

Comparative Study of AI Agent Architecture Efficiency in Table Processing Tasks

Timur Gabbasov, Aleksandr Sandalov, Aleksandr Maksimov,
Maria Khodorchenko, Nikolay Butakov

ITMO University
Industrial AI Research Lab
Saint Petersburg, Russia

Abstract

Tables are central to business document workflows, but practical table-processing tasks vary substantially in structure, content, and required action. We compare eight ReAct-agent configurations built on the same Qwen3.6-35B-A3B model. The design crosses four tool profiles—Minimal, Search, Spreadsheet, and Full—with the presence or absence of Markdown-based procedural skills. We evaluate the configurations on 40 tasks from company workflows, FinQA, and SpreadsheetBench. Across three complete replicates, comprising 960 agent–task episodes, Full with skills obtains the highest aggregate success rate of 75.0% and a mean score of 0.8061. Full without skills reaches 65.0%, while Spreadsheet without skills reaches 73.3%. Aggregated across tool profiles, skills increase success from 69.4% to 72.3%, but their effect is heterogeneous: negative for Minimal and Spreadsheet and positive for Search and Full. Overall success remains stable across replicates at 70.3%, 70.9%, and 71.3%, although individual configuration results vary more substantially. These findings suggest that procedural guidance is most promising when an agent must coordinate a broad tool environment rather than as a universal augmentation.

1 Introduction

Business processes continuously produce and consume semi-structured documents: financial workbooks, operational reports, patent tables, translation sheets, inventories, reconciliations, and ad hoc analytical files. Unlike plain text, a spreadsheet combines values, formulas, formatting, sheet structure, and implicit layout conventions. A useful agent must therefore do more than answer a question about isolated cells. It may need to inspect a workbook, locate relevant regions, aggregate values, interpret long text, edit formulas or content, and save a valid artifact while preserving unrelated information.

Large language models make it possible to construct general-purpose agents for these workflows, but performance depends not only on the model. The surrounding architecture determines what the model can inspect, which operations it can execute, how it searches long tables, whether it delegates analysis to specialized tool wrappers, and whether it can consult reusable task guidance. In practice, the same model can behave very differently when equipped with only Python, with search tools, with spreadsheet-specific execution components, or with a combined tool environment.

Current spreadsheet and table benchmarks provide valuable test cases, but each benchmark represents only part of the business task surface. SpreadsheetBench focuses on real-world spreadsheet manipulation (Ma et al., 2024), while FinQA focuses on numerical reasoning over financial reports (Chen et al., 2021). Company workflows additionally include long textual tables, translation-oriented documents, and compact but semantically dense accounting sheets. This motivates an evaluation that combines public benchmark tasks with tasks obtained from actual company practice.

This work asks three research questions:

- RQ1:** Which table-processing task types occur in business practice, and how completely are they represented by existing benchmarks?
- RQ2:** Which organization of an AI agent most effectively solves business-oriented table-processing tasks?
- RQ3:** Does a modern LLM-based agent benefit from explicit skills in addition to executable tools?

We address these questions through a controlled comparative study. All configurations use the same ReAct loop (Yao et al., 2023) and Qwen3.6-35B-A3B model (Qwen Team, 2026). We vary the available toolset and the presence of skills, producing

eight configurations in a 4×2 design. The agent operates locally, without Internet access, and may execute Python with pandas, openpyxl, and numpy. Each episode is allowed up to 100 agent-loop iterations.

The paper makes two main contributions:

- A heterogeneous evaluation set that combines 20 tasks from real company practice with public FinQA and SpreadsheetBench samples. The company tasks cover both question answering and editing across nine operational domains, including accounting, patent, and translation-oriented tables.
- A three-replicate comparative study of tool organization and skill augmentation for a fixed ReAct agent and model, comprising 960 episodes and reporting quality, latency, iteration, token, and replicate-level stability measurements for all eight configurations.
- An empirical analysis showing that the effect of procedural skills is tool-profile dependent: positive for the Search and Full profiles in aggregate, but absent or slightly negative for the narrower Minimal and Spreadsheet profiles.

2 Related Work

2.1 Benchmarks for tables and spreadsheets

Research on table reasoning includes question answering and verification over semi-structured tables, such as WikiTableQuestions (Pasupat and Liang, 2015), TabFact (Chen et al., 2020), TaPas (Herzig et al., 2020), and TaBERT (Yin et al., 2020). These benchmarks helped establish table understanding as a distinct reasoning problem, but their output is primarily textual and they generally do not require modifying and returning a workbook.

FinQA introduced numerical reasoning over financial reports by combining text, tables, and executable programs (Chen et al., 2021). It is relevant to business analytics, but it primarily evaluates question answering and numerical derivation rather than general spreadsheet editing. SpreadsheetBench introduced real-world spreadsheet manipulation and artifact-based evaluation (Ma et al., 2024). It includes diverse workbook operations and is substantially closer to end-to-end office automation. However, public benchmarks still leave gaps relative to internal business workflows, particularly long text-heavy tables, translation sheets,

and domain-specific documents whose structure is shaped by company processes rather than benchmark design.

Our goal is not to replace these benchmarks. Instead, we use public samples as anchors and add tasks from real company practice to study whether conclusions about agent architecture remain consistent across heterogeneous task families.

2.2 Agent organizations for table processing

Several systems organize LLMs around actions, planning, validation, and structured workbook access. SheetCopilot (Li et al., 2023), SpreadsheetLLM (Dong et al., 2024), SheetAgent (Chen et al., 2025), and related systems explore tool use, spreadsheet encodings, multi-step execution, and specialized reasoning components. ReAct interleaves reasoning and acting (Yao et al., 2023), while CodeAct uses executable code as an action space (Wang et al., 2024). These approaches motivate our base architecture: a ReAct agent that can invoke Python and optional specialized tools. Beyond spreadsheets, document-processing approaches include topic-specialized generation models (Nevezhin et al., 2020) and hybrid retrieval combining text with graph-community summaries (Karkout et al., 2026).

The present study focuses on two architectural choices. The first is *tool specialization*: whether the agent receives only a general Python execution tool, search-oriented tools, spreadsheet-specific tool wrappers, or the union of these capabilities. The second is *skill augmentation*. We define a skill as a Markdown instruction file injected into the agent’s system prompt. Skills contain generalized experience and recommended procedures for particular table types; they do not execute code and are distinct from tools. Unlike prior systems that propose a particular integrated spreadsheet-agent architecture, we hold the base model and ReAct loop fixed while varying executable tool access and procedural prompt guidance as separate experimental factors.

3 Dataset

3.1 Composition

The evaluation set contains 40 tasks from three sources. Twenty tasks originate from the real operational practice of one company, ten are sampled from FinQA, and ten are sampled from SpreadsheetBench. The corporate subset is evenly di-

Source	Tasks	Primary mode
Corporate	20	10 QA / 10 editing
FinQA	10	Financial QA
SpreadsheetBench	10	Workbook manipulation

Table 1: Evaluation-set composition. Corporate tasks capture real company workflows and heterogeneous document structures; FinQA contributes numerical reasoning over financial reports and tables; SpreadsheetBench is a public real-world spreadsheet-editing benchmark.

vided into 10 question-answering tasks and 10 editing tasks. Each configuration processes the same 40 tasks in three complete replicates, yielding 120 episodes per configuration and 960 agent-task episodes in total. Table 1 summarizes the 40 unique tasks; replicates repeat the same task set while allowing stochastic agent trajectories to vary.

3.2 Corporate table families

The company-derived tasks span nine operational domains: accounting (4 tasks) and human resources, IT, incident management, operations, patents, procurement, project management, and translation (2 tasks each). Each domain contributes equal numbers of question-answering and editing tasks. Structurally, the tasks were selected to represent three recurring table families.

Accounting tables. Relatively small but semantically dense workbooks. They require careful interpretation of financial labels, values, formulas, periods, and aggregation rules.

Patent tables. Long tables containing extensive free text. These tasks stress retrieval, evidence localization, and synthesis across text-heavy rows and columns.

Translation-oriented tables. Long documents in which text must be located, interpreted, or transformed while preserving tabular organization and workbook structure.

These categories broaden the test surface beyond a single spreadsheet format. They vary in length, text density, numeric content, and whether the required output is an answer or a modified artifact.

4 Methods

4.1 Base agent and environment

All configurations use the same ReAct agent with Qwen3.6-35B-A3B (Qwen Team, 2026) as

the backbone model, served in its instruct (non-thinking) configuration with native tool calling. The agent alternates between reasoning, selecting an action, observing the tool result, and continuing until it returns an answer or output artifact. The execution environment is local and has no Internet access. Python execution supports pandas, openpyxl, and numpy for data analysis and workbook manipulation. Full model, sampling, and serving details are listed in Appendix A.

Each task permits at most 100 agent-loop iterations. No separate hard limits were imposed on elapsed time or token consumption; these quantities are measured as outcomes. This design prioritizes task completion while allowing efficiency differences to emerge from actual agent behavior.

4.2 Tools

The experiment uses five functional capabilities: a Python execution tool, an aspect-summarization tool, an exhaustive table-search tool, a tabular-analysis subagent, and a workbook-manipulation subagent. They are grouped into four toolsets:

- **Minimal:** the Python execution tool only. This is the general-purpose baseline.
- **Search:** the Python execution tool plus aspect summarization and exhaustive table search. This toolset targets long and text-heavy tables through decomposition and systematic retrieval.
- **Spreadsheet:** the Python execution tool plus tabular-analysis and workbook-manipulation tool wrappers. This toolset delegates structured analysis and workbook operations to specialized components.
- **Full:** all five capabilities. This toolset maximizes the available action space and requires the model to select among both search-oriented and spreadsheet-oriented tools.

The five capabilities differ in scope. `python_executor` runs code in an isolated persistent worker whose filesystem access is restricted to the episode workspace and whose imports are controlled by an allowlist. `aspect_summary` deterministically profiles workbook sheets, candidate headers, samples, and density using `openpyxl`. `exhaustive_search` performs bounded local token search over workbook cells. `pandas_analysis_agent` loads a

rectangular table into a DataFrame and executes supplied analysis code. `xlsx_subagent` provides workbook previews and executes `openpyxl`-based operations in an isolated runtime. Despite their names, the latter two are specialized tool wrappers rather than independently reasoning LLM agents.

4.3 Skills

Skills are declarative Markdown instruction files rather than executable tools. They describe runtime constraints, task-type branching, tool-selection conditions, table-family characteristics, expected output formats, workbook-preservation requirements, and final validation procedures.

Skills are not retrieved or selected by the model during an episode. When the skill condition is enabled, the experiment harness deterministically selects the files before the first model request and injects their content into the system prompt. Selection uses the task-family label and the active tool profile (*Minimal*, *Search*, *Spreadsheet*, or *Full*). Each episode receives one family-specific skill and one generic skill for the corresponding tool profile, subject to a 12,000-character limit. Skill exposure is therefore guaranteed, although the experiment does not measure whether the model follows every individual instruction.

The skills were authored from tool contracts, recurring table-family characteristics, required task interfaces, and preliminary smoke-run observations. They contain no task-specific reference answers, target values, or hidden workbook contents. However, because the skills encode knowledge of the task formats and evaluation interface, the experiment evaluates their utility on a fixed benchmark rather than transfer to entirely unseen task families. Every tool profile is evaluated both without and with skills, yielding the eight configurations in the 4×2 design.

5 Experiments

5.1 Methodology and metrics

We execute three complete replicates of the experimental matrix. In each replicate, all eight configurations process the same 40 tasks, producing 320 episodes per replicate and 960 episodes in total. The task-order seed is fixed within each replicate and shared across configurations. Stochastic decoding may nevertheless produce different trajectories across replicates.

Tasks are executed with concurrency 15. The

complete three-replicate matrix takes approximately 7 hours 31 minutes of wall-clock time. Because episodes run concurrently, average episode duration must not be summed to estimate wall-clock runtime.

We report five primary quantities: success rate, mean score, average time per task, average number of iterations, and average token usage. Success rate measures the fraction of tasks judged successfully completed. Mean score preserves partial credit from the task evaluators. Average time is the full duration of an individual episode. Average iterations measures the number of agent-loop steps used. Average tokens is the mean total (prompt plus completion) token count per episode; a prompt/completion breakdown is given in Appendix B.

Question-answering tasks are scored by answer matching; FinQA tasks use a FinQA-style evaluator. Editing tasks are scored by cell-level comparison of the produced workbook against a reference. In `literal` mode, values, text, and required blank cells must match and formulas are not accepted in place of values. In `formula_exact` mode, the formula string must match exactly. In `formula_semantic` mode, an alternative formula is accepted only when its computed behavior and direct dependencies satisfy the task rule. Values, formulas, sheet names, and merged ranges outside the allowed edit surface must remain unchanged. Editing success requires a score of 1.0 and successful preservation; the continuous score records the fraction of matched target cells and therefore retains partial credit for nearly correct workbooks.

We report aggregate results over all three replicates together with replicate-level outcomes. For skill-versus-no-skill comparisons within each tool profile, we use an exploratory paired task-cluster sign-flip test. The resulting p -values are uncorrected; none of the four profile-level success effects passes a Bonferroni threshold of $\alpha = 0.0125$. We therefore treat the significance analysis as exploratory and emphasize effect heterogeneity and replicate-level stability rather than binary significance claims.

5.2 Results

Table 2 reports the primary metrics for all eight configurations, and Figure 1 visualizes success rates.

Across the three replicates, Full+Skills is the best aggregate configuration, with 90 successful episodes out of 120 (75.0%) and a mean score of

Toolset	Skills	Success rate	Mean score	Time (s)	Iterations	Tokens (k)
Minimal	no	84/120 (70.0%)	0.7742	136.4	7.10	41.3
Minimal	yes	81/120 (67.5%)	0.7745	194.5	7.79	96.5
Search	no	83/120 (69.2%)	0.7526	160.0	7.49	48.5
Search	yes	89/120 (74.2%)	0.8021	159.3	7.22	63.3
Spreadsheet	no	88/120 (73.3%)	0.7833	151.1	7.38	58.5
Spreadsheet	yes	87/120 (72.5%)	0.7933	188.8	7.72	89.4
Full	no	78/120 (65.0%)	0.7616	178.5	7.02	60.7
Full	yes	90/120 (75.0%)	0.8061	176.3	6.95	72.5
All	—	680/960 (70.8%)	0.7810	168.1	7.33	66.3

Table 2: Aggregate results over three complete replicates (120 episodes per configuration). Time, iterations, and tokens are per-episode means; tokens include prompt and completion tokens.

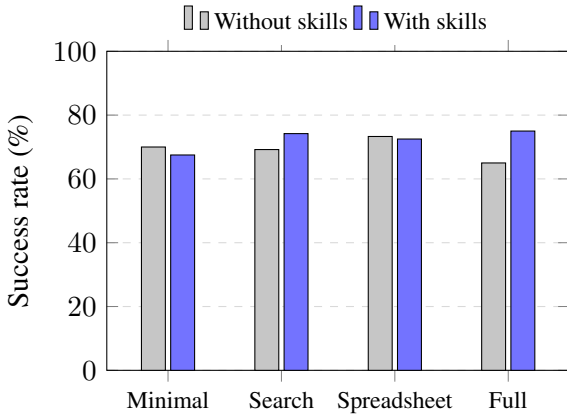


Figure 1: Aggregate success rate by tool profile and skill condition over three complete replicates. Each bar summarizes 120 episodes.

Replicate	Success	Rate	Mean score
1	225/320	70.3%	0.7659
2	227/320	70.9%	0.7880
3	228/320	71.3%	0.7891

Table 3: Aggregate stability across the three replicates.

0.8061. Full without skills reaches 65.0%, producing an aggregate within-profile difference of 10 percentage points. The size of this difference varies across replicates: 20 points in Replicate 1, 5 points in Replicate 2, and 5 points in Replicate 3.

The aggregate success rate is stable across replicates, ranging from 70.3% to 71.3%. Individual configurations vary more substantially. In particular, Full+Skills decreases from 82.5% in Replicate 1 to 72.5% in Replicate 2 and 70.0% in Replicate 3. This illustrates why the primary interpretation should rely on the three-replicate aggregate rather than on the strongest individual run.

Search improves from 69.2% without skills to 74.2% with skills. In contrast, Minimal decreases

from 70.0% to 67.5%, and Spreadsheet changes from 73.3% to 72.5%. Skills therefore do not provide a uniform benefit: their observed effect depends on the active tool profile.

Among configurations without skills, Spreadsheet is strongest at 73.3%, while Full is weakest at 65.0%. This provides evidence against a simple “more tools is better” interpretation. A broad action space may introduce tool-selection and coordination costs, although the current design does not isolate tool count from tool type.

Table 4 breaks down success rates by task category. Across all configurations and replicates, Corporate Editing is the easiest category at 96.7% success, followed by Corporate QA at 84.6%. FinQA reaches 71.7%, while SpreadsheetBench is substantially harder at 30.4%. Full+Skills obtains the strongest SpreadsheetBench result at 40.0%, but even this remains far below its performance on the corporate subsets. The contrast indicates that aggregate results should be reported together with benchmark-family breakdowns.

5.3 Discussion

RQ1: business task coverage. The combined dataset exposes a broader task surface than either public source alone. FinQA contributes financial reasoning, SpreadsheetBench contributes artifact-oriented workbook manipulation, and the corporate subset adds real operational QA and editing across nine business domains, including accounting, patent, and translation-oriented tables. The observed category gap also indicates that benchmark identity matters: SpreadsheetBench tasks are markedly harder than the corporate subsets in this sample. This does not prove that existing benchmarks are generally incomplete, but it supports the narrower claim that no single included benchmark

Toolset	Skills	QA	Editing	Corp. QA	Corp. Edit	FinQA	SpreadsheetBench
Minimal	no	78.3%	61.7%	83.3%	96.7%	73.3%	26.7%
Minimal	yes	76.7%	58.3%	86.7%	93.3%	66.7%	23.3%
Search	no	75.0%	63.3%	80.0%	93.3%	70.0%	33.3%
Search	yes	81.7%	66.7%	90.0%	100.0%	73.3%	33.3%
Spreadsheet	no	81.7%	65.0%	86.7%	100.0%	76.7%	30.0%
Spreadsheet	yes	80.0%	65.0%	90.0%	96.7%	70.0%	33.3%
Full	no	70.0%	60.0%	70.0%	96.7%	70.0%	23.3%
Full	yes	81.7%	68.3%	90.0%	96.7%	73.3%	40.0%

Table 4: Category-level success rates aggregated over three replicates. QA and Editing contain 60 episodes per configuration; each benchmark-family column contains 30 episodes per configuration.

covers all table types encountered in the studied company workflow.

RQ2: effective agent organization. The results do not support a simple “more tools is better” rule. Without skills, the Spreadsheet toolset outperforms the Full toolset by 8.3 percentage points. Specialized tools may reduce the burden of selecting among heterogeneous actions and may better match structured workbook operations. Search-oriented tools alone do not produce the strongest aggregate performance, although they may remain useful for long textual tables.

The best aggregate result comes from combining the Full toolset with skills. This is consistent with the hypothesis that a broad architecture can benefit from guidance about how to navigate its action space. In other words, tool availability and tool orchestration should be treated as separate design problems.

RQ3: value of skills. Aggregated over all tool profiles, skill-enabled configurations reach 347/480 successes (72.3%), compared with 333/480 (69.4%) without skills, an observed difference of 2.9 percentage points. Mean score increases from 0.7679 to 0.7940.

The effect is heterogeneous across tool profiles. Skills change success by -2.5 points for Minimal, $+5.0$ for Search, -0.8 for Spreadsheet, and $+10.0$ for Full. The Full contrast has an uncorrected exploratory $p = 0.0319$, but it does not pass the Bonferroni threshold of $\alpha = 0.0125$ for four profile-level comparisons. The results therefore support a profile-dependent interpretation rather than a universal benefit of skills.

Efficiency trade-offs. Across tool profiles, skills increase average episode time from 156.5 to 179.7 seconds, iterations from 7.25 to 7.42, and token us-

age from 52.2k to 80.4k. The overhead is strongly profile dependent. Minimal grows from 41.3k to 96.5k tokens, Search from 48.5k to 63.3k, Spreadsheet from 58.5k to 89.4k, and Full from 60.7k to 72.5k.

Full is the only profile in which skills improve aggregate success while slightly reducing both mean time (178.5 to 176.3 seconds) and iterations (7.02 to 6.95). Search also improves success without a meaningful latency increase. Thus, the practical value of skills depends on whether their quality gain offsets their additional prompt cost in a particular tool environment. Because the model is self-hosted, token-priced API cost is undefined; a complete cost analysis would join the recorded token and wall-time measurements with server-side infrastructure telemetry.

5.4 Failure Analysis

The hardest tasks are concentrated in FinQA and SpreadsheetBench. Across the eight configurations and three replicates, `finqa_qa_002`, `sb_edit_1845`, and `sb_edit_54242` obtain no successful episodes out of 24. Failures include incorrect formula dependencies, incomplete extraction from long tables, output-format mismatches, and workbook-preservation violations.

Binary success can obscure partial progress on strict editing tasks. For example, `sb_edit_13_1` has a mean cell-level score of 0.9566 but only 4/24 successful episodes because editing requires a perfect score of 1.0. Reporting both mean score and binary success is therefore necessary for interpreting workbook-editing performance.

6 Conclusion

We presented a three-replicate comparative study of eight table-processing agent configurations, covering 40 tasks and 960 agent-task episodes.

Full+Skills achieves the highest aggregate success rate of 75.0%, while Spreadsheet without skills is the strongest no-skill configuration at 73.3%. Without skills, the broad Full profile reaches only 65.0%, showing that access to a larger action space does not by itself guarantee better performance.

Skills provide a small positive aggregate difference of 2.9 percentage points, but their effect depends strongly on the tool profile. The largest positive contrast occurs for Full, while Minimal and Spreadsheet show no success-rate improvement. Procedural guidance should therefore be treated as an architecture-dependent orchestration mechanism rather than a universally beneficial prompt addition.

Future work should enlarge the number and diversity of unique tasks, evaluate additional models and organizations, isolate individual tool contributions, compare skill-loading and retrieval strategies, and measure instruction adherence directly. Releasing tool schemas, skills, evaluator specifications, and non-confidential task metadata would further improve reproducibility.

Limitations

- The experiment includes three replicates, which provide an initial view of stochastic stability but remain insufficient for precise population-level estimates. Replicates repeat the same fixed set of 40 tasks and therefore increase trajectory coverage rather than task-set diversity.
- The evaluation set has only 40 tasks and is purposively assembled. The corporate tasks come from one company and may not represent other industries or organizations.
- The eight configurations differ in tool availability, and the Full toolset has a larger action space. The study compares complete architectures rather than isolating every individual tool effect.
- Skills are deterministically injected into the prompt, but the study does not measure instruction-level adherence or identify which skill sections influence individual decisions.
- Success and score definitions are summarized in Section 5.1, but the full task-by-task evaluation protocol should be documented in a released benchmark protocol.

- The corporate workbooks cannot be fully released because of confidentiality constraints, limiting exact external replication.

Acknowledgments

This work supported by the Ministry of Economic Development of the Russian Federation (IGK 000000C313925P4C0002), agreement No139-152025-010.

References

- Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyu Zhou, and William Yang Wang. 2020. [TabFact: A large-scale dataset for table-based fact verification](#). In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*.
- Yibin Chen, Yifu Yuan, Zeyu Zhang, Yan Zheng, Jinyi Liu, Fei Ni, Jianye Hao, Hangyu Mao, and Fuzheng Zhang. 2025. [SheetAgent: Towards a generalist agent for spreadsheet reasoning and manipulation via large language models](#). In *Proceedings of the ACM Web Conference 2025 (WWW '25)*. Association for Computing Machinery.
- Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. [FinQA: A dataset of numerical reasoning over financial data](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711. Association for Computational Linguistics.
- Haoyu Dong, Jianbo Zhao, Yuzhang Tian, Junyu Xiong, Mengyu Zhou, Yun Lin, José Cambronero, Yeye He, Shi Han, and Dongmei Zhang. 2024. [Encoding spreadsheets for large language models](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 20728–20748. Association for Computational Linguistics.
- Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. [TaPas: Weakly supervised table parsing via pre-training](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333. Association for Computational Linguistics.
- Maha Karkout, Maria Andreevna Khodorchenko, Nikolay Alekseevich Butakov, and Denis Nasonov. 2026. [Efficient KG-augmented RAG with reusable graph community summaries](#). In *Proceedings of the Knowledge Graphs and Large Language Models Workshop (KG-LLM) @ LREC26*, pages 110–119, Palma, Mallorca (Spain). ELRA Language Resources Association (ELRA).

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. [Efficient memory management for large language model serving with PagedAttention](#). In *Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP)*.

Hongxin Li, Jingran Su, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. 2023. [SheetCopilot: Bringing software productivity to the next level through large language models](#). In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*.

Zeyao Ma, Bohan Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. 2024. [SpreadsheetBench: Towards challenging real world spreadsheet manipulation](#). In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024), Datasets and Benchmarks Track*.

Egor Nevezhin, Nikolay Butakov, Maria Khodorchenko, Maxim Petrov, and Denis Nasonov. 2020. [Topic-driven ensemble for online advertising generation](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2273–2283, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. [Compositional semantic parsing on semi-structured tables](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Qwen Team. 2026. [Qwen3.6-35B-A3B model card](#). Accessed August 2026.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. [Executable code actions elicit better LLM agents](#). In *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. [ReAct: Synergizing reasoning and acting in language models](#). In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. [TaBERT: Pretraining for joint understanding of textual and tabular data](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426. Association for Computational Linguistics.

A Model and Serving Configuration

All eight configurations use the identical model deployment and decoding configuration. The model is Qwen3.6-35B-A3B (Qwen Team, 2026),

a mixture-of-experts model run in its instruct (non-thinking) configuration with native tool calling, served through an OpenAI-compatible endpoint by vLLM 0.19.1 (Kwon et al., 2023) in bf16 precision on two H200 GPUs. Table 5 lists the decoding and client parameters.

Parameter	Value
Temperature	0.7
Top- p	0.8
Top- k	20
Min- p	0.0
Presence penalty	1.5
Repetition penalty	1.0
Max output tokens	32,768
Context window	262,144 tokens
Thinking mode	disabled
Tool-call mode	native, tool_choice=auto
Request timeout	600 s
Max API retries	3

Table 5: Decoding and client parameters shared by all configurations.

The execution environment is offline (no Internet access) and runs Python 3.10.20 with pandas 2.3.3, openpyxl 3.1.5, and numpy 2.2.6. Each episode is limited to 100 agent-loop iterations; no time or token limits are imposed. Within each replicate, the task order is shuffled with a fixed seed shared by all eight configurations, and no API retries occurred in any run.

B Token Usage Breakdown

Table 6 reports per-episode mean prompt tokens, completion tokens, and tool calls for each configuration, aggregated over all three replicates. Prompt tokens dominate total usage; skills increase prompt size substantially because the selected Markdown files are injected into the system prompt, while completion counts remain within a narrow band across configurations.

Toolset	Skills	Prompt	Compl.	Tool calls
Minimal	no	39,132	2,183	5.72
Minimal	yes	93,600	2,946	6.80
Search	no	45,956	2,497	6.49
Search	yes	60,817	2,455	6.21
Spreadsheet	no	56,095	2,420	6.38
Spreadsheet	yes	86,545	2,886	6.72
Full	no	57,821	2,833	6.02
Full	yes	69,819	2,668	5.95

Table 6: Per-episode mean prompt tokens, completion tokens, and tool calls aggregated over three replicates.