

TokenCliff: Benchmarking Output-Token Budget Sensitivity in Multi-Turn LLM Agents

Bairui Zhang¹ Weixuan Liu² Defan Xue³ Yongqi Zhang^{1,*}

¹The Hong Kong University of Science and Technology (Guangzhou)

²East China Normal University

³Beijing University of Posts and Telecommunications

barry.zhang@connect.hkust-gz.edu.cn, yongqizhang@hkust-gz.edu.cn

Abstract

Output-token budgets are a first-order deployment constraint for large language model (LLM) agents, yet existing benchmarks typically evaluate them under a single fixed or generous budget. We introduce **TokenCliff**, a benchmark for measuring multi-turn agent performance across episode-level output-token budgets. It contains 660 deterministic tasks spanning four capability-oriented domains—file operations, data transformation, tool use, and planning—plus a compositional multi-step split. We propose two budget-sensitivity metrics: **Budget Elasticity**, which captures robustness across budget levels, and the **Cliff Index**, which captures the sharpest adjacent-budget performance drop. Evaluating 10 LLMs at five budget levels from 200 to 8,192 tokens yields 33,000 episodes. Results show that (1) accuracy drops by up to 32 percentage points between adjacent budgets; (2) neither peak performance nor larger model size necessarily implies stronger budget elasticity; and (3) budget sensitivity is substantially greater in the evaluated multi-turn setting than in the single-shot diagnostic. Code and data are available in our public repository [here](#) .

1 Introduction

Large language model (LLM) agents solve tasks over multiple turns by planning, calling tools, observing results, and revising their decisions before producing a final answer (Liu et al., 2026b; McCleary and Ghawaly, 2026). These characteristics drive the evaluation of multi-turn reasoning performance across various domains, such as software engineering and web search (Fan et al., 2025; Liu et al., 2026b). Despite their effectiveness, deploying LLM agents in real-world scenarios is often hindered by the massive token consumption, which poses a significant financial burden on users. However, current evaluations mainly focus on measur-

ing peak performance under a fixed inference configuration with a generous or unspecified budget. This practice, however, obscures a critical deployment question: **how does an agent behave when the available output-token budget is tightened?**

Existing studies of agents under computational or token-budget constraints can be broadly grouped into four categories: cost- and efficiency-oriented agent evaluation (Kapoor et al., 2025; Fan et al., 2025; Tripathy et al., 2026); budget-aware reasoning evaluation and token-efficient reasoning methods (Wang et al., 2024; Han et al., 2025); explicit tool-call budgets (Liu et al., 2026b) and joint tool-call and completion-token constraints (McCleary and Ghawaly, 2026); and budget-aware inference methods spanning reasoning-token control, monetary tool-use constraints, search-time resource allocation, and context management (Xu et al., 2026; Liu et al., 2026a; Li et al., 2026b; Fang et al., 2026; Wu et al., 2026). These studies motivate a complementary evaluation setting in which a shared episode-level output-token cap is swept across complete multi-turn trajectories while tasks, environments, interaction protocols, and scoring are held fixed.

To bridge this gap, we introduce **TokenCliff**, a benchmark for evaluating how multi-turn reasoning performance of LLM agents varies under different output-token budgets. Specifically, TokenCliff comprises 660 deterministic tasks across four capability-oriented domains—file operations, data transformation, tool use, and planning—plus a compositional Multi-Step split. Each task is evaluated under five total output-token budgets for the entire multi-turn interaction: 200, 500, 1,500, 4,000, and 8,192 tokens. By measuring performance at each budget level, we obtain a performance-budget curve for each model. We further introduce two budget-sensitivity metrics, **budget elasticity** and the **Cliff Index**, to quantify both the overall robustness of LLM agents under tighter output-token

*Corresponding author.

budgets and the largest abrupt performance drop between adjacent budget levels.

We evaluate 10 LLMs on 660 tasks under five output-token budget levels, resulting in 33,000 evaluation episodes. Each task is executed in a self-contained deterministic environment. Tasks with hidden environment state expose it through tools, while some planning and chained-computation tasks operate directly on constraints provided in the task description. Given the task description, available tools, environment metadata, and token budget, the agent interacts with the environment through a fixed action interface: THINK for planning, USE for tool calls, and SUBMIT for the final answer. Experiments reveal three key findings: LLM agents can suffer performance cliffs of up to 32 percentage points; peak performance does not necessarily imply strong budget elasticity; and performance cliffs are substantially more pronounced in multi-turn settings than in single-shot settings.

In summary, our contributions are as follows:

- We introduce **TokenCliff**, a benchmark of 660 deterministic tasks across four capability-oriented domains and a compositional Multi-Step split for evaluating LLM agents under explicit output-token budgets.
- We develop a controlled evaluation protocol that runs each task at five budget levels and traces a performance-budget curve for each model.
- We propose two budget-sensitivity metrics, **budget elasticity** and the **Cliff Index**, to measure both overall robustness to tighter budgets and the largest performance drop between adjacent budget levels.

2 Related Work

Existing evaluations of LLM agents span diverse tasks and resource considerations, including software engineering (Fan et al., 2025), web search (Liu et al., 2026b), energy-aware software issue resolution (Tripathy et al., 2026), and long-horizon real-world autonomous tasks (Li et al., 2026a). In addition to task success, recent studies emphasize cost and resource consumption (Kapoor et al., 2025; Fan et al., 2025), while Kapoor et al. (Kapoor et al., 2025) also highlight reproducibility and benchmark-design issues relevant to real-world deployment. For instance, AgencyBench (Li

et al., 2026a) evaluates agents in 1M-token real-world scenarios and shows that models can differ substantially in their resource efficiency. TokenCliff complements these benchmarks by focusing on a different question: how multi-turn agents behave when their token budget changes. Instead of adding another real-world domain, TokenCliff uses controlled and deterministic environments, making the budget itself the main experimental variable.

Recent work has shown growing interest in evaluating LLM agents under explicit resource constraints, highlighting the importance of resource-aware evaluation in practical settings. These constraints are often defined in terms of tool usage, remaining budget, or overall deployment cost. For example, BCAS (McCleary and Ghawaly, 2026) exposes the remaining budget to RAG agents and uses it to decide when agents should call external tools. BATS (Liu et al., 2026b) studies web-search agents under explicit tool-call budgets, measuring how agent performance scales as the number of allowed search actions changes. Other work studies budget allocation during inference (Fang et al., 2026) or analyzes the system-level cost of scaling agentic reasoning, showing diminishing returns as test-time compute increases (Kim et al., 2026).

However, existing budget-aware agent evaluations often focus on tool-call limits, system or monetary budgets, context/resource allocation, or post-hoc cost measurements. TokenCliff instead isolates the episode-level output-token budget and systematically varies it while holding the remaining evaluation protocol fixed. Instead of introducing a new task domain, TokenCliff evaluates how agents’ multi-turn performance varies across token budgets and models.

3 Curation of TokenCliff

3.1 Task Definition

Let a task be $x = (I, E, \mathcal{T}, V)$, where I is the instruction, E is a deterministic environment with hidden state, $\mathcal{T}$ is the set of tools exposed to the agent, and V is a deterministic verifier. A model m interacts with E over multiple turns by emitting visible actions and receiving observations after tool calls. Given an output-token budget b , the episode ends when the model submits an answer, reaches the turn limit, or exhausts the budget. The binary outcome is

$$r(m, x, b) \in \{0, 1\},$$

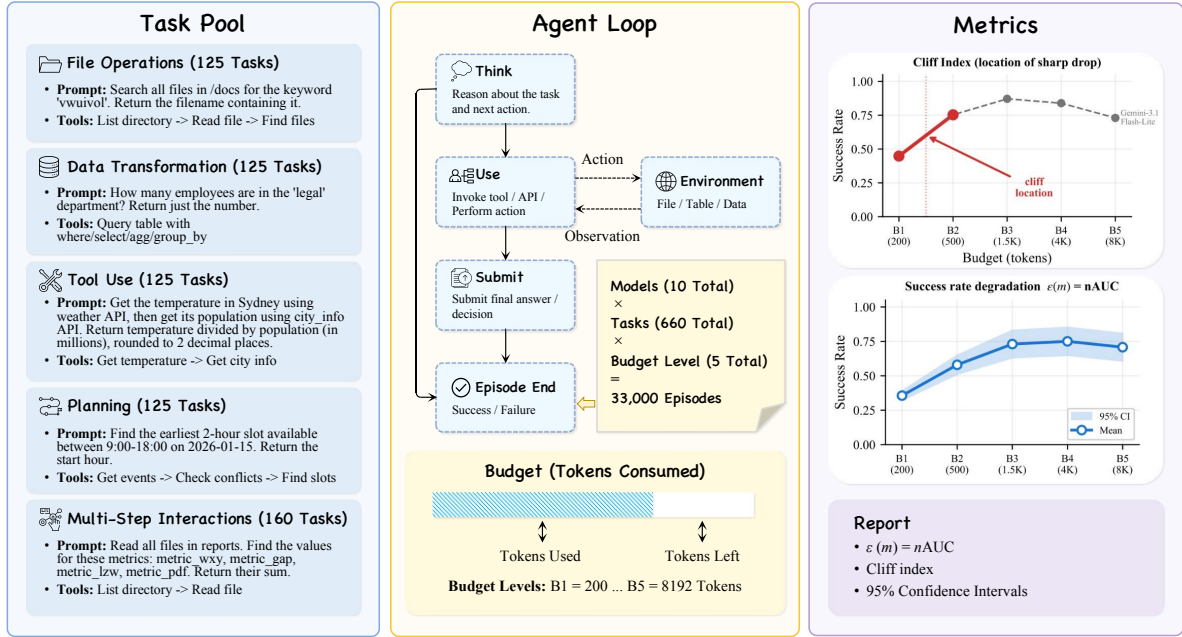


Figure 1: Overview of TokenCliff. The benchmark fixes tasks, environments, prompts, action format, and scoring, then varies only the output-token budget. Each model is evaluated across 660 tasks and 5 budget levels to produce a success-budget curve summarized by budget elasticity ϵ and Cliff Index.

where 1 indicates that the submitted answer is accepted by V . For a task distribution $\mathcal{D}$ and budget levels $B = \{b_1, \dots, b_K\}$, the **budget-response function** is

$$s(m, b) = \mathbb{E}_{x \sim \mathcal{D}}[r(m, x, b)].$$

TokenCliff treats b as the experimental variable of interest; the task instance, environment, prompt, action format, and verifier are held fixed. Environments and verifiers are deterministic and do not use LLM judges, which reduces scoring noise when comparing budgets.

Figure 1 illustrates the overall design. TokenCliff contains four capability-oriented domains: **File Operations** (125 tasks; navigating virtual filesystems and aggregating file content), **Data Transformation** (125; filtering, aggregation, and joins), **Tool Use** (125; sequential and conditional API calls), and **Planning** (125; scheduling, optimization, and dependency resolution). A separate compositional **Multi-Step** split (160 tasks) extends file, data-processing, and chained-reasoning tasks with larger inputs or longer operation sequences. All environments are self-contained Python simulations: given the same inputs, each environment returns the same observations, and no external services are called. Representative examples are in Appendix E.

The corpus is generated deterministically in two stages. First, a base generator produces 500 synthetic tasks spanning the first four domains (File Operations, Data Transformation, Tool Use, Planning), with 125 tasks each. For every task, the generator samples an environment state (e.g., virtual files, database tables, or API endpoints populated with seed-controlled randomized content) and then programmatically derives the ground-truth answer from that environment. A second generator produces 160 Multi-Step tasks: large-table filtering and aggregation (50 tasks; 30–59 rows), multi-file extraction and aggregation (50; 8–15 files), and chained arithmetic computations (60; 4–6 operations). All tasks are evaluated using exact-match or substring-match metrics against the programmatically generated ground truth. Each task is serialized as a self-contained JSON file containing the prompt, available tools, complete environment state, ground truth, and difficulty label. For tool-mediated tasks, the complete environment state remains hidden from the agent and is exposed only through the specified interfaces, while deterministic verifiers access it for scoring. The released 660 JSON files constitute the canonical benchmark and can be deterministically regenerated from the provided scripts with identical outputs (see Appendix L for the generation pseudocode).

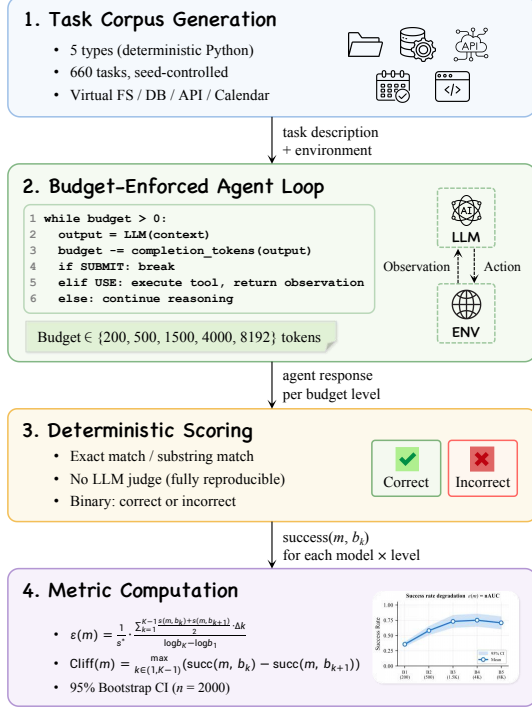


Figure 2: Budget-capped agent loop. The task suite contains four capability-oriented domains plus a compositional Multi-Step split. At each turn, the model may produce visible planning followed by USE or SUBMIT; after an observation, it may think again, call another tool, or submit. All generated output tokens count against one episode-level budget.

3.2 Evaluation Protocol and Budget Sensitivity Metrics

As shown in Figure 2, each episode runs the following loop:

- The agent receives the task description and available tools. For tasks with hidden state, it receives environment metadata rather than the raw state; reasoning-only tasks receive their constraints in the task description.
- The prompt requests exactly one marker per generation: THINK: for visible planning, USE: tool(args) for a tool call, or SUBMIT: answer for the final answer. Some models nevertheless emit planning followed by an action in one generation. The evaluator therefore processes SUBMIT if present, otherwise executes the first well-formed USE, and otherwise continues on THINK. A USE action sends arguments to the environment, which executes the tool and returns an observation.
- The loop terminates when the agent emits SUBMIT, when the episode budget is ex-

hausted, or when 20 turns have elapsed.

- The episode budget caps total **output tokens across all turns**. Let b be the episode budget and c_τ the API-reported completion tokens used at turn τ . Before turn t , the evaluator computes the remainder $r_t = b - \sum_{\tau < t} c_\tau$ and sets $\text{max_tokens}_t = \min(r_t, 500)$, where 500 is the fixed per-turn cap across all models and budgets; this prevents one generation from consuming the entire episode budget while keeping the per-turn constraint constant. Thus, the cap is enforced during generation rather than by truncating a completed trajectory. Two fallback rules apply when an episode ends without a well-formed SUBMIT: (i) after budget exhaustion or the turn limit, the last model output is scored as the answer; (ii) an output matching none of THINK, USE, or SUBMIT is scored as the final answer.

The system prompt states the total episode budget, and the evaluator provides an updated remaining-budget estimate before each subsequent generation. The complete template is provided in Appendix M.

We evaluate five increasing output-token budgets on an approximately logarithmic scale, as shown in Table 1. The grid covers an extreme setting where only one or two short turns are possible, intermediate settings that allow limited exploration, and a high-budget condition that is the least restrictive setting in our protocol.

Level	Tokens	Design Intent
B1 (Extreme)	200	Very limited interaction
B2 (Tight)	500	Limited multi-turn interaction
B3 (Moderate)	1,500	Extended exploration
B4 (Comfortable)	4,000	Room for backtracking
B5 (High)	8,192	Least restrictive setting

Table 1: Budget levels. Budget = total output tokens across all agent turns.

For model m evaluated at the ordered budgets $B = \{b_1, \dots, b_K\}$ ($K = 5$ here), $s(m, b_k)$ denotes its mean task success at the k -th budget. We define budget elasticity as the **log-budget trapezoidal area under the success-budget curve, normalized by peak performance**:

$$\varepsilon(m) = \frac{1}{s^*} \cdot \frac{\sum_{k=1}^{K-1} \frac{s(m, b_k) + s(m, b_{k+1})}{2} \cdot \Delta_k}{\log b_K - \log b_1}, \quad (1)$$

where $s^* = \max_k s(m, b_k)$ and $\Delta_k = \log b_{k+1} - \log b_k$. The log-scale integration weights budget transitions proportionally to their multiplicative factor, preventing high-budget levels from dominating. Normalizing by peak makes ε a relative curve-shape metric rather than an absolute capability score: a model maintaining 80% of its peak scores $\varepsilon = 0.8$ regardless of absolute capability. When $s^* = 0$, we define $\varepsilon(m) = 0$. We also report the unnormalized log-budget AUC, $A_{\log}(m) = s^* \varepsilon(m)$, as absolute budget-averaged capability.

The **Cliff Index** measures the largest single-step performance gain between adjacent budget levels (equivalently, the steepest drop when budget is reduced):

$$\text{Cliff}(m) = \max_{1 \leq k < K} (s(m, b_{k+1}) - s(m, b_k)), \quad (2)$$

where $b_1 < b_2 < \dots < b_K$. A high Cliff Index indicates that most performance loss is concentrated at one observed adjacent budget boundary; “Cliff Loc.” denotes the transition attaining this maximum.

4 Experiments and Results

4.1 Experimental Design

We organize the experiments around four questions:

- Do output-token budgets induce sharp performance cliffs in multi-turn agents?
- Does peak task performance predict budget elasticity?
- Are budget cliffs specific to multi-turn interaction rather than single-shot truncation?
- Which domains or task complexities amplify budget sensitivity?

We evaluate 10 models spanning open and proprietary model families, smaller and larger variants, and multiple API providers; full model metadata and token-accounting settings are reported in Table 7 of Appendix C. Unless otherwise stated, all experiments use the full 660-task TokenCliff suite described in Section 3. We denote the five output-token budgets as B1–B5; the budget is counted over all model-generated output tokens across turns. The full evaluation contains 660 tasks $\times$ 5 budget levels $\times$ 10 models = 33,000 agent episodes. We report 95% bootstrap confidence intervals from 2,000

independent within-budget resamples for key metrics and representative budget levels (Table 10 in Appendix G). All models use the same protocol prompt, the same action format, temperature 0 to minimize sampling variance, and deterministic environment seeds.

4.2 Budget-Response Results

Table 2 presents the full budget-response results. Success rates range from 0.271 (Qwen2.5-7B at B1) to 0.973 (Qwen3.5-397B at B4), while budget elasticity ε ranges from 0.794 (DeepSeek-V3) to 0.919 (Gemini-3.1-Flash-Lite). The table is sorted by ε so that the difference between peak capability and graceful degradation is visible. We additionally report $A_{\log}$ to preserve absolute budget-averaged capability. All models drop substantially at B1, with no model exceeding 57% success under the 200-token budget. Cliff Index values range from 0.080 to 0.320, and 9 of 10 models cliff at the B1→B2 transition.

Budget constraints produce large cliffs. Across 10 models on 660 tasks, Qwen3.5-397B exhibits the largest Cliff Index of 0.320 (B2: 71% → B1: 39%), followed by DeepSeek-V3 at 0.300 (B3: 86% → B2: 56%) and Gemini-3.1-Flash-Lite at 0.262 (B2: 83% → B1: 57%). These drops show that agent performance can change abruptly between adjacent budget levels rather than degrading smoothly.

Peak performance is not a sufficient proxy for elasticity. Gemini-3.1-Flash-Lite ($\varepsilon=0.919$, CI [0.899, 0.926]), Qwen2.5-72B ($\varepsilon=0.910$, CI [0.859, 0.923]), and GPT-4o-mini ($\varepsilon=0.900$, CI [0.859, 0.913]) top the elasticity ranking despite lower peak accuracy than Qwen3.6-Plus (peak=0.961, $\varepsilon=0.839$) or Qwen3.5-397B (peak=0.973, $\varepsilon=0.833$). The top differences have overlapping CIs, but the leading models remain well above DeepSeek-V3 ($\varepsilon=0.794$, CI [0.771, 0.807]). Within the Qwen3.5 MoE family, the larger 397B model ($\varepsilon=0.833$) is less elastic than the smaller 35B ($\varepsilon=0.843$). Across 10 models, peak performance and ε have Spearman $\rho = -0.462$ ($p = 0.179$). Given the small model sample, we treat this as a descriptive trend: peak performance alone is not a sufficient proxy for elasticity in this model set.

Cliff location as a behavioral diagnostic. Nine of ten models cliff at B1→B2, making 200→500 tokens the observed adjacent boundary with the

Model	Success Rate by Budget Level					Metrics			Cliff Loc.
	B1 (200)	B2 (500)	B3 (1.5K)	B4 (4K)	B5 (8K)	ϵ	$A_{\log}$	Cliff	
Gemini-3.1-Flash-Lite	0.567	0.829	0.936	0.932	0.932	0.919	0.860	0.262	B1→B2
Qwen2.5-72B	0.353	0.532	0.605	0.609	0.597	0.910	0.554	0.179	B1→B2
GPT-4o-mini	0.388	0.636	0.723	0.735	0.727	0.900	0.661	0.248	B1→B2
Claude-Haiku-4.5	0.477	0.724	0.838	0.833	0.850	0.897	0.763	0.247	B1→B2
Qwen2.5-7B	0.271	0.352	0.359	0.405	0.368	0.884	0.357	0.080	B1→B2
Qwen3.6-Flash	0.374	0.565	0.753	0.733	0.758	0.860	0.651	0.191	B1→B2
Qwen3.5-35B	0.374	0.564	0.695	0.752	0.758	0.843	0.639	0.189	B1→B2
Qwen3.6-Plus	0.426	0.680	0.915	0.961	0.958	0.839	0.806	0.255	B1→B2
Qwen3.5-397B	0.391	0.711	0.906	0.973	0.964	0.833	0.810	0.320	B1→B2
DeepSeek-V3	0.300	0.561	0.861	0.918	0.917	0.794	0.729	0.300	B2→B3

Table 2: Main results: 10 models evaluated on 660 tasks across 5 budget levels. B1–B5 are total episode-level output-token budgets. ϵ is peak-normalized log-budget AUC (relative retention); $A_{\log} = s^*\epsilon$ is unnormalized log-budget AUC; Cliff is the largest adjacent-level gain, and Cliff Loc. is its budget transition. Metrics are computed from unrounded success rates, so differences derived from displayed values may vary by 0.001. Models are sorted by ϵ ; bold denotes the maximum in each numerical column.

largest change for most models. DeepSeek-V3 instead cliffs at B2→B3 (0.300), showing continued budget sensitivity beyond 500 tokens.

4.3 Interaction and Task-Level Effects

Multi-turn vs. single-shot reasoning. As a diagnostic, we compare the agent loop with a single-shot setting on the same 500 non-multistep tasks. Single-shot mode provides the full environment state and elicits one answer, so it also changes information access and prompt composition. Table 3 shows smaller single-shot cliffs (0.006–0.018) than agent-loop cliffs (0.084–0.236); paired bootstrap intervals for the ϵ gap exclude zero for all three models. Thus, budget sensitivity is amplified in the evaluated multi-turn setting rather than explained solely by one-shot output length. Full per-budget single-shot results are reported in Table 8 of Appendix D.

Model	Single-Shot		Agent Loop	
	ϵ	Cliff	ϵ	Cliff
GPT-4o-mini	0.991	0.018	0.887	0.236
Qwen2.5-72B	0.977	0.012	0.917	0.136
Qwen2.5-7B	0.989	0.006	0.884	0.084

Table 3: Diagnostic on the same 500 non-multistep tasks. Single-shot ϵ is near 1.0, whereas the matched agent-loop curves have lower ϵ and larger cliffs.

Domain-level effects. Table 4 breaks down ϵ by task domain.

Across models, file operations is the lowest-elasticity domain for 6 of 10 models, data transformation for 3, and planning for 1. Qwen3.6-Flash

Model	Data	File	Tool	Plan	Multi
Gemini-3.1-FL	.857	.865	.964	.886	.961
Qwen2.5-72B	.858	.901	.887	.900	.883
GPT-4o-mini	.889	.872	.925	.817	.938
Claude-Haiku-4.5	.901	.729	.938	.876	.915
Qwen2.5-7B	.822	.874	.857	.925	.881
Qwen3.6-Flash	.858	.658	.924	.796	.883
Qwen3.5-35B	.813	.756	.905	.774	.880
Qwen3.6-Plus	.763	.750	.941	.842	.853
Qwen3.5-397B	.807	.792	.866	.841	.849
DeepSeek-V3	.716	.704	.877	.801	.833

Table 4: Per-domain ϵ computed over B1–B5 (200, 500, 1,500, 4,000, and 8,192 episode-level output tokens) on four capability-oriented domains and the compositional Multi-Step split.

shows the largest cross-domain variation (population std = 0.093), followed by Claude-Haiku-4.5 (0.074), indicating that budget sensitivity can vary substantially by task type. Among models with tool_use data, Gemini-3.1-Flash-Lite achieves the highest tool_use elasticity (0.964). The relatively high Multi-Step ϵ values indicate stronger retention relative to each model’s domain-specific peak. Multi-Step is a compositional split rather than an ordinal difficulty tier. Table 9 in Appendix F reports complementary domain-level performance statistics.

Multi-step diagnostic. As a supplementary diagnostic, we extract the 50 large-table tasks (multistep_001–050), which form one subset of the 160-task Multi-Step split, from the main 33,000-episode evaluation. Table 5 reports all five budgets for three models. Qwen3.5-35B’s Cliff in-

creases from 0.189 on the full benchmark to 0.360 on this subset. Details are in Appendix A.

5 Diagnostics and Robustness

The results above show that budget-response curves vary across models and differ more strongly in the agent loop than in the single-shot diagnostic. Figure 3 presents representative degradation curves and the relationship between peak performance and elasticity. We next examine whether scoring and protocol artifacts explain the cliffs.

5.1 Failure Outcomes Across Budgets

To understand *how* models fail under budget constraints, we group the 10,994 failed episodes by observable terminal outcome: reaching the budget boundary without SUBMIT, reaching the 20-turn limit without SUBMIT, submitting a wrong answer, submitting on the first turn without any tool use, or another format/tool/no-submit outcome; the full distribution is reported in Table 6 of Appendix B. Budget-boundary failures dominate for Qwen3.5/3.6, DeepSeek-V3, and Gemini-3.1-Flash-Lite (55–89%), whereas wrong submitted answers are the largest category for GPT-4o-mini and Qwen2.5 (50–69%). Qwen3.6-Flash also reaches the turn limit in 20.3% of its failed episodes. Thus, similar aggregate cliffs can reflect different observable failure outcomes.

5.2 Token Utilization and Efficient Stopping

Budget utilization ($\text{tokens_used} / \text{budget_tokens}$) shows how aggressively models consume their allocation. At B1 (200 tokens), Qwen3.5-397B uses 93% of its budget on average, while Gemini-3.1-Flash-Lite uses 85%. At B3 (1,500 tokens), successful episodes usually use fewer tokens than failed ones: GPT-4o-mini averages 363 tokens on successes vs. 422 on failures; Qwen3.6-Plus shows a larger gap (544 vs. 1,315). Across most model-budget combinations, successful episodes use a smaller fraction of the available budget than failed episodes. Detailed utilization statistics are reported in Table 12 of Appendix I.

5.3 Cliff Boundary Case Studies

Trajectory examples in Appendix J show how cliffs arise at the episode level. For one Qwen3.5-397B data-transformation task, the B1 run uses all 200 tokens in two generations and ends in a truncated THINK without SUBMIT. At B2, the same task succeeds in two generations using 150 tokens because

the second generation reaches SUBMIT. Other failures end differently: some models submit an incorrect answer within budget, while others repeat tool calls until the 20-turn limit. These cases illustrate that the same aggregate cliff can correspond to different observable trajectory outcomes.

5.4 Robustness Checks and Budget-Aware Prompting

The matched single-shot diagnostic in Table 3 produces much flatter budget-response curves than the agent loop. Under the stricter scoring rule in Table 11 of Appendix H, which counts every episode without a parser-recognized SUBMIT: as incorrect, absolute elasticities decrease but large budget cliffs remain. The metric-sensitivity analysis in Table 14 of Appendix N also yields nearly identical model rankings under log-trapezoidal and normalized-mean formulations ($\rho = 0.988$), differing only in the order of GPT-4o-mini and Claude-Haiku-4.5.

We also test budget-allocation prompting (BAP). The standard protocol already gives the total and remaining output-token budget; BAP is a separate ablation adding a strategy reminder to plan carefully and use fewer tokens per turn. Its effects are small and mixed on 50 tasks (Table 13 in Appendix K), providing no evidence that the main findings are caused by this extra prompt-engineering reminder.

6 Discussion

Peak performance is not budget robustness.

The most consistent pattern across our results is that high peak accuracy does not predict graceful degradation. Qwen3.5-397B reaches the highest peak ($s^*=0.973$) but ranks 9th of 10 in ε ; DeepSeek-V3 reaches $s^*=0.918$ but has the second-largest cliff. Conversely, Gemini-3.1-Flash-Lite, Qwen2.5-72B, and GPT-4o-mini achieve relative elasticity values of 0.919, 0.910, and 0.900, respectively, despite lower peak performance than Qwen3.6-Plus and Qwen3.5-397B. Model selection under variable budgets should therefore use the full success-budget curve and deployment budget distribution rather than ε alone.

Cliff location is a behavioral diagnostic. Cliff location reports the observed adjacent budget boundary with the largest success change. Nine of ten models cliff at B1→B2, indicating that 200 tokens is often insufficient for the typical agent loop: planning, one tool call, observation, and sub-

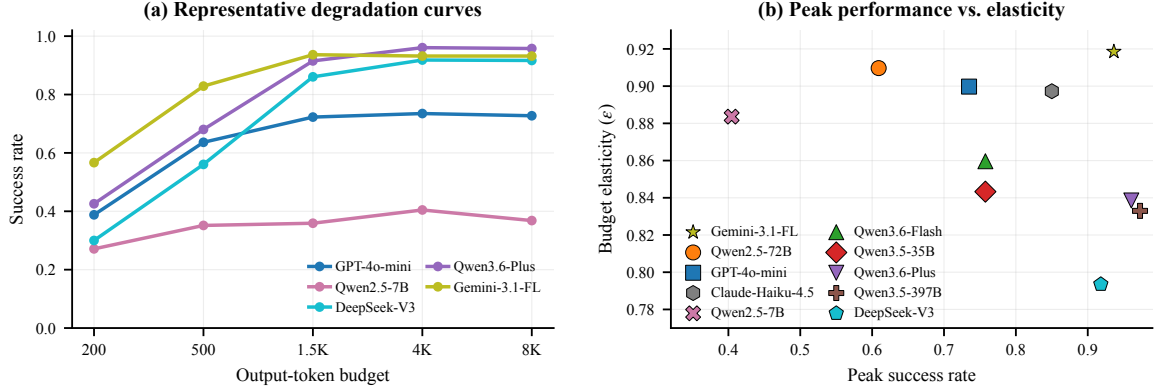


Figure 3: Complementary analysis of budget elasticity over 10 models and 660 tasks. (a) Budget-response curves for representative models. (b) Peak success versus relative elasticity; high peak capability does not necessarily imply graceful degradation.

mission. DeepSeek-V3 is the exception, cliffing at B2→B3 and showing continued budget sensitivity beyond 500 tokens. Combined with the terminal-outcome analysis in §5, cliff location provides a behavioral diagnostic of budget sensitivity but does not directly identify internal model strategy.

Output tokens, not full deployment cost. TokenCliff controls observable episode-level output-token budget, not total deployment cost or intrinsic model efficiency. Full cost also depends on input-token growth, tool latency, pricing, and hidden or separately reported reasoning tokens. Accordingly, our cross-model results compare models under their evaluated API settings and a common observable output-token budget protocol. They should not be interpreted as measurements of architecture-level efficiency.

Limitations of ϵ . The log-trapezoidal nAUC is shape-only by construction, so a model that succeeds uniformly at 30% across budgets still scores $\epsilon = 1.0$ despite low absolute capability. We therefore report ϵ with per-budget success, peak performance, Cliff, cliff location, and $A_{\log}$ rather than as an absolute capability or efficiency score.

Implications for agent evaluation. Budget constraints should be evaluated over complete agent trajectories. Although single-shot changes information access, its flatter curves and trajectory analysis suggest that the observed effect is not explained solely by one-shot response length. Failures also differ across models: some exhaust the budget before submission, whereas others submit within budget but remain incorrect.

Interpreting task-group differences. Domain-level elasticity measures retention relative to each task-group peak, not absolute difficulty. The Multi-Step split composes file, data-processing, and chained-reasoning operations but is not a uniformly harder tier. Its high ϵ can coexist with lower success because normalization removes the domain-specific peak. Reporting per-budget success and $A_{\log}$ alongside ϵ separates absolute capability from degradation shape and exposes task-dependent behavior.

7 Conclusion

We introduced TokenCliff, a benchmark that treats the output-token budget as a controlled variable in multi-turn LLM agent evaluation. Sweeping five budget levels across 660 deterministic tasks and 10 models (33,000 episodes), we measure how an agent’s success rate changes as its budget tightens, and summarize the resulting curve with two metrics: budget elasticity ϵ and the Cliff Index.

Three findings stand out from the experiments. First, multi-turn agent interactions produce performance cliffs of up to 32 percentage points between adjacent budget levels, while the 500-task single-shot diagnostic shows much smaller drops, suggesting that budget sensitivity is amplified in the evaluated multi-turn setting. Second, peak accuracy and model scale are not sufficient proxies for graceful degradation: Gemini-3.1-Flash-Lite ($\epsilon=0.919$), Qwen2.5-72B ($\epsilon=0.910$), and GPT-4o-mini ($\epsilon=0.900$) lead the elasticity ranking despite lower peak accuracy than Qwen3.5-397B (peak=0.973, $\epsilon=0.833$), and within the same Qwen3.5 MoE family the larger 397B is less elastic than the smaller 35B. Third, nine of ten models

cliff at B1→B2 (200→500 tokens), the observed adjacent boundary with the largest drop for most models.

For deployment, budget elasticity measures relative performance retention, while $A_{\log}$ and the full success-budget curve preserve absolute capability. Actual model selection should use these absolute curves together with the deployment budget distribution. The Cliff Index identifies the observed adjacent boundary with the largest drop. We release TokenCliff to support the development of budget-aware training methods and adaptive inference strategies for LLM agents.

Limitations

TokenCliff uses synthetic environments, which may not capture all complexities of real-world agent tasks. Our findings, including cliff locations and elasticity rankings, are established within these controlled simulations. Whether these patterns transfer to production agent deployments (e.g., SWE-bench, web navigation) remains an open question requiring validation on naturalistic benchmarks. We scope our claims to synthetic multi-turn settings and treat transfer validation as future work.

Additionally, the multi-step diagnostic in Table 5 is a subset analysis of 50 large-table tasks drawn from the main evaluation. It reports three models and should not replace the full 660-task results.

Token counting uses API-reported `completion_tokens` when available and `cl100k_base` otherwise. Tokenizer conventions, usage fields, and native-reasoning exposure differ across providers, so comparisons concern deployed API configurations under a shared observable output-budget protocol, not intrinsic architectural efficiency. Input-token growth, latency, tool costs, and byte-level normalization remain future work.

Reproducibility Statement

All code, tasks, and evaluation scripts are publicly available. Key reproducibility details:

- **Tasks:** All 660 tasks are deterministic and seed-controlled. The released JSON files constitute the canonical benchmark instance.
- **Decoding:** All models queried with `temperature=0` (greedy decoding) to minimize sampling variance; closed-provider APIs may still exhibit backend-induced nondeterminism.

- **Hardware:** All experiments run on CPU; no GPU required. The evaluation harness uses only CPU for orchestration and API calls.
- **Cost:** Total API cost for the full 33,000-episode evaluation was approximately \$85 USD across all providers.
- **Runtime:** Full evaluation completes in ~48 hours with rate-limited API access.

Ethical Considerations

TokenCliff uses fully synthetic tasks with no human subjects, personal data, or sensitive content. All environments are deterministic simulations. We evaluate only publicly available API models.

Acknowledgments

This work was supported by Guangdong Basic and Applied Basic Research Foundation Project 2025A1515010304, Guangdong Province Project 2024QN11X088, Guangzhou Science and Technology Planning Project 2025A03J4491. This work was also partially supported by the Research Travel Grant of the Red Bird MPhil (RBM) Program at The Hong Kong University of Science and Technology (Guangzhou). We thank our RBM project supervisor, Tongge Huang, for academic support.

References

- Zhiyu Fan, Kirill Vasilevski, Dayi Lin, Boyuan Chen, Yihao Chen, Zhiqing Zhong, Jie M. Zhang, Pin-jia He, and Ahmed E. Hassan. 2025. [SWE-Effi: Re-evaluating software AI agent system effectiveness under resource constraints](#). *arXiv preprint arXiv:2509.09853*.
- Zhengru Fang, Senkang Forest Hu, Zhonghao Chang, Yu Guo, Yihang Tao, Hongyao Liu, Mengzhe Ruan, Jun Huang, and Yuguang Fang. 2026. [Inference-time budget control for LLM search agents](#). *arXiv preprint arXiv:2605.05701*.
- Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. [Token-budget-aware LLM reasoning](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855, Vienna, Austria. Association for Computational Linguistics.
- Sayash Kapoor, Benedikt Stroebel, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. 2025. [AI agents that matter](#). *Transactions on Machine Learning Research*.

Jiin Kim, Byeongjun Shin, Jinha Chung, and Minsoo Rhu. 2026. [The cost of dynamic reasoning: Demystifying AI agents and test-time scaling from an AI infrastructure perspective](#). In *2026 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 1–16. IEEE Computer Society.

Keyu Li, Junhao Shi, Yang Xiao, Mohan Jiang, Jie Sun, Yunze Wu, Dayuan Fu, Shijie Xia, Xiaojie Cai, Tianze Xu, Weiye Si, Wenjie Li, Dequan Wang, and Pengfei Liu. 2026a. [AgencyBench: Benchmarking the frontiers of autonomous agents in 1M-token real-world contexts](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7422–7440, San Diego, California, United States. Association for Computational Linguistics.

Yushu Li, Wenlong Deng, Jiajin Li, and Xiaoxiao Li. 2026b. [Spend less, reason better: Budget-aware value tree search for LLM agents](#). *arXiv preprint arXiv:2603.12634*.

Hanbing Liu, Chunhao Tian, Nan An, Ziyuan Wang, Pinyan Lu, Changyuan Yu, and Qi Qi. 2026a. [Budget-constrained agentic large language models: Intention-based planning for costly tool use](#). *arXiv preprint arXiv:2602.11541*.

Tengxiao Liu, Zifeng Wang, Jin Miao, I-Hung Hsu, Jun Yan, Jiefeng Chen, Rujun Han, Fangyuan Xu, Yanfei Chen, Ke Jiang, Samira Daruki, Yi Liang, William Yang Wang, Tomas Pfister, and Chen-Yu Lee. 2026b. [Budget-aware tool-use enables effective agent scaling](#). In *Third Conference on Language Modeling*.

Kyle A. McCleary and James M. Ghawaly. 2026. [Quantifying the accuracy and cost impact of design decisions in budget-constrained agentic LLM search](#). In *Proceedings of the Fifteenth Language Resources and Evaluation Conference*, pages 10291–10300, Palma de Mallorca, Spain. ELRA Language Resource Association.

Arihant Tripathy, Ch Pavan Harshit, and Karthik Vaidhyanathan. 2026. [SWEnergy: An empirical study on energy efficiency in agentic issue resolution frameworks with SLMs](#). In *Proceedings of the 2026 International Workshop on Agentic Engineering*, pages 104–111. Association for Computing Machinery.

Junlin Wang, Siddhartha Jain, Dejiao Zhang, Baishakhi Ray, Varun Kumar, and Ben Athiwaratkun. 2024. [Reasoning in token economies: Budget-aware evaluation of LLM reasoning strategies](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19916–19939, Miami, Florida, USA. Association for Computational Linguistics.

Yong Wu, YanZhao Zheng, TianZe Xu, ZhenTao Zhang, YuanQiang Yu, JiHuai Zhu, Chao Ma, BinBin Lin, BaoHua Dong, HangCheng Zhu, RuoHui Huang,

and Gang Yu. 2026. [ContextBudget: Budget-aware context management for long-horizon search agents](#). *arXiv preprint arXiv:2604.01664*.

Yuhui Xu, Hanze Dong, Lei Wang, Doyen Sahoo, Junnan Li, and Caiming Xiong. 2026. [Scalable chain of thoughts via elastic reasoning](#). In *International Conference on Learning Representations*.

A Multi-Step Diagnostic Details

Table 5 reports a subset analysis of the 50 large-table tasks (multistep_001–050) drawn from the canonical 160-task Multi-Step split. All values are extracted from the main 33,000-episode evaluation rather than a separate rerun.

Model	B1	B2	B3	B4	B5	ϵ	$A_{\log}$	Cliff
Qwen3.5-35B	.520	.880	.900	.960	.940	.901	.865	.360
GPT-4o-mini	.440	.660	.640	.700	.720	.892	.642	.220
Qwen2.5-72B	.240	.520	.540	.520	.340	.877	.474	.280

Table 5: Main-run subset analysis of the 50 large-table tasks from the 160-task Multi-Step split. All five budgets are shown; all three Cliff locations are B1→B2.

B Failure Taxonomy Details

Model	BudgLim	TurnLim	WrongA	EarlySub	Other
Gemini-3.1-Flash-Lite	60.6	2.4	29.4	1.5	6.0
Qwen2.5-72B	26.1	4.1	59.4	10.3	0.1
GPT-4o-mini	40.4	0.9	49.5	9.1	0.1
Claude-Haiku-4.5	50.9	10.3	30.5	4.4	3.9
Qwen2.5-7B	13.8	3.0	69.0	14.0	0.3
Qwen3.6-Flash	54.5	20.3	21.7	3.0	0.6
Qwen3.5-35B	66.8	10.0	21.0	2.0	0.3
Qwen3.6-Plus	89.1	6.0	4.6	0.0	0.3
Qwen3.5-397B	88.8	2.9	7.9	0.0	0.4
DeepSeek-V3	82.0	3.0	13.7	0.3	0.9

Table 6: Observable terminal outcomes (% of failures). BudgLim = no SUBMIT after using at least 95% of the budget; TurnLim = no SUBMIT at the 20-turn limit; WrongA = wrong submitted answer; EarlySub = first-turn submission with no USE anywhere in the trajectory; Other combines format, tool, and remaining no-submit outcomes. Categories are mutually exclusive. Percentages are independently rounded to one decimal place and may not sum to exactly 100%.

C Model Details

Provider interfaces differ in tokenizer conventions and reasoning-token exposure. In a matched ablation across four Qwen3.x models and all five budgets, we varied only enable_thinking; effects on success were mixed, while DashScope separately exposed nonzero reasoning usage. We therefore interpret cross-model results as comparisons of

Model(s)	Source	Serving route	Native reasoning setting	Separate reasoning usage
Qwen2.5-7B/72B	Qwen	DashScope	Default; no exposed switch	Not separately exposed
Qwen3.5-35B/397B	Qwen	DashScope	enable_thinking=False	Exposed in thinking mode
Qwen3.6-Flash/Plus	Qwen	DashScope	enable_thinking=False	Exposed in thinking mode
GPT-4o-mini	OpenAI	OFOX	Default; no exposed switch	Not separately exposed
Claude-Haiku-4.5	Anthropic	OFOX	Default; no exposed switch	Not separately exposed
DeepSeek-V3	DeepSeek	OFOX	Default; no exposed switch	Not separately exposed
Gemini-3.1-Flash-Lite	Google	OFOX	Default; no exposed switch	Not separately exposed

Table 7: Model configuration and token accounting. Models outside DashScope use an OpenAI-compatible gateway. The evaluator budgets API-reported completion_tokens, using c1100k_base only when API usage is unavailable.

the evaluated serving configurations, not intrinsic hidden-reasoning efficiency.

D Single-Shot Evaluation Details

In single-shot mode, the entire environment state is provided in the prompt and the model produces a single response. Results on 500 tasks:

Model	B1	B2	B3	B4	B5	ϵ	Cliff
GPT-4o	.580	.570	.558	.512	.564	.955	.052
GPT-4o-mini	.456	.474	.472	.476	.472	.991	.018
Qwen2.5-72B	.490	.484	.488	.488	.500	.977	.012
Qwen3-32B	.450	.442	.434	.440	.442	.978	.006
Qwen2.5-7B	.384	.378	.384	.380	.366	.989	.006
Qwen3-8B	.362	.366	.364	.366	.366	.997	.004

Table 8: Single-shot results (500 tasks). ϵ is log-budget trapezoidal nAUC normalized by peak. Values range from 0.955 to 0.997, showing substantially flatter budget-response curves than in the agent loop.

For the paired comparison, we jointly resample the 500 matched task IDs across both settings and all five budgets 2,000 times. The resulting 95% confidence intervals for $\epsilon_{\text{single}} - \epsilon_{\text{loop}}$ are [0.079, 0.133] for GPT-4o-mini, [0.041, 0.114] for Qwen2.5-72B, and [0.040, 0.147] for Qwen2.5-7B.

E Task Examples

File Operations. “List all files in /data and find the one with the most lines. Return only the filename.” Agent must call `ls(/data)`, then `stat()` on each file, compare line counts, and submit.

Data Transformation. “How many employees are in the ‘legal’ department?” Agent must call `query(employees)` to retrieve data, filter by department, count, and submit.

Tool Use. “Get temperatures of Tokyo, London, NYC. Find the hottest city’s population. Multiply by exchange rate.” Requires 5 sequential API calls.

Planning. “Schedule 3 one-hour meetings avoiding conflicts.” The task provides synthetic constraints, and the agent must reason over them and submit a valid schedule; this domain measures constraint reasoning under a token budget rather than interaction with a real calendar API.

Multi-Step. “Read all files in /reports, find the values of four specified metrics, and return their sum.” This benchmark task requires listing the directory, extracting values distributed across 15 files, and aggregating them.

F Domain-Level Performance and Budget Sensitivity

Table 9 aggregates the main 33,000-episode evaluation across all 10 models. Each base-domain value uses 1,250 episodes per budget; Multi-Step uses 1,600. Avg. Cliff first takes each model’s largest adjacent change within a domain and then averages across models.

Domain	B1 Succ.	B5 Succ.	Avg. Cliff
File Operations	0.230	0.678	0.271
Planning	0.379	0.854	0.283
Data Transformation	0.369	0.846	0.269
Multi-Step	0.413	0.796	0.281
Tool Use	0.564	0.734	0.122

Table 9: Domain-level performance and budget sensitivity. B1/B5 are mean success across models; Avg. Cliff averages each model’s largest adjacent change within the domain.

G Full Confidence Intervals

Table 10 reports 95% bootstrap confidence intervals (2,000 independent within-budget resamples; seed 20260829) for representative budget levels.

Model	B1 CI		B3 CI		B5 CI	
	Lo	Hi	Lo	Hi	Lo	Hi
Gemini-3.1-Flash-Lite	.529	.605	.918	.955	.912	.952
Qwen2.5-72B	.318	.389	.567	.642	.561	.633
GPT-4o-mini	.350	.424	.689	.758	.692	.761
Claude-Haiku-4.5	.439	.515	.809	.867	.823	.877
Qwen2.5-7B	.236	.305	.324	.397	.332	.405
Qwen3.6-Flash	.338	.412	.720	.785	.723	.791
Qwen3.5-35B	.338	.409	.662	.732	.724	.791
Qwen3.6-Plus	.388	.464	.894	.936	.941	.971
Qwen3.5-397B	.355	.427	.883	.927	.948	.977
DeepSeek-V3	.264	.336	.833	.885	.895	.938

Table 10: 95% bootstrap CIs for B1, B3, B5 (representative levels).

H Strict Submission Scoring

The main evaluation scores the last model output when an episode ends without a parser-recognized SUBMIT:, which avoids discarding recoverable final answers. As a robustness check, Table 11 applies a stricter rule: a successful answer is counted only if a model generation contains a parser-recognized SUBMIT:. This rule lowers absolute success rates and elasticities, especially for models that often produce direct answers, but the qualitative finding remains: multi-turn agents still exhibit large adjacent-budget cliffs under explicit output-token budgets.

Model	ϵ_{main}	ϵ_{strict}	Cliff _{strict}
Gemini-3.1-FL	.919	.913	.286
Qwen2.5-72B	.910	.890	.174
GPT-4o-mini	.900	.864	.333
Claude-Haiku-4.5	.897	.876	.342
Qwen2.5-7B	.884	.873	.108
Qwen3.6-Flash	.860	.823	.248
Qwen3.5-35B	.843	.795	.253
Qwen3.6-Plus	.839	.787	.312
Qwen3.5-397B	.833	.787	.338
DeepSeek-V3	.794	.722	.403

Table 11: Strict no-SUBMIT robustness. ϵ_{strict} and Cliff_{strict} are recomputed after counting every successful episode without a parser-recognized SUBMIT: as incorrect.

I Token Efficiency Analysis

Table 12 reports mean budget utilization and token use for successful and failed episodes.

J Trajectory Examples

Each numbered turn below is one model generation. Labels such as THINK+USE denote combined gen-

Model	Utilization Rate			Tokens (Succ / Fail)		
	B1	B3	B5	B1	B3	B5
GPT-4o-mini	.90	.25	.05	156/194	363/422	384/560
Qwen3.6-Plus	.91	.41	.09	157/200	544/1315	725/1543
Qwen3.5-397B	.93	.38	.07	166/200	504/1252	676/1250
DeepSeek-V3	.95	.46	.12	168/198	616/1154	876/2180
Claude-Haiku-4.5	.91	.27	.05	164/199	356/643	389/643
Gemini-3.1-Flash-Lite	.85	.23	.05	151/195	335/371	392/258

Table 12: Budget utilization and token usage for successful vs. failed episodes. Successful episodes usually use fewer tokens in the reported model-budget cells.

erations that do not strictly follow the one-marker instruction but are recognized by the permissive parser. If a generation contains USE, the environment executes the action and appends its observation before the next generation.

Successful multi-turn (Qwen3.6-Plus, B3=1500 tokens, data_transform_001):

Turn 1 (THINK+USE, 64 tok): Plans a count query and calls query for employees in the legal department.

Turn 2 (THINK+USE, 170 tok): After observing one returned row, requests all matching legal-department rows to verify the count.

Turn 3 (THINK+SUBMIT, 33 tok): “The query returned a list with exactly one employee... SUBMIT: 1.” → ✓ Correct. Total: 267/1500 tokens used.

Budget exhaustion failure (Qwen3.5-397B, B1=200 tokens, data_transform_029):

Turn 1 (THINK+USE, 72 tok): Plans a count query and calls query for employees in the sales department.

Turn 2 (truncated THINK, 128 tok): “The query returned all rows instead of filtering for ‘sales’ and counting... None of them are...”

→ ✗ Budget exhausted (200/200). Never reached SUBMIT.

Turn-limit failure (Qwen2.5-7B, B4=4000 tokens, data_transform_063):

Turn 1 (THINK+USE, 124 tok): Plans a join-and-aggregate query and calls query.

Turns 2–20 (USE, 54 tok each): Repeats the same unsupported join query without reaching SUBMIT. → ✗ Reached the 20-turn limit after using 1,150/4,000 tokens.

K Budget-Aware Prompting (BAP)

The standard TokenCliff protocol already provides the total and remaining episode-level output-token budget. Budget-Aware Prompting (BAP) is a separate ablation that adds the strategy reminder “*Plan carefully. Use minimal tokens per turn.*” without changing the budget constraint. We evaluate it on

Model	Standard	BAP	Δ
Qwen2.5-72B	0.576	0.592	+0.016
GPT-4o-mini	0.844	0.816	-0.028
Qwen3.5-35B	0.936	0.940	+0.004

Table 13: BAP ablation on 50 Data Transformation tasks. Standard and BAP are mean success across B1–B5 without and with the additional strategy reminder; BAP is not used in the main evaluation.

50 Data Transformation tasks and report mean success across the five budgets.

The small, mixed changes do not indicate a systematic benefit. This 50-task, three-model ablation does not establish effects for other prompting strategies.

L Task Generation Pseudocode

Algorithm below shows the simplified generation logic for one domain (Data Transform). Other domains follow the same three-step pattern with domain-specific environment and template choices.

```
def generate_data_transform_task(seed):
    random.seed(seed)

    # Step 1: Synthesize environment
    depts = [engineering, sales, hr,
             legal, marketing, finance]
    employees = []
    for i in range(random.randint(15, 30)):
        employees.append({
            name: random_string(8),
            dept: random.choice(depts),
            salary: random.randint(50k, 200k),
            city: random.choice(cities)
        })

    # Step 2: Choose template + compute answer
    template = random.choice([
        count_by_dept,
        max_salary_dept,
        cross_table_join, ...
    ])
    if template == count_by_dept:
        dept = random.choice(depts)
        answer = count(e for e in employees
                       if e.dept == dept)
        prompt = f"How many employees in '{dept}'? Return number."

    # Step 3: Package as JSON
    return {
        id, prompt, tools=[query],
        environment={tables: {employees}},
        ground_truth=str(answer)
    }
```

Pseudocode for task generation (Data Transform domain). Full scripts are included in the released code.

M System Prompt Template

The following is the system-prompt template used in the implementation; placeholders are filled with episode-specific content.

```
You are an agent solving a task. You
have a total budget of {budget} output
tokens across ALL turns.
Task: {task.prompt}
Available tools:
{tools_desc}
Environment:{env_hint}
Each turn, respond with EXACTLY ONE of:
THINK: <your reasoning>
USE: tool_name(arguments)
SUBMIT: <your final answer>
Rules:
- USE calls a tool and returns its
  output. Arguments are strings or JSON.
- THINK lets you reason (costs tokens
  but no tool call).
- SUBMIT ends the task. Give ONLY the
  answer after SUBMIT:
- Be efficient—you have limited budget.
- For query tool: query(table_name) or
  query({"table": "name"})
- For API tools: call({"endpoint": "name",
  "param": "value"}) or
  tool_name({"param": "value"})
```

The evaluator then sends “Begin.” Before each subsequent generation after a THINK step or tool observation, it appends: “Budget remaining: ~{remaining} tokens. Continue.”

N Metric Sensitivity

Table 14 reports nearly identical rankings under the log-budget trapezoidal and normalized-mean formulations ($\rho = 0.988$); only GPT-4o-mini and Claude-Haiku-4.5 exchange positions.

Model	Log-Trap ε	Mean ε
Gemini-3.1-FL	0.9185 (1)	0.8961 (1)
Qwen2.5-72B	0.9097 (2)	0.8851 (2)
GPT-4o-mini	0.8998 (3)	0.8734 (4)
Claude-Haiku-4.5	0.8973 (4)	0.8759 (3)
Qwen2.5-7B	0.8836 (5)	0.8674 (5)
Qwen3.6-Flash	0.8596 (6)	0.8404 (6)
Qwen3.5-35B	0.8433 (7)	0.8296 (7)
Qwen3.6-Plus	0.8386 (8)	0.8202 (8)
Qwen3.5-397B	0.8330 (9)	0.8109 (9)
DeepSeek-V3	0.7935 (10)	0.7746 (10)

Table 14: Ranking comparison using unrounded success rates: log-trapezoidal vs. normalized mean.