzalez, J. E. Gorilla: Large language model connected with massive apis. In Globersons, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J. M., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL <https://arxiv.org/abs/2305.15334>.
- Patil, S. G., Mao, H., Yan, F., Ji, C. C.-J., Suresh, V., Stoica, I., and Gonzalez, J. E. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL <https://openreview.net/forum?id=2GmDdhBdDk>.
- Prabhakar, A., Liu, Z., Zhu, M., Zhang, J., Awalgaonkar, T. M., Wang, S., Liu, Z., Chen, H., Hoang, T. Q., Niebles, J. C., Heinecke, S., Yao, W., Wang, H., Savarese, S., and

- Xiong, C. APIGen-MT: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL <https://openreview.net/forum?id=qk60RqQ4Cu>.
- Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., Zhao, S., Hong, L., Tian, R., Xie, R., Zhou, J., Gerstein, M., Li, D., Liu, Z., and Sun, M. Toollm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL <https://openreview.net/forum?id=dHng200Jjr>.
- Salimpour, S., Fu, L., Rachwał, K., Bertrand, P., O’Sullivan, K., Jakob, R., Keramat, F., Militano, L., Toffetti, G., Edelman, H., et al. Towards embodied agentic ai: Review and classification of llm-and vlm-driven robot autonomy and interaction. *ArXiv preprint*, abs/2508.05294, 2025. URL <https://arxiv.org/abs/2508.05294>.
- Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL <https://arxiv.org/abs/2302.04761>.
- Shao, Z., Luo, Y., Lu, C., Ren, Z., Hu, J., Ye, T., Gou, Z., Ma, S., and Zhang, X. Deepseekmath-v2: Towards self-verifiable mathematical reasoning. *ArXiv preprint*, abs/2511.22570, 2025. URL <https://arxiv.org/abs/2511.22570>.
- Wang, X., Chen, Y., Yuan, L., Zhang, Y., Li, Y., Peng, H., and Ji, H. Executable code actions elicit better LLM agents. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL <https://openreview.net/forum?id=jJ9BoXAfFa>.
- Xu, R., Liu, H., Nag, S., Dai, Z., Xie, Y., Tang, X., Luo, C., Li, Y., Ho, J. C., Yang, C., and He, Q. SimRAG: Self-improving retrieval-augmented generation for adapting large language models to specialized domains. In Chiruzzo, L., Ritter, A., and Wang, L. (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 11534–11550, Albuquerque, New Mexico, 2025a. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.575. URL <https://aclanthology.org/2025.naacl-long.575/>.
- Xu, Z., Soria, A. M., Tan, S., Roy, A., Agrawal, A. S., Poovendran, R., and Panda, R. Toucan: Synthesizing 1.5 m tool-agentic data from real-world mcp environments. *ArXiv preprint*, abs/2510.01179, 2025b. URL <https://arxiv.org/abs/2510.01179>.
- Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *ArXiv preprint*, abs/2505.09388, 2025. URL <https://arxiv.org/abs/2505.09388>.
- Yao, S., Shinn, N., Razavi, P., and Narasimhan, K. R. τ -bench: A benchmark for tool-agent-user interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=roNSXZpUDN>.
- Yin, F., Wang, Z., Hsu, I.-H., Yan, J., Jiang, K., Chen, Y., Gu, J., Le, L., Chang, K.-W., Lee, C.-Y., Palangi, H., and Pfister, T. Magnet: Multi-turn tool-use data synthesis and distillation via graph translation. In Che, W., Nabende, J., Shutova, E., and Pilehvar, M. T. (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32600–32616, Vienna, Austria, 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1566. URL <https://aclanthology.org/2025.acl-long.1566/>.
- Yue, L., Somasekharan, N., Cao, Y., and Pan, S. Foam-agent: Towards automated intelligent cfd workflows. *ArXiv preprint*, abs/2505.04997, 2025. URL <https://arxiv.org/abs/2505.04997>.
- Zeng, A., Lv, X., Zheng, Q., Hou, Z., Chen, B., Xie, C., Wang, C., Yin, D., Zeng, H., Zhang, J., et al. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *ArXiv preprint*, abs/2508.06471, 2025. URL <https://arxiv.org/abs/2508.06471>.

605 A. Appendix

606 A.1. Qualitative Analysis of Learned Vocabulary

607 To elucidate the inner workings of AgentVocab, this section provides a qualitative analysis of its induced vocabulary.
 608 The tokens, extracted directly from the training corpora, serve as empirical evidence of how our framework captures the
 609 underlying regularities of agentic tool-use. We categorize these entries into two distinct classes, corresponding to the
 610 dual-branch strategy of our method. For clarity of presentation, we report 100 representative tokens for each category. In
 611 the lists provided below, each line corresponds to a single token. Tokens exceeding the line width are wrapped, with the
 612 continuation marked by a blue arrow (→).
 613

614 **Structural Tokens** The following list presents representative structural tokens. These entries constitute the structural
 615 backbone of agent interactions, primarily consisting of high-frequency JSON Schema definitions, function signatures,
 616 and syntactic delimiters. By treating these complex, recurring sequences as atomic units, AgentVocab not only reduces
 617 sequence length but also enhances the model’s robustness against syntactic formatting errors, as the model can generate
 618 entire structural skeletons in fewer, more reliable decoding steps.
 619

```

620 {"type": "function", "function": {"name": "
621 ", "parameters": {"type": "object", "properties": {"
622 ": {"type": "string", "description": "
623 ", "description": "
624 "], "additionalProperties": false, "$schema": "
625 ", "type": "object"}}}
626 ": {"type": "number", "description": "
627 ", "parameters": {"properties": {"
628 ", "
629 "}}, "required": ["
630 ": {"title": "
631 ", "type": "string"}}, "required": ["
632 ": {"type": "string", "enum": ["
633 ", "arguments": {"
634 "], "title": "
635 {"name": "
636 ", "type": "string"}, "
637 ": {"type": "boolean", "description": "
638 ": {"type": "number", "default":
639 ", "type": "integer"}}, "required": ["
640 "}, "
641 , "description": "
642 ", "type": "number"}}, "required": ["
643 ": {"type": "array", "items": {"type": "string"}, "description": "
644 ", "parameters": {"type": "object", "properties": {}}}
645 ": {"type": "array", "items": {"type": "string"}}, "
646 ": {"type": "string"}, "
647 "}, {"type": "string", "const": "
648 ", "type": "number"}, "
649
650
651
652
653
654
655
656
657
658
659

```

```

660 ": {"type": "number", "minimum":
661 ": "
662 ", "type": "integer"}, "
663 "}}}
664 ", "parameters": {"properties": {}, "title": "
665 "}}, "additionalProperties": false, "$schema": "
666 <tools>
667 </tools>
668 ": {"type": "array", "items": {"type": "object", "properties": {"
669 ", "
670 ", "title": "
671 ": {"type": "array", "description": "
672 "], "description": "
673 "}, {"type": "number", "const":
674 ": {"default": null, "title": "
675 ": {"description": "
676 ", "title": "
677 "], "type": "object"}}}
678 ": {"type": "object", "properties": {"
679 ", "parameters": {"type": "object", "properties": {}, "required": []}}}
680 ", "parameters": {"type": "object", "properties": {"title": {"type": "string", "de
681 → scription": "
682 default", "parameters": {"type": "object", "properties": {"
683 ": {"type": "integer", "description": "
684 "}, "format": {"type": "string", "description": "
685 ": {"type": "string"}}, "required": ["
686 ", "maximum":
687 ", "type": "boolean"}, "
688 ": {"default":
689 ": {"default": false, "title": "
690 "]]}}
691 ": {"anyOf": [{"type": "string"}, {"type": "null"}], "default": null, "description
692 → ": "
693 ": {"type": "array", "items": {"type": "string"}}, "required": ["
694 "], "default": "
695 ": {"type": "object", "description": "
696 integer", "parameters": {"type": "object", "properties": {"number": {"type": "numb
697 → er", "description": "
698 ": {"type": "array", "items": {"type": "number"}, "description": "
699 ", "arguments": {}}
700 ", "type": "string"}}, "title": "
701 ", "parameters": {"type": "object", "properties": {"name": {"type": "string", "des
702 → cription": "
703
704
705
706
707
708
709
710
711
712
713
714

```

```

715  }}, "additionalProperties": false, "$schema": "
716  ": {"type": "string", "default": "
717  ": {"default": "
718  ": {"type": "array", "items": {"type": "string"}, "default": [], "description": "
719  ", "parameters": {"type": "object", "properties": {"type": {"anyOf": [{"type": "st
720  → ring", "const": "
721  ", "items": {"type": "string"}}, "
722  ": {"type": "array", "items": {"type": "string", "enum": ["array", "string", "
723  ": {"type": "array", "items": {"type": "string", "enum": ["
724  ": {"type": "integer", "default":
725  ", "description": "", "parameters": {"properties": {"
726  ": {"type": "array", "items": {"type": "number"}, "minItems":
727  default: false}}}, "required": ["
728  ": {"type": "boolean", "default": true, "description": "
729  ": {"type": "integer", "minimum":
730  ": {"type": "string", "minLength":
731  }, "description": {"type": "string", "description": "
732  ", "type": "boolean"}}, "required": ["
733  default: true"}, "
734  ", "parameters": {"type": "object", "properties": {"name": {"type": "string", "min
735  → Length":
736  ", "items": {"type": "object", "properties": {"
737  ", "name": "
738  ", "parameters": {"properties": {}, "type": "object"}}}
739  }}, "required": ["number"], "additionalProperties": false, "$schema": "
740  ", "type": "string"}, {"type": "null"}}, "default": null, "description": "
741  ", "type": "integer"}}, "title": "
742  ", "type": "string"}}, "required": ["name"], "title": "
743  ": {"anyOf": [{"type": "integer"}, {"type": "null"}], "default":
744  ": {"anyOf": [{"type": "string", "const": "
745  ": {"type": "integer", "exclusiveMinimum":
746  ": {"type": "boolean", "default": false, "description": "
747  ": {"type": "object", "additionalProperties": {}, "description": "
748  ", "type": "

```

Content Tokens Complementing the structural primitives, content tokens capture the semantic regularities within tool-calling arguments and agent responses. This category includes common parameter values (e.g., status codes, default arguments), domain-specific keywords, and frequent fragments from system prompts. This adaptation allows the model to reason and generate responses using higher-level semantic concepts rather than low-level subwords, further improving decoding efficiency and contextual coherence.

Qwen, created by Alibaba Cloud.
 by Alibaba Cloud. You are a helpful
 You may call one or more functions

770 one or more functions to assist with the
771 ://json-schema.org/draft-0
772 /draft-07/schema#"}
773 %20
774 to assist with the
775 one or more functions
776 You are a helpful
777 /202
780 %2C%20
782 60
783 08/2
784 - Context7-compatible library ID:
786 2025-08-
787 "202
788 2,
789 -08-
791 url": "https
792 0.7128,-7
794 28,-74.00
795 43
796 query.
798 return a
799 one or
800 1, "
802 Spec: https://
803 .000
804 5, 0
806 advanced-calculator-server
807 ": "advanced-cal
808 85
810 00:00
811 .org/openapi/stripe/Get
812 "open_library_work_key": "/works
814 .000Z", "open
815 "weather
816 /2025, 1
818 "get
819 as coordinates (lat,lng) in
820 -compatible library ID:
822 - Context7
823 "unicode": [
824

```
825 The text description of the image to generate
826 the image to generate"}, "model":
827 flux\{"). Available options: \{"flux
828 of the generated image (default:
829 image (default: 102
830 1024)"), "
831 -mcp-server-get
832 -states-weather-get
833 025-08-1
834 location as coordinates (lat,lng)
835 lat,lng) in decimal degrees.
836 in decimal degrees. Example: '4
837 Example: '40.71
838 <eventTime>2
839 eventTime>2025-
840 5",
841 00, "
842 </popularity>
843 <link>https://
844 3A%7B%22
845 . Perfect for '
846 =1
847 codigo": 3
848 "const
849 from the
850 1-
851 :00
852 const":
853 2",
854 name": "trends-hub-get
855 Z</eventTime>
856 <cover>
857 10,
858 08-2
859 ": {"prompt":
860 , with
861 % rain
862 **
863 n Returns:\{
864 , the
865 %3D&from=32
866
```

```

880 from=321841
881 drawing-tool-for-ai-assistants
882
883 ], "first_publish_year": 2
884 Z", "open": "11
885 I need
886 as coordinates (lat
887 ,lng) in
888 href=\{\}\{\}\{"
889 -imagegen-server-generateImage", "
890 .7128,-74
891 8,-74.006
892 7128,-74.
893 degrees. Example:
894 555e01a1
895 timestamp": "2
900

```

901 A.2. Training Hyperparameters

903 We use Qwen2.5-7B-Instruct as the backbone model and perform full-parameter supervised fine-tuning with the Ms-Swift framework. Training is performed on a single node equipped with 8 NVIDIA A800 GPUs. We use the AdamW optimizer with a learning rate of 2×10^{-5} , $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. To accommodate long-context agentic interactions, the maximum sequence length is set to 32,768 tokens. The per-device batch size is set to 1, with gradient accumulation over 8 steps, resulting in an effective global batch size of 64 across all GPUs. Training efficiency and memory usage are further optimized using DeepSpeed ZeRO Stage-3 and Flash Attention 2.

910 A.3. Full Interaction Traces for Case Study

911 In this section, we provide the complete, turn-by-turn interaction trajectories for the case study discussed in Section 4.4. The scenario is set in the **Telecom** domain of the τ^2 -bench evaluation.

914 **Scenario Description.** The user reports a connectivity failure (“mobile data is not working”). Environment logs indicate that the account has exceeded its 15.0 GB data limit (actual usage: 15.1 GB). To resolve the task, the agent must correctly identify this limit violation from the structured tool outputs and execute a `refuel_data` action.

918 **Trace 1: Qwen2.5-7B-Instruct (Failure Case).** The base *Qwen2.5-7B-Instruct* model fails to ground its reasoning in the numerical constraints provided by the tool outputs. Instead of recognizing the data limit violation, it enters a redundant diagnostic loop spanning 18 turns, repeatedly attempting irrelevant technical remedies such as APN resets and SIM reseating. The interaction eventually terminates without resolving the underlying issue, illustrating how the absence of structure-aware representations can lead to unstable and inefficient agent behavior. The complete interaction trajectory is shown below.

923 System Prompt

```

924 <instructions>
925 You are a customer service agent that helps the user according to the <policy> provided
926 below.
927 In each turn you can either:
928 - Send a message to the user.
929 - Make a tool call.
930 You cannot do both at the same time.
931 Try to be helpful and always follow the policy. Always make sure you generate valid JSON
932 only.
933 </instructions>
934 <policy>

```

AgentVocab: Structure-Aware Vocabulary Adaptation for Efficient LLM Agents

```
935 <main_policy>
936 # Telecom Agent Policy
937 The current time is 2025-02-25 12: 08: 00 EST.
938 As a telecom agent, you can help users with technical support, overdue bill
939 payment, line suspension, and plan options.
940 You should not provide any information, knowledge, or procedures not provided by the user
941 or available tools, or give subjective recommendations or comments.
942 You should only make one tool call at a time, and if you make a tool call, you should not
943 respond to the user simultaneously. If you respond to the user, you should not make a
944 tool call at the same time.
945 You should deny user requests that are against this policy.
946 You should transfer the user to a human agent if and only if the request cannot be handled
947 within the scope of your actions. To transfer, first make a tool call to
948 transfer_to_human_agents, and then send the message 'YOU ARE BEING TRANSFERRED TO A HUMAN
949 AGENT. PLEASE HOLD ON.' to the user.
950 You should try your best to resolve the issue for the user before transferring the user to
951 a human agent.
952
953 ## Domain Basics
954 ### Customer
955 Each customer has a profile containing:
956 - customer ID
957 - full name
958 - date of birth
959 - email
960 - phone number
961 - address (street, city, state, zip code)
962 - account status
963 - created date
964 - payment methods
965 - line IDs associated with their account
966 - bill IDs
967 - last extension date (for payment extensions)
968 - goodwill credit usage for the year
969 There are four account status types: Active, Suspended, Pending Verification,
970 and Closed.
971 ### Payment Method
972 Each payment method includes:
973 - method type (Credit Card, Debit Card, PayPal)
974 - account number last 4 digits
975 - expiration date (MM/YYYY format)
976
977 ### Line
978 Each line has the following attributes:
979 - line ID
980 - phone number
981 - status
982 - plan ID
983 - device ID (if applicable)
984 - data usage (in GB)
985 - data refueling (in GB)
986 - roaming status
987 - contract end date
988 - last plan change date
989 - last SIM replacement date
990 - suspension start date (if applicable)
991 There are four line status types: Active, Suspended, Pending Activation, and
992 Closed.
```

990 ### Plan
991 Each plan specifies:
992 - plan ID
993 - name
994 - data limit (in GB)
995 - monthly price
996 - data refueling price per GB
997 ### Device
998 Each device has:
999 - device ID
1000 - device type (phone, tablet, router, watch, other)
1001 - model
1002 - IMEI number (optional)
1003 - eSIM capability
1004 - activation status
1005 - activation date
1006 - last eSIM transfer date
1007 ### Bill
1008 Each bill contains:
1009 - bill ID
1010 - customer ID
1011 - billing period (start and end dates)
1012 - issue date
1013 - total amount due
1014 - due date
1015 - line items (charges, fees, credits)
1016 - status
1017 There are five bill status types: ****Draft****, ****Issued****, ****Paid****, ****Overdue****,
1018 ****Awaiting Payment****, and ****Disputed****.
1019 ## Customer Lookup
1020 You can look up customer information using:
1021 - Phone number
1022 - Customer ID
1023 - Full name with date of birth
1024 For name lookup, date of birth is required for verification purposes.
1025 ## Overdue Bill Payment
1026 You can help the user make a payment for an overdue bill.
1027 To do so you need to follow these steps:
1028 - Check the bill status to make sure it is overdue.
1029 - Check the bill amount due
1030 - Send the user a payment request for the overdue bill.
1031 - This will change the status of the bill to AWAITING PAYMENT.
1032 - Inform the user that a payment request has been sent. They should:
1033 - Check their payment requests using the `check_payment_request` tool.
1034 - If the user accepts the payment request, use the `make_payment` tool to make the payment.
1035 - After the payment is made, the bill status will be updated to PAID.
1036 - Always check that the bill status is updated to PAID before informing the user that the
1037 bill has been paid.
1038 Important:
1039 - A user can only have one bill in the AWAITING PAYMENT status at a time.
1040 - The `send_payment_request` tool will not check if the bill is overdue. You should always
1041 check that the bill is overdue before sending a payment request.
1042 ## Line Suspension
1043 When a line is suspended, the user will not have service.
1044 A line can be suspended for the following reasons:
1045 - The user has an overdue bill.
1046 - The line's contract end date is in the past.
1047 You are allowed to lift the suspension after the user has paid all their overdue bills.
1048 You are not allowed to lift the suspension if the line's contract end date is in the past,
1049

1045 even if the user has paid all their overdue bills.
1046 After you resume the line, the user will have to reboot their device to get service.
1047
1048 ## Data Refueling
1049 Each plan specify the maxium data usage per month.
1050 If the user's data usage for a line exceeds the plan's data limit, data connectivity will
1051 be lost.
1052 You can add more data to the line by "refueling" data at a price per GB specified by the
1053 plan.
1054 The maximum amount of data that can be refueled is 2GB.
1055 To refuel data you should:
1056 - Ask them how much data they want to refuel
1057 - Confirm the price
1058 - Apply the refueled data to the line associated with the phone number the user provided.
1059 ## Change Plan
1060 You can help the user change to a different plan.
1061 To do so you need to follow these steps
1062 - Make sure you know what line the user wants to change the plan for.
1063 - Gather available plans
1064 - Ask the user to select one.
1065 - Calculate the price of the new plan.
1066 - Confirm the price.
1067 - Apply the plan to the line associated with the phone number the user provided.
1068 ## Data Roaming
1069 If a line is roaming enabled, the user can use their phone's data connection in areas
1070 outside their home network.
1071 We offer data roaming to users who are traveling outside their home network.
1072 If a user is traveling outside their home network, you should check if the line is roaming
1073 enabled. If it is not, you should enable it at no cost for the user.
1074 ## Technical Support
1075 You must first identify the customer.
1076 </main_policy>
1077 <tech.support_policy>
1078 # Introduction
1079 This document serves as a comprehensive guide for technical support agents. It provides
1080 detailed procedures and troubleshooting steps to assist users experiencing common issues
1081 with their phone's cellular service, mobile data connectivity, and Multimedia Messaging
1082 Service (MMS). The manual is structured to help agents efficiently diagnose and resolve
1083 problems by outlining how these services work, common issues, and the tools available for
1084 resolution.
1085 The main sections covered are:
1086 * **Understanding and Troubleshooting Your Phone's Cellular Service**:
1087 Addresses issues related to network connection, signal strength, and SIM card problems.
1088 * **Understanding and Troubleshooting Your Phone's Mobile Data**:
1089 Focuses on problems with internet access via the cellular network, including speed and connectivity.
1090 * **Understanding and Troubleshooting MMS (Picture/Video Messaging)**:
1091 Covers issues related to sending and receiving multimedia messages.
1092 Make sure you try all the possible ways to resolve the user's issue before transferring to
1093 a human agent.
1094 # What the user can do on their device
1095 Here are the actions a user is able to take on their device.
1096 You must understand those well since as part of technical support you will have to help
1097 the customer perform series of actions
1098 ## Diagnostic Actions (Read-only)
1099 1. **check_status_bar** - Shows what icons are currently visible in your phone's status
1100 bar (the area at the top of the screen).
1101 - Airplane mode status (" Airplane Mode" when enabled)
1102 - Network signal strength (" No Signal", " Poor", " Fair", " Good", " Excellent")
1103 - Network technology (e.g., "5G", "4G", etc.)
1104 - Mobile data status (" Data Enabled" or " Data Disabled")
1105

AgentVocab: Structure-Aware Vocabulary Adaptation for Efficient LLM Agents

1100- Data saver status (" Data Saver" when enabled)
1101- Wi-Fi status (" Connected to [SSID]" or " Enabled")
1102- VPN status (" VPN Connected" when connected)
1103- Battery level (" [percentage]%">
1104 2. ****check_network_status**** - Checks your phone's connection status to cellular networks
1105 and Wi-Fi. Shows airplane mode status, signal strength, network type, whether mobile data
1106 is enabled, and whether data roaming is enabled. Signal strength can be "none", "poor"
1107 (1bar), "fair" (2 bars), "good" (3 bars), "excellent" (4+ bars).
1108 3. ****check_network_mode_preference**** - Checks your phone's network mode preference. Shows
1109 the type of cellular network your phone prefers to connect to (e.g., 5G, 4G, 3G, 2G).
1110 4. ****check_sim_status**** - Checks if your SIM card is working correctly and displays its
1111 current status. Shows if the SIM is active, missing, or locked with a PIN or PUK code.
1112 5. ****check_data_restriction_status**** - Checks if your phone has any data-limiting features
1113 active. Shows if Data Saver mode is on and whether background data usage is restricted
1114 globally.
1115 6. ****check_apn_settings**** - Checks the technical APN settings your phone uses to connect
1116 to your carrier's mobile data network. Shows current APN name and MMSC URL for picture
1117 messaging.
1118 7. ****check_wifi_status**** - Checks your Wi-Fi connection status. Shows if Wi-Fi is turned
1119 on, which network you're connected to (if any), and the signal strength.
1120 8. ****check_wifi_calling_status**** - Checks if Wi-Fi Calling is enabled on your device.
1121 This feature allows you to make and receive calls over a Wi-Fi network instead of using
1122 the cellular network.
1123 9. ****check_vpn_status**** - Checks if you're using a VPN (Virtual Private Network)
1124 connection. Shows if a VPN is active, connected, and displays any available connection
1125 details.
1126 10. ****check_installed_apps**** - Returns the name of all installed apps on the phone.
1127 11. ****check_app_status**** - Checks detailed information about a specific app. Shows its
1128 permissions and background data usage settings.
1129 12. ****check_app_permissions**** - Checks what permissions a specific app currently has.
1130 Shows if the app has access to features like storage, camera, location, etc.
1131 13. ****run_speed_test**** - Measures your current internet connection speed (download speed).
1132 Provides information about connection quality and what activities it can support.
1133 Download speed can be "unknown", "very poor", "poor", "fair", "good", or "excellent".
1134 14. ****can_send_mms**** - Checks if the messaging app can send MMS messages.
1135
1136 ## Fix Actions (Write/Modify)
1137 1. ****set_network_mode_preference**** - Changes the type of cellular network your phone
1138 prefers to connect to (e.g., 5G, 4G, 3G). Higher-speed networks (5G, 4G) provide faster
1139 data but may use more battery.
1140 2. ****toggle_airplane_mode**** - Turns Airplane Mode ON or OFF. When ON, it disconnects all
1141 wireless communications including cellular, Wi-Fi, and Bluetooth.
1142 3. ****reset_sim_card**** - Simulates removing and reinserting your SIM card. This can help
1143 resolve recognition issues.
1144 4. ****toggle_data**** - Turns your phone's mobile data connection ON or OFF. Controls
1145 whether your phone can use cellular data for internet access when Wi-Fi is unavailable.
1146 5. ****toggle_roaming**** - Turns Data Roaming ON or OFF. When ON, roaming is enabled and
1147 your phone can use data networks in areas outside your carrier's coverage.
1148 6. ****toggle_data_saver_mode**** - Turns Data Saver mode ON or OFF. When ON, it reduces data
1149 usage, which may affect data speed.
1150 7. ****set_apn_settings**** - Sets the APN settings for the phone.
1151 8. ****reset_apn_settings**** - Resets your APN settings to the default settings.
1152 9. ****toggle_wifi**** - Turns your phone's Wi-Fi radio ON or OFF. Controls whether your
1153 phone can discover and connect to wireless networks for internet access.
1154 10. ****toggle_wifi_calling**** - Turns Wi-Fi Calling ON or OFF. This feature allows you to
1155 make and receive calls over Wi-Fi instead of the cellular network, which can help in areas
1156 with weak cellular signal.
1157 11. ****connect_vpn**** - Connects to your VPN (Virtual Private Network).
1158 12. ****disconnect_vpn**** - Disconnects any active VPN (Virtual Private Network) connection.
1159 Stops routing your internet traffic through a VPN server, which might affect connection
1160 speed or access to content.
1161 13. ****grant_app_permission**** - Gives a specific permission to an app (like access to
1162 storage, camera, or location). Required for some app functions to work properly.
1163 14. ****reboot_device**** - Restarts your phone completely. This can help resolve many
1164 temporary software glitches by refreshing all running services and connections.
1165

```
1155 # Understanding and Troubleshooting Your Phone's Cellular Service
1156 This section details for agents how a user's phone connects to the cellular network (often
1157 referred to as "service") and provides procedures to troubleshoot common issues. Good
1158 cellular service is required for calls, texts, and mobile data.
1159 ## Common Service Issues and Their Causes
1160 If the user is experiencing service problems, here are some common causes:
1161 * **Airplane Mode is ON**: This disables all wireless radios, including cellular.
1162 * **SIM Card Problems**:
1163 * Not inserted or improperly seated.
1164 * Locked due to incorrect PIN/PUK entries.
1165 * **Incorrect Network Settings**: APN settings might be incorrect resulting in a loss of
1166 service.
1167 * **Carrier Issues**: Your line might be inactive due to billing problems.
1168 ## Diagnosing Service Issues
1169 `check_status_bar()` can be used to check if the user is facing a service issue.
1170 If there is cellular service, the status bar will return a signal strength indicator.
1171 ## Troubleshooting Service Problems
1172 ### Airplane Mode
1173 Airplane Mode is a feature that disables all wireless radios, including cellular. If it
1174 is enabled, it will prevent any cellular connection.
1175 You can check if Airplane Mode is ON by using `check_status_bar()` or
1176 `check_network_status()`.
1177 If it is ON, guide the user to use `toggle_airplane_mode()` to turn it OFF.
1178 ### SIM Card Issues
1179 The SIM card is the physical card that contains the user's information and allows the
1180 phone to connect to the cellular network.
1181 Problems with the SIM card can lead to a complete loss of service.
1182 The most common issue is that the SIM card is not properly seated or the user has entered
1183 the wrong PIN or PUK code.
1184 Use `check_sim_status()` to check the status of the SIM card.
1185 If it shows "Missing", guide the user to use `reseat_sim_card()` to ensure the SIM card is
1186 correctly inserted.
1187 If it shows "Locked" (due to incorrect PIN or PUK entries), **escalate to technical
1188 support for assistance with SIM security**.
1189 If it shows "Active", the SIM itself is likely okay.
1190 ### Incorrect APN Settings
1191 Access Point Name (APN) settings are crucial for network connectivity.
1192 If `check_apn_settings()` shows "Incorrect", guide the user to use `reset_apn_settings()`
1193 to reset the APN settings.
1194 After resetting the APN settings, the user must be instructed to use `reboot_device()` for
1195 the changes to apply.
1196 ### Line Suspension
1197 If the line is suspended, the user will not have cellular service.
1198 Investigate if the line is suspended. Refer to the general agent policy for guidelines on
1199 handling line suspensions.
1200 * If the line is suspended and the agent can lift the suspension (per general policy),
1201 verify if service is restored.
1202 * If the suspension cannot be lifted by the agent (e.g., due to contract end date as
1203 mentioned in general policy, or other reasons not resolvable by the agent), **escalate to
1204 technical support**.
1205 # Understanding and Troubleshooting Your Phone's Mobile Data
1206 This section explains for agents how a user's phone uses mobile data for internet access
1207 when Wi-Fi is unavailable, and details troubleshooting for common connectivity and speed
1208 issues.
1209 ## What is Mobile Data?
1210 Mobile data allows the phone to connect to the internet using the carrier's cellular
1211 network. This enables browsing websites, using apps, streaming video, and
1212 sending/receiving emails when not connected to Wi-Fi. The status bar usually shows icons
1213 like "5G", "LTE", "4G", "3G", "H+", or "E" to indicate an active mobile data connection
```

1210 and its type.

1211 ## Prerequisites for Mobile Data

1212 For mobile data to work, the user must first have **cellular service**. Refer to the

1213 "Understanding and Troubleshooting Your Phone's Cellular Service" guide if the user does

1214 not have service.

1215 ## Common Mobile Data Issues and Causes

1216 Even with cellular service, mobile data problems might occur. Common reasons include:

1217 * **Airplane Mode is ON**: Disables all wireless connections, including mobile data.

1218 * **Mobile Data is Turned OFF**: The main switch for mobile data might be disabled in the

1219 phone's settings.

1220 * **Roaming Issues (When User is Abroad)**:

1221 * Data Roaming is turned OFF on the phone.

1222 * The line is not roaming enabled.

1223 * **Data Plan Limits Reached**: The user may have used up their monthly data allowance,

1224 and the carrier has slowed down or cut off data.

1225 * **Data Saver Mode is ON**: This feature restricts background data usage and can make

1226 some apps or services seem slow or unresponsive to save data.

1227 * **VPN Issues**: An active VPN connection might be slow or misconfigured, affecting data

1228 speeds or connectivity.

1229 * **Bad Network Preferences**: The phone is set to an older network technology like

1230 2G/3G.

1231 ## Diagnosing Mobile Data Issues

1232 `run_speed_test()` can be used to check for potential issues with mobile data.

1233 When mobile data is unavailable a speed test should return 'no connection'.

1234 If data is available, a speed test will also return the data speed.

1235 Any speed below 'Excellent' is considered slow.

1236 ## Troubleshooting Mobile Data Problems

1237 ### Airplane Mode

1238 Refer to the "Understanding and Troubleshooting Your Phone's Cellular Service" section for

1239 instructions on how to check and turn off Airplane Mode.

1240 ### Mobile Data Disabled

1241 Mobile data switch allows the phone to connect to the internet using the carrier's

1242 cellular network.

1243 If `check_network_status()` shows mobile data is disabled, guide the user to use

1244 `toggle_data()` to turn mobile data ON.

1245 ### Addressing Data Roaming Problems

1246 Data roaming allows the user to use their phone's data connection in areas outside their

1247 home network (e.g. when traveling abroad).

1248 If the user is outside their carrier's primary coverage area (roaming) and mobile data

1249 isn't working, guide them to use `toggle_roaming()` to ensure Data Roaming is ON.

1250 You should check that the line associated with the phone number the user provided is

1251 roaming enabled. If it is not, the user will not be able to use their phone's data

1252 connection in areas outside their home network.

1253 Refer to the general policy for guidelines on enabling roaming.

1254 ### Data Saver Mode

1255 Data Saver mode is a feature that restricts background data usage and can affect data

1256 speeds.

1257 If `check_data_restriction_status()` shows "Data Saver mode is ON", guide the user to use

1258 `toggle_data_saver_mode()` to turn it OFF.

1259 ### VPN Connection Issues

1260 VPN (Virtual Private Network) is a feature that encrypts internet traffic and can help

1261 improve data speeds and security.

1262 However in some cases, a VPN can cause speed to drop significantly.

1263 If `check_vpn_status()` shows "VPN is ON and connected" and performance level is "Poor",

1264 guide the user to use `disconnect_vpn()` to disconnect the VPN.

1265 ### Data Plan Limits Reached

1266 Each plan specify the maxium data usage per month.

1267 If the user's data usage for a line associated with the phone number the user provided

1268 exceeds the plan's data limit, data connectivity will be lost.

1269

1265 The user has 2 options:
1266 - Change to a plan with more data.
1267 - Add more data to the line by "refueling" data at a price per GB specified by the plan.
1268 Refer to the general policy for guidelines on those options.

1269 ### Optimizing Network Mode Preferences
1270 Network mode preferences are the settings that determine the type of cellular network the
1271 phone will connect to.
1272 Using older modes like 2G/3G can significantly limit speed.
1273 If `check_network_mode_preference()` shows "2G" or "3G", guide the user to use
1274 `set_network_mode_preference(mode: str)` with the mode `"4g_5g_preferred"` to allow the
1275 phone to connect to 5G.

1276 # Understanding and Troubleshooting MMS (Picture/Video Messaging)
1277 This section explains for agents how to troubleshoot Multimedia Messaging Service (MMS),
1278 which allows users to send and receive messages containing pictures, videos, or audio.

1279 ## What is MMS?
1280 MMS is an extension of SMS (text messaging) that allows for multimedia content. When a
1281 user sends a photo to a friend via their messaging app, they're typically using MMS.

1282 ## Prerequisites for MMS
1283 For MMS to work, the user must have cellular service and mobile data (any speed).
1284 Refer to the "Understanding and Troubleshooting Your Phone's Cellular Service" and
1285 "Understanding and Troubleshooting Your Phone's Mobile Data" sections for more
1286 information.

1287 ## Common MMS Issues and Causes
1288 * **No Cellular Service or Mobile Data Off/Not Working**:
1289 The most common reasons. MMS
1290 relies on these.
1291 * **Incorrect APN Settings**:
1292 Specifically, a missing or incorrect MMSC URL.
1293 * **Connected to 2G Network**:
1294 2G networks are generally not suitable for MMS.
1295 * **Wi-Fi Calling Configuration**:
1296 In some cases, how Wi-Fi Calling is configured can
1297 affect MMS, especially if your carrier doesn't support MMS over Wi-Fi.
1298 * **App Permissions**:
1299 The messaging app needs permission to access storage (for the
1300 media files) and usually SMS functionalities.

1301 ## Diagnosing MMS Issues
1302 `can_send_mms()` tool on the user's phone can be used to check if the user is facing an MMS
1303 issue.

1304 ## Troubleshooting MMS Problems
1305 ### Ensuring Basic Connectivity for MMS
1306 Successful MMS messaging relies on fundamental service and data connectivity. This
1307 section covers verifying these prerequisites.
1308 First, ensure the user can make calls and that their mobile data is working for other apps
1309 (e.g., browsing the web). Refer to the "Understanding and Troubleshooting Your Phone's
1310 Cellular Service" and "Understanding and Troubleshooting Your Phone's Mobile Data"
1311 sections if needed.

1312 ### Unsuitable Network Technology for MMS
1313 MMS has specific network requirements; older technologies like 2G are insufficient. This
1314 section explains how to check the network type and change it if necessary.
1315 MMS requires at least a 3G network connection; 2G networks are generally not suitable.
1316 If `check_network_status()` shows "2G", guide the user to use
1317 `set_network_mode_preference(mode: str)` to switch to a network mode that includes 3G, 4G,
1318 or 5G (e.g., `"4g_5g_preferred"` or `"4g_only"`).

1319 ### Verifying APN (MMSC URL) for MMS
1320 MMSC is the Multimedia Messaging Service Center. It is the server that handles MMS
1321 messages. Without a correct MMSC URL, the user will not be able to send or receive MMS
1322 messages.
1323 Those are specified as part of the APN settings. Incorrect MMSC URL, are a very common
1324 cause of MMS issues.
1325 If `check_apn_settings()` shows MMSC URL is not set, guide the user to use
1326 `reset_apn_settings()` to reset the APN settings.
1327 After resetting the APN settings, the user must be instructed to use `reboot_device()` for
1328 the changes to apply.
1329

```
1320 ### Investigating Wi-Fi Calling Interference with MMS
1321 Wi-Fi Calling settings can sometimes conflict with MMS functionality.
1322 If `check_wifi_calling_status()` shows "Wi-Fi Calling is ON", guide the user to use
1323 `toggle_wifi_calling()` to turn it OFF.
1324 ### Messaging App Lacks Necessary Permissions
1325 The messaging app needs specific permissions to handle media and send messages.
1326 If `check_app_permissions(app_name="messaging")` shows "storage" and "sms" permissions are
1327 not listed as granted, guide the user to use `grant_app_permission(app_name="messaging",
1328 permission="storage")` and `grant_app_permission(app_name="messaging", permission="sms")`
1329 to grant the necessary permissions.
1329 </tech_support_policy>
1330 </policy>
1331 # Tools
1332 You may call one or more functions to assist with the user query.
1333
1334 You are provided with function signatures within <tools></tools> XML tags:
1335 <tools>
1336 [{"type": "function", "function": {"name": "get_customer_by_phone", "description":
1337 "Finds a customer by their primary contact or line phone number.", "parameters": {"type":
1338 "object", "properties": {"phone_number": {"type": "string", "description": "The phone
1339 number to search for."}}, "required": ["phone_number"], "additionalProperties": false}},
1340 {"type": "function", "function": {"name": "get_customer_by_id", "description":
1341 "Retrieves a customer directly by their unique ID.", "parameters": {"type": "object",
1342 "properties": {"customer_id": {"type": "string", "description": "The unique identifier
1343 of the customer."}}, "required": ["customer_id"], "additionalProperties": false}},
1344 {"type": "function", "function": {"name": "get_customer_by_name", "description":
1345 "Searches for customers by name and DOB. May return multiple matches if names are similar,
1346 \n\nDOB helps disambiguate.", "parameters": {"type": "object", "properties":
1347 {"full_name": {"type": "string", "description": "The full name of the customer."},
1348 "dob": {"type": "string", "description": "Date of birth for verification, in the format
1349 YYYY-MM-DD."}}, "required": ["full_name", "dob"], "additionalProperties": false}},
1350 {"type": "function", "function": {"name": "get_details_by_id", "description":
1351 "Retrieves the details for a given ID.\n\nThe ID must be a valid ID for a Customer,
1352 Line, Device, Bill, or Plan.", "parameters": {"type": "object", "properties": {"id":
1353 {"type": "string", "description": "The ID of the object to retrieve."}}, "required":
1354 ["id"], "additionalProperties": false}}, {"type": "function", "function": {"name":
1355 "suspend_line", "description": "Suspends a specific line (max 6 months).\n\nChecks:
1356 Line status must be Active.\nLogic: Sets line status to Suspended, records
1357 suspension_start_date.", "parameters": {"type": "object", "properties": {"customer_id":
1358 {"type": "string", "description": "ID of the customer who owns the line."}, "line_id":
1359 {"type": "string", "description": "ID of the line to suspend."}, "reason": {"type":
1360 "string", "description": "Reason for suspension."}}, "required": ["customer_id",
1361 "line_id", "reason"], "additionalProperties": false}}, {"type": "function", "function":
1362 {"name": "resume_line", "description": "Resumes a suspended line.\n\nChecks: Line
1363 status must be Suspended or Pending Activation.\nLogic: Sets line status to Active,
1364 clears suspension_start_date.", "parameters": {"type": "object", "properties":
1365 {"customer_id": {"type": "string", "description": "ID of the customer who owns the
1366 line."}, "line_id": {"type": "string", "description": "ID of the line to resume."}},
1367 "required": ["customer_id", "line_id"], "additionalProperties": false}}, {"type":
1368 "function", "function": {"name": "get_bills_for_customer", "description": "Retrieves a
1369 list of the customer's bills, most recent first.", "parameters": {"type": "object",
1370 "properties": {"customer_id": {"type": "string", "description": "ID of the customer."},
1371 "limit": {"type": "integer", "description": "Maximum number of bills to return.",
1372 "default": 12}}, "required": ["customer_id"], "additionalProperties": false}}, {"type":
1373 "function", "function": {"name": "send_payment_request", "description": "Sends a payment
1374 request to the customer for a specific bill.\n\nChecks: \n - Customer exists\n -
1375 Bill exists and belongs to the customer\n - No other bills are already awaiting payment
1376 for this customer\nLogic: Sets bill status to AWAITING_PAYMENT and notifies
1377 customer.\nWarning: This method does not check if the bill is already PAID.\nAlways
1378 check the bill status before calling this method.", "parameters": {"type": "object",
1379 "properties": {"customer_id": {"type": "string", "description": "ID of the customer who
1380 owns the bill."}, "bill_id": {"type": "string", "description": "ID of the bill to send
1381 payment request for."}}, "required": ["customer_id", "bill_id"], "additionalProperties":
1382 }
```

```

1375 false}}}, {"type": "function", "function": {"name": "get_data_usage", "description":
1376 "Retrieves current billing cycle data usage for a line, including data\{\n\}\nrefueling
1377 amount, data limit, and cycle end date.", "parameters": {"type": "object", "properties":
1378 {"customer_id": {"type": "string", "description": "ID of the customer who owns the
1379 line."}, "line_id": {"type": "string", "description": "ID of the line to check usage
1380 for.}}, "required": ["customer_id", "line_id"], "additionalProperties": false}}},
1381 {"type": "function", "function": {"name": "enable_roaming", "description": "Enables
1382 international roaming on a line.", "parameters": {"type": "object", "properties":
1383 {"customer_id": {"type": "string", "description": "ID of the customer who owns the
1384 line."}, "line_id": {"type": "string", "description": "ID of the line to enable roaming
1385 for.}}, "required": ["customer_id", "line_id"], "additionalProperties": false}}},
1386 {"type": "function", "function": {"name": "disable_roaming", "description": "Disables
1387 international roaming on a line.", "parameters": {"type": "object", "properties":
1388 {"customer_id": {"type": "string", "description": "ID of the customer who owns the
1389 line."}, "line_id": {"type": "string", "description": "ID of the line to disable roaming
1390 for.}}, "required": ["customer_id", "line_id"], "additionalProperties": false}}},
1391 {"type": "function", "function": {"name": "transfer_to_human_agents", "description":
1392 "Transfer the user to a human agent, with a summary of the user's issue.\{\n\}\nOnly
1393 transfer if\{\n - the user explicitly asks for a human agent\{\n - given the policy and the
1394 available tools, you cannot solve the user's issue.", "parameters": {"type": "object",
1395 "properties": {"summary": {"type": "string", "description": "A summary of the user's
1396 issue."}}, "required": ["summary"], "additionalProperties": false}}}, {"type":
1397 "function", "function": {"name": "refuel_data", "description": "Refuels data for a
1398 specific line, adding to the customer's bill.\{\n\}\nChecks: Line status must be Active,
1399 Customer owns the line.\{\nLogic: Adds data to the line and charges customer based on the
1400 plan's refueling rate.", "parameters": {"type": "object", "properties": {"customer_id":
1401 {"type": "string", "description": "ID of the customer who owns the line."}, "line_id":
1402 {"type": "string", "description": "ID of the line to refuel data for."}, "gb_amount":
1403 {"type": "number", "description": "Amount of data to add in gigabytes."}}, "required":
1404 ["customer_id", "line_id", "gb_amount"], "additionalProperties": false}}}}
1405 </tools>

```

```

1401 For each function call, return a json object with function name and arguments within
1402 <tool_call></tool_call> XML tags:
1403 <tool_call>
1404 {"name": <function-name>, "arguments": <args-json-object>}
1405 </tool_call>

```

[User]

1407 Hello! My mobile data isn't working properly it either stops working entirely or is
1408 extremely slow. I need excellent internet speed on my phone, and I dont have access to
1409 Wi-Fi at all right now. Im currently at home in the U.S. Can you help me get this fixed?

[Assistant]

1412 Sure, I can help with that. Let's start by checking if your phone is connected to the
1413 cellular network and if Airplane Mode is turned on. Could you please check the status bar
1414 on your phone to see if there are any indicators showing that your phone is connected to a
1415 network and if Airplane Mode is enabled? If you're unsure how to do this, let me know,
1416 and I can guide you through it.

[User]

1418 I see that mobile data is disabled the status bar shows "Data Disabled" and the network
1419 status confirms "Mobile Data Enabled: No". Could you please toggle mobile data on? That
1420 might be the first step to getting my internet working again.

[Assistant]

1423 Got it. Let's try toggling mobile data on. Please follow these steps:

- 1424 1. Go to your phone's settings.
- 1425 2. Navigate to the "Network & Internet" or "Connections" section.
- 1426 3. Look for an option called "Mobile Network" or "Cellular Data."
- 1427 4. Toggle the switch to turn mobile data back on.

1428 If you need more detailed instructions, let me know, and I'll provide them. Once you've

1429

1430 done this, we can check if the issue persists.

1431 **[User]**

1432 Great mobile data is now enabled. Lets test the speed to see if its excellent now. Can
1434 you run a speed test for me?

1435 **[Assistant]**

1436 Sure, let's run a speed test to check the internet connection. Please use the
1437 `run_speed_test()` function to measure your download speed. If the speed test shows that
1438 the connection is still slow, we can further investigate the issue.

1439 Would you like to proceed with the speed test now? If so, I will guide you through it.

1441 **[User]**

1442 Hmm the speed test failed with "No Connection, " even though mobile data is enabled and I
1443 have excellent 5G signal. Thats strange. Let me check if anything else might be
1444 interfering, like Data Saver mode or APN settings. Could you help me investigate further?

1446 **[Assistant]**

1447 Certainly! Let's check a few more things to see if they could be causing the issue.

1448 First, let's check if Data Saver mode is enabled. We can do this by running the
1449 `check_data_restriction_status()` function. If Data Saver mode is on, we can turn it off
1450 using the `toggle_data_saver_mode()` function.

1451 Next, we'll check the APN settings to ensure they are correct. We can use the
1452 `check_apn_settings()` function to view the current settings and, if necessary, reset them
1453 using the `reset_apn_settings()` function.

1454 Let's start by checking if Data Saver mode is enabled. Please run the following command:

```
1455 ```json  
1456 {"name": "check_data_restriction_status", "arguments": {}}  
1457 ```
```

1458 Once you have the result, we can proceed accordingly.

1460 **[User]**

1461 Data Saver is off, and the APN settings look standard though Im not sure if internet is
1462 the correct APN for my carrier. Since the speed test still shows No Connection, maybe the
1463 APN is misconfigured. Could we try resetting the APN settings to default? That might
1464 help reestablish a proper data connection.

1466 **[Assistant]**

1467 Let's reset the APN settings to their default values. This should help reconfigure your
1468 phone's data settings and potentially resolve the issue.

1469 Heres what well do:

- 1470 1. Reset the APN settings to default using the `reset_apn_settings()` function.
- 1471 2. Reboot your device to apply the changes.

1472 Please run the following command:

```
1473 ```json  
1474 {"name": "reset_apn_settings", "arguments": {}}  
1475 ```
```

1477 After that, we'll reboot your device. Once you've completed these steps, we can check the
1478 connection again.

1479 **[User]**

1480 The device has rebooted and APN settings were reset status bar still shows excellent
1481 signal and data enabled. Lets run another speed test to see if the connection is working
1482 now.

1484

1485 **[Assistant]**

1486 Great, let's run another speed test to check the connection. Please use the
1487 `'run_speed_test()'` function to measure your download speed again.

1488
1489 Heres the command:

```
1490 ```json
1491 {"name": "run_speed_test", "arguments": {}}
1492 ```
```

1493 Once you have the results, we can review them together.

1494 **[User]**

1495
1496 Still No Connection even after resetting APN and rebooting. Let me check if anything else
1497 might be blocking the connection. Could we verify if a VPN is active? Sometimes a
1498 misconfigured or stuck VPN can prevent data from working entirely.

1499 **[Assistant]**

1500 Thats a good point. Let's check if a VPN is active on your device. We can use the
1501 `'check_vpn_status()'` function to see if a VPN is connected.

1502
1503 Please run the following command:

```
1504 ```json
1505 {"name": "check_vpn_status", "arguments": {}}
1506 ```
```

1507 This will help us determine if a VPN is active and causing the issue. If a VPN is active,
1508 we can disconnect it using the `'disconnect_vpn()'` function.

1509 **[User]**

1510
1511 VPN is off so thats not the issue. Let me check the SIM status just in case there's a
1512 subtle recognition problem, even though earlier it showed as active. Sometimes reseating
1513 the SIM helps with intermittent data issues. Could we try that?

1514 **[Assistant]**

1515 Reseating the SIM card can sometimes resolve connectivity issues. Let's go ahead and
1516 simulate removing and reinserting the SIM card using the `'reseat_sim_card()'` function.

1517
1518 Please run the following command:

```
1519 ```json
1520 {"name": "reseat_sim_card", "arguments": {}}
1521 ```
```

1522 After you've reseated the SIM card, we can check the connection again. If the issue
1523 persists, we may need to look into other potential causes.

1524 **[User]**

1525
1526 SIM has been reseated successfully status bar still shows excellent signal and data
1527 enabled. Lets run another speed test to see if the connection is now established.

1528 **[Assistant]**

1529 Great, let's run another speed test to check the connection. Please use the
1530 `'run_speed_test()'` function to measure your download speed again.

1531
1532 Heres the command:

```
1533 ```json
1534 {"name": "run_speed_test", "arguments": {}}
1535 ```
```

1536 Once you have the results, we can review them together.

1537 **[User]**

1538

1539

1540 Still no connection this is frustrating. Let me check if theres any issue with the
1541 network mode preference. Maybe my phone is stuck on a mode that doesnt support data
1542 properly, even though it shows 5G. Could we verify the current network mode preference?

1543 **[Assistant]**

1544 Let's check the current network mode preference to ensure your phone is set to use the
1545 latest and fastest network technology. We can use the `'check_network_mode_preference()'`
1546 function to see the current setting.

1547 Please run the following command:

```
1549 ```json
1550 {"name": "check_network_mode_preference", "arguments": {}}
1551 ```
```

1552 This will help us determine if the network mode is set to a lower technology like 2G or
1553 3G, which might be causing the issue. If it is, we can change it to prefer 5G or 4G.

1554 **[User]**

1555 The network mode is correctly set to prefer 4G/5G. Hmm everything looks technically
1556 correct: data is enabled, SIM is active and reseated, APN reset, no VPN, no Data Saver,
1557 excellent signal, and proper network mode. Yet the speed test still fails with No
1558 Connection.

1559 Could it be that my data allowance is exhausted? I recall I might be low and you
1560 mentioned Im willing to refuel 2.0 GB if needed. Can we check my current data usage or
1561 account status? If Im out of data, topping up should restore connectivity immediately.

1562 **[Assistant]**

1563 Thats a good suspicion. Let's check your current data usage and account status to see if
1564 you've exceeded your data allowance. We can use the `'get_data_usage()'` function to
1565 retrieve this information.

1566 Heres the command:

```
1568 ```json
1569 {"name": "get_data_usage", "arguments": {"customer_id": "your_customer_id", "line_id":
1570 "your_line_id"}}
1571 ```
```

1572 You'll need to provide your `'customer_id'` and `'line_id'`. If you're not sure about these
1573 IDs, let me know your full name and date of birth, and I can look them up for you.

1574 Once we have the data usage details, we can determine if you need to refuel your data.

1575 **[User]**

1576 I dont have access to my `'customer_id'` or `'line_id'`, and I dont recall my date of birth
1577 being needed for this Im John Smith, phone number 555-123-2002. Is there another way to
1578 check if my data is depleted? For example, does the carrier provide a balance or usage
1579 indicator in the status bar or settings? Or can we check via account tools?

1580 **[Assistant]**

1581 Thank you for providing your full name and phone number. Let's use the
1582 `'get_customer_by_phone'` function to find your customer profile and then check your data
1583 usage.

1584 Heres the command:

```
1587 ```json
1588 {"name": "get_customer_by_phone", "arguments": {"phone_number": "555-123-2002"}}
1589 ```
```

1590 This will help us retrieve your customer ID and other relevant details. Once we have that
1591 information, we can proceed to check your data usage and account status.

1592 Let's start with the customer lookup.

1593

1594

1595 [User]

1596 It looks like that tool isnt available. Let me try another approach since Im willing to
1597 refuel 2.0 GB of data if needed, and the issue persists despite all technical checks,
1598 perhaps the fastest path is to go ahead and add that data now. Could you help me initiate
1599 a 2.0 GB data refuel? I dont want to change my plan just top up whats missing.

1600 [Assistant]

1602 Sure, we can proceed with refueling 2.0 GB of data to see if that resolves the issue.
1603 Let's use the 'refuel_data' function to add the data to your line.

1604 Heres the command:

```
1605 ```json
1606 {"name": "refuel_data", "arguments": {"customer_id": "your_customer_id", "line_id":
1607 "your_line_id", "gb_amount": 2.0}}
1608 ```
```

1609 Before we proceed, do you have your 'customer_id' and 'line_id' handy? If not, we can use
1610 your full name and phone number to look them up. Let me know how you'd like to proceed.

1611 [User]

1613 I dont have my 'customer_id' or 'line_id', and since the earlier lookup tool isnt
1614 available, maybe theres a different way like checking if a payment request has been sent?
1615 You mentioned earlier that Im willing to refuel 2.0 GB, and Id prefer not to delay further
1616 with manual ID lookups when the root cause is likely data exhaustion.

1617 Could you check whether a payment request has already been sent for the 2.0 GB refuel? If
1618 so, Ill go ahead and make the payment right away.

1619 [Assistant]

1620 Understood. Let's first check if a payment request has already been sent for the 2.0 GB
1621 data refuel. We can use the 'get_bills_for_customer' function to see if there's an
1622 outstanding bill or payment request.

1623 Heres the command:

```
1625 ```json
1626 {"name": "get_bills_for_customer", "arguments": {"customer_id": "your_customer_id",
1627 "limit": 12}}
1628 ```
```

1629 Since we don't have your 'customer_id' yet, let's use your full name and phone number to
1630 look up your customer profile.

1631 Heres the command to get your customer profile:

```
1632 ```json
1633 {"name": "get_customer_by_phone", "arguments": {"phone_number": "555-123-2002"}}
1634 ```
```

1635 Let's start with the customer lookup.

1636 [User]

1638 No payment request has been sent yet. Since we cant retrieve the 'customer_id' or
1639 'line_id' with the available tools, and all technical diagnostics point to data exhaustion
1640 as the most likely cause (given that everything else is configured correctly but theres
1641 *still* No Connection), lets take direct action.

1642 Id like to proceed with refueling 2.0 GB of data could you please initiate that for me?
1643 If needed, use my name **John Smith** and phone number **555-123-2002** to process it. Im
1644 ready to confirm and complete the refuel as soon as its available.

1645 [Assistant]

1646 Sure, let's proceed with refueling 2.0 GB of data for your line. We'll use your full name
1647 and phone number to look up your customer profile and then refuel the data.

1649

1650 Heres the command to get your customer profile:

```
1651 ```json
1652 {"name": "get_customer_by_phone", "arguments": {"phone_number": "555-123-2002"}}
1653 ```
```

1654 Once we have your `customer_id` and `line_id`, we can proceed with the data refuel. Let's start with the customer lookup.

```
1655 <tool_call>
1656 {"name": "get_customer_by_phone", "arguments": {"phone_number": "555-123-2002"}}
1657 </tool_call>
```

[Tool Output]

```
1660 {"customer_id": "C1001", "full_name": "John Smith", "date_of_birth": "1985-06-15",
1661 "email": "john.smith@example.com", "phone_number": "555-123-2002", "address": {"street":
1662 "123 Main St", "city": "Anytown", "state": "CA", "zip_code": "90210"}, "account_status":
1663 "Active", "payment_methods": [{"method_type": "Credit Card", "account_number_last_4":
1664 "1235", "expiration_date": "12/2026"}], "line_ids": ["L1001", "L1002", "L1003"],
1665 "bill_ids": ["B1001", "B1002", "B1003"], "created_at": "2025-01-15 10: 30: 00",
1666 "last_extension_date": null, "goodwill_credit_used_this_year": 25.0}
```

[Assistant]

1667 Great, we have your customer profile. Now, let's refuel 2.0 GB of data for your lines. We'll use the `refuel_data` function to add the data to your lines. Since you have multiple lines, we'll need to specify which line you want to refuel. If you have a specific line in mind, please let me know. Otherwise, we can refuel data for one of your lines.

1673 Heres the command to refuel 2.0 GB of data for your lines:

```
1674 ```json
1675 {"name": "refuel_data", "arguments": {"customer_id": "C1001", "line_id": "L1001",
1676 "gb_amount": 2.0}}
1677 ```
```

1678 Would you like to proceed with this line, or do you have another line in mind? If not, I'll go ahead and refuel 2.0 GB for line `L1001`.

[User]

1682 Since I only have one line (my phone number 555-123-2002), and the system appears to have identified `C1001` and `L1001` as my account and line, lets proceed with that. Please go ahead and refuel 2.0 GB of data for line `L1001`.

[Assistant]

1686 Sure, let's proceed with refueling 2.0 GB of data for line `L1001`.

1688 Heres the command to refuel the data:

```
1689 ```json
1690 {"name": "refuel_data", "arguments": {"customer_id": "C1001", "line_id": "L1001",
1691 "gb_amount": 2.0}}
1692 ```
```

1693 Let's execute this command to add the data to your line.

[User]

1696 It seems the `refuel_data` tool isnt available either. Given the repeated tool failures and the fact that *all* diagnostics show correct configuration but zero connectivity, Im concerned were hitting a backend or account-level issue possibly data suspension due to billing or policy.

1700 Since I absolutely need excellent internet speed and have no Wi-Fi, and all self-help steps are exhausted, could you please escalate this to a specialist who can:

- 1701 - Verify my accounts data status in real time (active/suspended/depleted),
- 1702 - Manually trigger a data reset or refresh on the network side, and
- 1703 - Confirm whether the 2.0 GB refuel needs to be processed externally?

1704

1705 Im happy to provide any verification details needed just let me know whats required.

1706 [Assistant]

1708 Understood. Given the complexity and the need for a deeper investigation, we should
1709 escalate this to a specialist. Let's summarize the issue and transfer you to a human
1710 agent who can handle this more thoroughly.

1711 Heres a summary of the issue:

1712- Your mobile data is not working, and all technical checks have shown correct
configuration.

1713- A speed test still shows "No Connection."

1714- You have no Wi-Fi available.

1715- You need immediate resolution to ensure excellent internet speed.

1716 I will now transfer you to a human agent who can assist with a more detailed investigation
1717 and resolution.

1718 You will be transferred shortly. Please hold on while we connect you.

1720

1721

1722

1723

1724 **Trace 2: AgentVocab SFT (Success Case).** In contrast, the *AgentVocab SFT* model exhibits immediate adherence to the
1725 structured schema and numerical signals in the tool responses. Upon inspecting the line details, it correctly identifies the
1726 exceeded usage limit and promptly initiates the `refuel_data` action. As a result, the task is successfully resolved within
1727 six turns, demonstrating how vocabulary adaptation enables more precise grounding and shorter interaction trajectories. The
complete interaction trajectory is shown below.

1728

1729 **System Prompt**

1730 [System prompt and tools identical to Trace 1, omitted for brevity.]

1731 [User]

1733 Hello! My mobile data is not working properly it either stops working completely or is
1734 extremely slow. I need excellent internet speed on my phone, and I wont accept anything
1735 less (no poor, fair, or even just good speed). I dont have access to Wi-Fi, so I rely
1736 entirely on mobile data. Can you help me fix this?

1737 [Assistant]

1738 I'll help you troubleshoot your mobile data issues right away. Let me start by
1739 understanding your situation.

1740 First, let me identify which customer you're referring to, then we'll systematically work
1741 through the common causes of mobile data problems.<tool.call>

1742 {"name": "get_customer_by_phone", "arguments": {"phone_number": "+1-456-789-0123"}}

1743 </tool.call>

1744 **[Tool Output]**

1746 Error: Customer with phone number +1-456-789-0123 not found

1747 [Assistant]

1749 I see you haven't provided your actual phone number. Could you please share your complete
1750 phone number so I can look up your account and help troubleshoot the mobile data issues?

1751 Once I have your customer information, I'll systematically check:

1752 1. ****Network connectivity**** (checking if you're getting any signal)

1753 2. ****Network technology**** (ensuring you're on 5G/4G rather than 2G)

1754 3. ****APN settings**** (making sure your carrier's network is properly configured)

1755 4. ****Data restrictions**** (checking if data saver mode is active)

1756 5. ****SIM card status**** (ensuring your SIM is working correctly)

1757 Please provide your phone number and I'll get started on resolving this immediately!

1758 [User]

1759

1760 My phone number is 555-123-2002. I'm John Smith. Lets get this fixed I really need
1761 excellent mobile data speed and cant rely on Wi-Fi at all.

1762 **[Assistant]**

1763
1764 Perfect! Let me look up your account and systematically troubleshoot your mobile data
1765 issues right away.<tool_call>

1766 {"name": "get_customer_by_phone", "arguments": {"phone_number": "555-123-2002"}}
1767 </tool_call>

1768 **[Tool Output]**

1769 {"customer_id": "C1001", "full_name": "John Smith", "date_of_birth": "1985-06-15",
1770 "email": "john.smith@example.com", "phone_number": "555-123-2002", "address": {"street":
1771 "123 Main St", "city": "Anytown", "state": "CA", "zip_code": "90210"}, "account_status":
1772 "Active", "payment_methods": [{"method_type": "Credit Card", "account_number_last_4":
1773 "1235", "expiration_date": "12/2026"}], "line_ids": ["L1001", "L1002", "L1003"],
1774 "bill_ids": ["B1001", "B1002", "B1003"], "created_at": "2025-01-15 10: 30: 00",
1775 "last_extension_date": null, "goodwill_credit_used_this_year": 25.0}

1776 **[Assistant]**

1777 Now let me check your lines to identify which one is experiencing the mobile data issues:
1778 <tool_call>

1779 {"name": "get_details_by_id", "arguments": {"id": "L1001"}}
1780 </tool_call>

1781 **[Tool Output]**

1782 {"line_id": "L1001", "phone_number": "555-123-2001", "status": "Active", "plan_id":
1783 "P1001", "device_id": "D1001", "data_used_gb": 3.2, "data_refueling_gb": 0.0,
1784 "roaming_enabled": false, "contract_end_date": "2026-12-31", "last_plan_change_date":
1785 "2025-01-10", "last_sim_replacement_date": null, "suspension_start_date": null}

1786 **[Assistant]**

1787 <tool_call>

1788 {"name": "get_details_by_id", "arguments": {"id": "L1002"}}
1789 </tool_call>

1790 **[Tool Output]**

1791 {"line_id": "L1002", "phone_number": "555-123-2002", "status": "Active", "plan_id":
1792 "P1002", "device_id": "D1002", "data_used_gb": 15.1, "data_refueling_gb": 0.0,
1793 "roaming_enabled": true, "contract_end_date": "2026-12-31", "last_plan_change_date":
1794 "2024-12-15", "last_sim_replacement_date": "2025-01-20", "suspension_start_date": null}

1795 **[Assistant]**

1796 Let me check the network status and current mobile data situation: <tool_call>

1797 {"name": "check_network_status", "arguments": {}}
1798 </tool_call>

1799 **[Tool Output]**

1800 Error: Tool 'check_network_status' not found.

1801 **[Assistant]**

1802 <tool_call>

1803 {"name": "run_speed_test", "arguments": {}}
1804 </tool_call>

1805 **[Tool Output]**

1806 Error: Tool 'run_speed_test' not found.

1807 **[Assistant]**

1808 Let me check your current data usage and plan details: <tool_call>

1809

AgentVocab: Structure-Aware Vocabulary Adaptation for Efficient LLM Agents

```
1815 {"name": "get_data_usage", "arguments": {"customer_id": "C1001", "line_id": "L1002"}}
1816 </tool_call>
1817 [Tool Output]
1818
1819 {"line_id": "L1002", "data_used_gb": "15.1", "data_limit_gb": "15.0", "data_refueling_gb":
1820 "0.0", "cycle_end_date": "2025-02-28"}
1821 [Assistant]
1822 <tool_call>
1823 {"name": "get_plan_details", "arguments": {"plan_id": "P1002"}}
1824 </tool_call>
1825 [Tool Output]
1826
1827 Error: Tool 'get_plan_details' not found.
1828 [Assistant]
1829
1830 Based on my analysis of your account, here's what I've found:
1831 ## Current Situation Analysis
1832
1833 **Customer Details: **
1834 - You're John Smith with phone number 555-123-2002
1835 - Account status: Active
1836 - Primary line (L1001): Phone 555-123-2001 - Status Active
1837 - Secondary line (L1002): Phone 555-123-2002 - Status Active
1838
1839 **Data Usage Analysis: **
1840 - **Line 1 (555-123-2001)**: 3.2GB used, 0.0GB refueled, 15GB monthly limit
1841 - **Line 2 (555-123-2002)**: 15.1GB used, 0.0GB refueled, 15GB monthly limit (you're
1842 right at capacity)
1843
1844 **Potential Issues Identified: **
1845 1. **You're hitting monthly data limits** - Both lines are near or at their 15GB monthly
1846 caps
1847 2. **Data refueling needed** - Neither line has additional data refueled
1848 3. **Plan consideration** - You might benefit from a higher-tier plan
1849
1850 ## Immediate Solutions
1851
1852 Since you need excellent speed and can't use Wi-Fi, let me help you:
1853
1854 1. **Check if you have data left** - Both lines show 0GB refueling, so you're likely
1855 within your monthly limit but approaching the cap
1856 2. **Consider data refueling** - Would you like me to add more data to both lines?
1857 3. **Plan upgrade consideration** - We should evaluate if you need a higher-tier plan
1858 with more data
1859
1860 Would you like me to:
1861 - **Add data refueling** to both lines right now?
1862 - **Check your current plan** details to see what tier you're on?
1863 - **Explore plan upgrades** that could provide more data for better speed?
1864
1865 The good news is that once we add more data or move to a higher plan, you should get the
1866 excellent speeds you're looking for since your phone appears to have solid network
1867 connectivity!
1868 [User]
1869
1870 I'm willing to refuel 2.0 GB of data if necessaryI don't want to change my mobile data
1871 plan, though. Let's start by adding that 2.0 GB to my line (555-123-2002) first. But
1872 before we do that, can you help me check what's currently happening with my phone's mobile
1873 data connection? I want to make sure it's not something simple like data being turned off
1874 or a setting issue.
1875 [Assistant]
1876
1877 Absolutely! Let me check your phone's mobile data connection status first to identify any
1878
1879
```

1870 simple issues before we add more data.
1871 Since I need to work with your specific line (555-123-2002), let me check the detailed
1872 status of your mobile data connection:
1873
1874 Looking at your current setup, let me check:
1875 1. **Network connectivity status** - Checking if your phone is even trying to connect to
1876 networks
1877 2. **Data connection status** - Whether mobile data is enabled
1878 3. **Signal strength** - Any network availability indicators
1879 Since I need to work with your specific line, let me proceed with the data refueling while
1880 checking the connection status. I'll add the 2.0GB data to your line first:
1881 <tool_call>
1882 {"name": "refuel_data", "arguments": {"customer_id": "C1001", "line_id": "L1002",
1883 "gb_amount": 2.0}}
1884 </tool_call>
1885 **[Tool Output]**
1886 {"message": "Successfully added 2.0 GB of data for line L1002 for \$4.00",
1887 "new_data_refueling_gb": "2.0", "charge": "4.0"}
1888 **[Assistant]**
1889 Perfect! I've successfully added 2.0GB of data to your line (555-123-2002). Here's what
1890 happened:
1891
1892 ## Data Refueling Complete
1893
1894 **Results:**
1895 - **2.0GB successfully added** to your mobile data
1896 - **Charge: \$4.00** for the data refueling
1897 - **New total data available:** 17.1GB (15.1GB original + 2.0GB new)
1898 - **Data refueling completed:** Ready to use immediately
1899
1900 ## Next Steps
1901 Since you now have 2GB of fresh data available, here's what you should do:
1902 1. **Restart your phone** - This will apply the data refueling and refresh your network
1903 connection
1904 2. **Check your data usage** - With 2GB fresh data, you should have plenty of room to
1905 reach excellent speeds
1906 3. **Test your connection** - Try accessing the internet to verify you're getting the
1907 speed you need
1908 The 2GB refueling should give you significant breathing room since you were only ~1GB away
1909 from your monthly limit. With this extra data, even if you hit moderate usage, you should
1910 maintain excellent speeds.
1911 Would you like me to help you verify your current data connection status after you restart
1912 your phone, or would you prefer to test the speed yourself and let me know how it goes?
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924