ScieRxiv
NOT PEER-REVIEWED
Original Article

Does the wiki grow smarter? A pilot evaluation of LLM-maintained knowledge bases in an exception-dense decision-support task

Bailing Zhang[1*]

[1]Institute of Data Science and Intelligent Technology, NingboTech University, Ningbo 315100, Zhejiang, China.

Correspondence to: Dr.Bailing Zhang. Institute of Data Science and Intelligent Technology, NingboTech University, Ningbo 315100, Zhejiang, China. E-mail: bailing.zhang1961@gmail.com

Received: 27 April 2026 | Approved: 12 May 2026 | Online: 12 May 2026

Abstract

Karpathy's recent proposal of an LLM Wiki, a persistent markdown knowledge base maintained by an LLM itself, promises accumulation of knowledge across ingest and queries as an alternative to retrieval-augmented generation (RAG). Whether this pattern delivers a measurable benefit over RAG in practice is an open empirical question. We evaluate it in wastewater anomaly diagnosis, a task in which most errors arise from missed exceptions and contraindications rather than from missed main rules. We construct a corpus of ten operating manuals, a stream of forty batched training cases, and a held-out test set of fifty cases stratified by six exception subtypes. We then build four systems ranging from a strong RAG baseline to a Wiki with compiled rule-checking views, and iterate the Wiki implementation three times with substantially different ingest policies. Across all three iterations and all six exception subtypes, at the scale tested, no Wiki variant exceeds the RAG baseline on either diagnosis accuracy or forbidden-action recall. We characterize three degenerate maintainer behaviors observed in the experiments, namely fragmentation, atomization, and over-merging, and

discuss what the negative result does and does not imply for the Karpathy pattern.

Keywords: LLM agents, knowledge management, retrieval-augmented generation, exception handling, decision support, negative result

INTRODUCTION

Retrieval-augmented generation (RAG) is the default approach for grounding a large language model's (LLM's) answers in a document collection: at query time, a retriever locates passages in raw sources and the model conditions on them[1]. Recent work has explored alternatives that precompile the corpus into a richer intermediate representation, such as a graph of entities and summaries[2] or an evolving agent memory organized by notes and links[3]. A simpler proposal from Karpathy takes a different route: let the LLM itself maintain a human-readable markdown wiki that sits between the raw sources and the user, with the LLM performing ingest, cross-linking, and lint-style health checks[4]. The central claim is accumulation: the wiki grows richer as sources and queries arrive, and each query benefits from all prior bookkeeping.

Whether this claim holds in practice, and under what conditions, is an open empirical question. The appeal of the pattern rests on an implicit model of the maintenance loop in which the LLM behaves as a disciplined editor: updating relevant pages rather than appending duplicates, condensing overlapping claims rather than fragmenting them, and reorganizing when a new source challenges an old one. None of these behaviors are guaranteed by the design; they depend on the prompts and workflows used to drive the LLM, and on the corpus. Karpathy's gist describes the pattern in prose; individual practitioners have reported mixed results and early community implementations flag quality-control as a known failure mode.

This paper asks a specific, falsifiable version of the question: *In an exception-dense decision-support task, does a naive LLM-maintained wiki exceed a strong RAG baseline?* We focus on exception-dense domains because they are where accumulation might matter most: the difficulty lies in knowing when not to apply the default rule, which treatments are contraindicated in which contexts, and which prerequisites have not yet been satisfied. These are the cross-referential facts a persistent wiki is supposed to surface.

Contributions

Our contribution is a pilot evaluation rather than a new architecture. We implement the Karpathy pattern three times with progressively different ingest strategies, evaluate each on a held-out test set stratified by exception subtype, and report what we found:

•A wastewater-anomaly-diagnosis task with dual-dimension ground truth (root-cause diagnosis and forbidden-action identification), stratified along exception subtype and difficulty.

•Four comparison systems (S0-S3) whose differences isolate the contribution of wiki compilation, file-back, and compiled rule-checking views.

•Three iterations of the ingest-and-maintain pipeline that exhibit distinct failure modes: fragmentation, atomization, and over-merging.

•A 50-case stratified evaluation showing that no iteration and no exception subtype yields a Wiki system that exceeds the RAG baseline at the scale tested.

•A discussion of what this result does and does not imply for the Karpathy pattern, and a short set of diagnostic signatures that future implementations can use to detect the three failure modes early.

EXPERIMENTAL

Task and data

The task is to propose a root-cause diagnosis for an observed anomaly in a wastewater treatment plant and to enumerate the treatment actions that should not be taken in the current context. Each case presents a short observation in Chinese, such as: SVI = 195 mL/g, MLSS = 1,300 mg/L, pH = 7.1, microscopy shows heavy filamentous bacteria, the operator requests chlorination.

The corpus consists of ten Chinese-language standard operating procedures (SOPs) covering activated sludge operations, sludge bulking, dissolved oxygen control, pH management, temperature and seasonality, nitrification, phosphorus removal, influent

shocks, sludge wastage, and common misjudgments. These documents are provided to all systems. A stream of forty training cases is organized into eight batches of five cases each; each Wiki system ingests the batches in order.

A held-out test set of fifty cases is stratified along two axes. By exception subtype: 14 hard-forbidden, 9 mutual-exclusion, 9 prerequisite, 6 temporal-order, 4 probabilistic, 7 narrative-only, and 1 normal-operation case. By difficulty: 10 easy, 22 medium, 18 hard. Each test case carries dual-dimension ground truth: a primary diagnosis with acceptable alternatives, and a list of forbidden actions with reason and severity labels. Both the training stream and the test set are synthetic but constructed to reflect the kinds of compound conditions and misjudgments documented in the SOPs.

Four systems

We evaluate four systems with progressively richer structure.

S0 (RAG-strong) uses a BM25-style retriever over the raw SOPs and the forty batched training cases, combined with an LLM prompt that asks for root-cause candidates, performs an explicit exception-screening step, and emits a list of forbidden actions. There is no persistent Wiki. S0 is intentionally a strong baseline: we give it the same structured prompting that the Wiki systems use for the query step, so any advantage the Wiki systems obtain is attributable to the Wiki itself rather than to prompting.

S1 (Wiki-only) compiles a Wiki from the SOPs and ingests batches one at a time. Queries retrieve from the Wiki rather than the raw sources. Query answers are not written back into the Wiki.

S2 (Wiki-full) is S1 plus a file-back step: after each query, a decision module may write a new Wiki page (a case or misjudgment entry) that becomes retrievable for subsequent queries. S2 is the canonical Karpathy pattern.

S3 (Wiki + compiled views) is S2 plus a compiled-view layer that converts a subset of exception pages into symbolic rule stubs. For each retained candidate, these stubs are checked before the LLM is allowed to retain it. This is the variant closest in spirit to our group's downstream constraint engine, but implemented entirely in Python.

The Wiki is stored as a flat directory of markdown files with YAML front matter. Pages are grouped by type: concept, exception, case, misjudgment, and anomaly_pattern. Retrieval at query time uses tag and keyword matching rather than embeddings, consistent with the pattern described in Karpathy's gist[4].

Metrics

We report four metrics computed per test case and aggregated by mean with 95% bootstrap confidence intervals over the 50 test cases (1000 resamples, seed 42).

Diagnosis-Top-1 is 1 if the first retained candidate matches the primary diagnosis or any acceptable alternative under a normalized substring match that ignores whitespace and punctuation, 0 otherwise.

Diagnosis-Top-3 is 1 if any of the top three retained candidates matches, 0 otherwise.

Forbidden-Action-Recall (hard) (FAR-hard) is the fraction of hard-severity forbidden actions that the system flags. An action is considered flagged if it appears in the system's flagged_actions output or if any candidate's triggered_exceptions field names the action by fuzzy match. This is the metric most directly tied to safety value.

Exception-Tag-Recall (ETR) is the fraction of the case's expected_exception_tags that appear in the system's output. It is a secondary metric measuring whether the system explicitly references rule names.

Evaluation protocol

All LLM calls use glm-4-flash via the Zhipu API at temperature 0. Each batch is ingested in case order; evaluation runs at five checkpoints (batches 0, 2, 4, 6, 8). All four systems are evaluated on identical test cases at identical checkpoints. Total LLM-call counts and token usage per run are reported in Table 1.

Implementation and three iterations

The three iterations share a high-level pipeline. Ingest reads a source document and decides which pages to create or update. Query searches the Wiki, generates candidates,

screens each candidate against relevant exception pages, and flags forbidden actions. File-back, when enabled, writes the query's conclusion as a new page. The differences between iterations lie entirely in the prompts driving ingest and, in v3, in a programmatic duplicate-detection guard on page creation.

Iteration v1: Fragmentation

Design. The v1 ingest prompt asks the LLM to extract relevant facts and decide, for each, whether it updates an existing page or seeds a new one. No constraint is placed on page naming, tag usage, or duplicate detection.

Outcome. After eight batches the S2 Wiki contained 98 concept pages, 58 exception pages, 35 case pages, and 0 misjudgment pages, totaling 191. Among the 58 exception pages, a simple title-similarity check (≥ 0.6 after normalization) finds 14 near-duplicate pairs forming clusters, the largest of which contains three pages all describing variants of the same pH-related chlorination restriction. Ingest logs show 50 ingest operations each of which created at least one new page; not a single ingest produced zero new pages. Pages remain narrow: the mean exception-page body is 676 bytes at batch 8, up from 557 at batch 0.

Iteration v2: Atomization

Design. v2 rewrote the extraction prompt to emphasize typed claims: hard_forbidden, mutual_exclusion, prerequisite, temporal_order, probabilistic, or exception. The update prompt asks the model to append the new atomic fact to the most relevant existing page rather than rewrite the page body. No dedupe guard is added.

Outcome. At batch 8 the S2 Wiki held 88 concept, 71 exception, and 38 case pages (total 197). The mean exception body (664 bytes) is similar to v1, but the content pattern differs: bodies become append-only logs of short sentences, each prefixed by an Update from batch_X header. Over 50 ingests, zero produced "+ 0 new" log lines, that is, each ingest created at least one new page. The near-duplicate pair count in the exception layer rose to 27 (largest cluster of 6).

Iteration v3: Over-merging

Design. v3 aimed to correct v2's append-only behavior. The extract prompt was revised

to produce full rule units with a four-tuple (trigger, rule, mechanism, alternative). The update prompt was told to rewrite the full page body on update rather than append. A programmatic duplicate guard blocks creation of any new page that shares at least two tags with an existing page and whose normalized title similarity exceeds 0.55.

Outcome. The combined prompt-plus-guard change produced the opposite failure. By batch 8 the S2 exception layer had collapsed to a single page (8,171 bytes) aggregating material from 19 distinct ingest anchors, covering chlorination, pH, MLSS, low temperature, toxic shock, and microscopy-related rules. Of 49 ingests attempted for the S2 system, 46 resulted in zero new pages (fraction merge-only = 0.939). The collapsed page shows a further behavior not anticipated by the design: rather than truly merging new material into existing mechanism text, the LLM simply appended new rule units under headers such as "New information: trigger… rule…" while leaving the prior body untouched. In effect, merging reduced to append-to-single-page rather than genuine consolidation. Duplicate segments and outright contradictions appear in the collapsed page: for the same scenario "NH3-N rises from 2.1 to 8.5", the page contains both "increasing wastage makes things worse" and "one should increase wastage to extend SRT" without any indication that these are in conflict.

RESULTS AND DISCUSSION

Wiki state across iterations

Table 1 summarizes the state of the S2 system at batch 8 across the three iterations. Page counts and duplicate statistics cleanly separate the three failure modes: v1 produces many small pages with clusters of near-duplicates; v2 produces a similar page count but shifts the duplication into per-page append logs; v3 collapses the exception layer entirely.

Table 1. Wiki state at batch 8 under the S2 system across three iterations. "+ 0 new" is the number of ingest operations out of 50 that created no new pages. Near-duplicate pairs are exception pages whose titles have normalized similarity ≥ 0.6. Runs are labeled by compiler version and test-set size (10 or 50 cases).

Statistic	v1/10case	v2/10case	v3/10case	v2/50case

Statistic	v1/10case	v2/10case	v3/10case	v2/50case
Concept pages	98	88	2	90
Exception pages	58	71	1	69
Case pages	35	38	40	190
Total pages	191	197	43	349
Mean exception-page body (bytes)	676	664	8,171	765
Near-duplicate pairs in exception layer	14	27	0 (N/A)	28
Largest near-duplicate cluster	3	6	1	8
Ingests (of 50) with +0 new pages	0	0	46/49	0
Total LLM calls in run	1,443	1,654	1,361	6,381

Cross-iteration comparison on the 10-case test set.

Table 2 gives the state of all four systems at batch 8 on the original 10-case test set under each Wiki iteration. S2 does not exceed the baseline S0 on Diagnosis-Top-1 in any iteration. On Forbidden-Action-Recall (hard), the gap widens under v3.

Table 2. Batch-8 results on the 10-case test set across Wiki iterations. Bootstrap 95% confidence intervals over test cases are shown where recorded. The v1 run used an earlier logging format without bootstrap CIs; only point estimates are shown for that row.

Iteration	System	Top-1 mean	Top-1 95% CI	FAR-hard mean	FAR-hard 95% CI
v1 (Fragmentation)	S0 (RAG)	0.200	n/r	n/r	n/r
	S1	0.000	n/r	n/r	n/r

Iteration	System	Top-1 mean	Top-1 95% CI	FAR-hard mean	FAR-hard 95% CI
	S2	0.000	n/r	n/r	n/r
	S3	0.000	n/r	n/r	n/r
v2 (Atomization)	S0 (RAG)	0.400	[0.10, 0.70]	0.375	[0.21, 0.55]
	S1	0.200	[0.00, 0.50]	0.258	[0.09, 0.43]
	S2	0.400	[0.10, 0.70]	0.267	[0.12, 0.43]
	S3	0.300	[0.10, 0.60]	0.258	[0.11, 0.43]
v3 (Over-merging)	S0 (RAG)	0.300	[0.00, 0.60]	0.383	[0.19, 0.58]
	S1	0.100	[0.00, 0.30]	0.308	[0.15, 0.47]
	S2	0.200	[0.00, 0.50]	0.167	[0.03, 0.33]
	S3	0.200	[0.00, 0.50]	0.225	[0.09, 0.38]

Several observations are worth drawing out explicitly. First, the absolute Diagnosis-Top-1 range is narrow (0.0-0.4). With ten test cases, a single case shifting classification corresponds to 0.1 in point estimate. Confidence intervals therefore span 0.3-0.7 width and overlap heavily between adjacent systems. Second, even allowing for this uncertainty, no Wiki system beats S0 in any iteration. Third, v3 shifts S2's FAR-hard down by 0.10 relative to v2, that is, the over-merging behavior is actively harmful, not neutral. Fourth, S3 (compiled views) neither hurts nor helps substantially over S2, suggesting the symbolic check layer has little to add when the underlying Wiki content is itself compromised.

50-case stratified evaluation

Table 2 calls for higher statistical power and for a question finer than "does the Wiki help overall?", specifically, "does the Wiki help on any class of exceptions?" The 50-case test set, stratified by exception subtype and difficulty, addresses both. All results below use the v2 compiler (the most neutral of the three).

Overall performance. At batch 8, S0 leads on both main metrics [Table 3]. The Diagnosis-Top-1 difference S0-S2 of 0.04 is within overlapping confidence intervals, so we do not claim statistical significance. The FAR-hard difference of 0.047 is likewise within CI overlap. What the stratified numbers below establish is the stronger claim that no subgroup shows a Wiki advantage, not that the overall gap is significant.

Table 3. Overall metrics at batch 8 on the 50-case test set (v2 compiler). Bootstrap 95% CIs over test cases. Total run token usage was 9.95 M prompt + 0.94 M completion across all systems.

System	Top-1		Top-3		FAR-hard		ETR	
S0 (RAG-strong)	0.100	[0.02, 0.20]	0.120	[0.04, 0.22]	0.244	[0.16, 0.34]	0.000	[0.00, 0.00]
S1 (Wiki-only)	0.020	[0.00, 0.06]	0.060	[0.00, 0.14]	0.184	[0.10, 0.28]	0.023	[0.00, 0.05]
S2 (Wiki-full)	0.060	[0.00, 0.14]	0.080	[0.02, 0.16]	0.197	[0.11, 0.29]	0.023	[0.00, 0.05]
S3 (Wiki + CV)	0.040	[0.00, 0.10]	0.060	[0.00, 0.14]	0.189	[0.10, 0.29]	0.023	[0.00, 0.05]

By exception subtype. Tables 4 and 5 report diagnosis and forbidden-action metrics stratified by the six exception subtypes. At no point does any Wiki system exceed the S0 baseline by more than 0.05 in any cell of either table; in most cells the Wiki systems are equal or below S0. The narrative-only subtype, in which correct diagnosis requires contextual reasoning rather than a rule lookup, is the only one where Wiki systems tie or slightly approach S0 (Top-1 0.286 for S0 and S2, 0.143 for S1 and S3).

Table 4. Diagnosis-Top-1 stratified by exception subtype at batch 8 (50-case test set). Numbers in parentheses are per-stratum n. h._forbid: hard_forbidden; mutex: mutual_exclusion; narr._only: narrative_only; prereq.: prerequisite; temp._ord.: temporal_order.

System	Overall	h._forbid	mutex	narr._only	normal	prereq.	temp._ord.
S0	0.100	0.214 (14)	0.000 (9)	0.286 (7)	0.000 (1)	0.000 (9)	0.000 (6)
S1	0.020	0.000 (14)	0.000 (9)	0.143 (7)	0.000 (1)	0.000 (9)	0.000 (6)
S2	0.060	0.071 (14)	0.000 (9)	0.286 (7)	0.000 (1)	0.000 (9)	0.000 (6)
S3	0.040	0.071 (14)	0.000 (9)	0.143 (7)	0.000 (1)	0.000 (9)	0.000 (6)

Table 5. Forbidden-Action-Recall (hard) stratified by exception subtype at batch 8. Numbers in parentheses are per-stratum n.

System	Overall	h._forbid	mutex	narr._only	normal	prereq.	temp._ord.
S0	0.244	0.280 (14)	0.365 (8)	0.533 (5)	0.000 (1)	0.139 (9)	0.000 (6)
S1	0.184	0.161 (14)	0.385 (8)	0.333 (5)	0.000 (1)	0.083 (9)	0.000 (6)
S2	0.197	0.179 (14)	0.385 (8)	0.400 (5)	0.000 (1)	0.083 (9)	0.000 (6)
S3	0.189	0.208 (14)	0.385 (8)	0.400 (5)	0.000 (1)	0.000 (9)	0.000 (6)

By difficulty. On the difficulty axis, all four systems score near-chance on every difficulty level. S0 has Top-1 of 0.100, 0.091, and 0.111 on easy, medium, and hard. S2 scores 0.000, 0.045, and 0.111 respectively. No Wiki system exceeds S0 in any difficulty bucket.

Learning curves. Figure 1 shows the full learning curves on the 50-case test set. The S2 Diagnosis-Top-1 trajectory visits {0.06, 0.06, 0.08, 0.04, 0.06} across batch checkpoints 0, 2, 4, 6, 8, that is, flat within noise as the Wiki grows from 51 to 321

pages. S0 shows a one-step bump from batch 0 (where no training cases are indexed) to batch 2, then plateaus. None of the four systems shows a consistent upward trend attributable to accumulating knowledge.

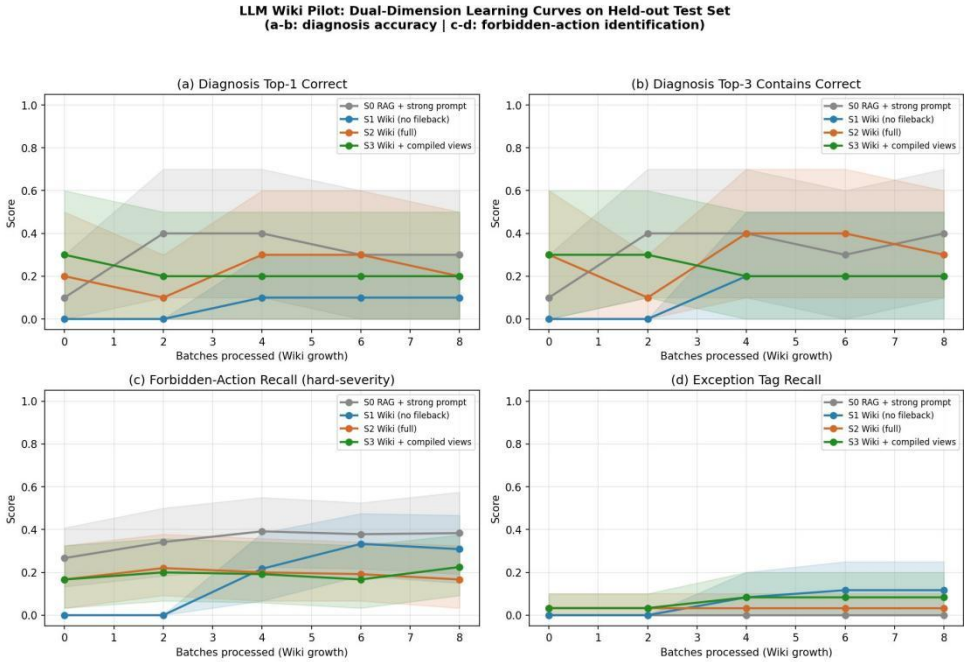

Figure 1. Learning curves on the 50-case held-out test set under the v2 compiler. Panels: (A) Diagnosis-Top-1; (B) Diagnosis-Top-3; (C) Forbidden-Action-Recall (hard severity); (D) Exception-Tag-Recall. Shaded regions are 95% bootstrap confidence intervals over test cases. Four systems plotted (S0 RAG-strong; S1 Wiki-only; S2 Wiki-full; S3 Wiki + compiled views). Abbreviations: RAG, retrieval-augmented generation; S0, RAG-strong (strong RAG baseline with no wiki); S1, Wiki-only; S2, Wiki-full (Wiki with file-back); S3, Wiki with compiled views; v2, second iteration of the ingest-and-maintain pipeline; CI, confidence interval.

Three degenerate maintainer behaviors

The most transferable observation from this work is the characterization of three ways in which the LLM-as-maintainer loop misbehaves. In v1, with a permissive ingest prompt, the LLM prefers to create new pages over updating existing ones; we call this fragmentation. In v2, with a prompt that asks for typed atomic facts, the LLM appends those facts to the most relevant page, producing pages that are append-only logs of near-duplicate sentences; we call this atomization. In v3, with a merge-oriented prompt and a duplicate-detection guard, the LLM collapses the entire exception layer into a single page that accretes material across ingests without reorganization; we call this over-

merging. We conjecture that these modes constitute the corners of a prompt-design trade-off and that a healthy wiki lives at an equilibrium that is not the natural resting state of any one prompt we tried.

Merging versus appending

The v3 collapsed page reveals a subtler failure beneath over-merging: the LLM, although instructed to produce a new full-page body that merges new information, in practice appended new rule units at the page's end under headers such as "New information: trigger… rule…" without revising the existing text. In one case the page contains two contradictory recommendations for the same observation (increase versus do not increase sludge wastage given a rise in effluent ammonia), because they were appended at different batches. The operation nominally changed from "append a line" (v2) to "rewrite the page" (v3), but functionally the LLM performed "append to the single page" in both cases. This suggests that future Wiki maintenance research should explicitly evaluate whether merging actually happened, not only whether a merge-oriented prompt was used.

Implicit context contamination

A smaller observation: in v1, the Wiki page for a general rule ("chlorination is forbidden at low pH") accumulated an update from a specific training case whose observed pH was in the normal range. The update was phrased as a qualification of the rule ("pH in normal range, therefore the contraindication does not apply"), which then became part of the retrieved context for every subsequent query touching that rule. We did not quantify the downstream effect of this, but we note it as a failure mode distinct from the three above: particular cases' reasoning leaking into general rules.

What the result does and does not say

The negative result supports a narrow conclusion: under our conditions (glm-4-flash, Chinese wastewater SOPs, 10 documents, 40 batched training cases, 50 test cases, three distinct ingest-prompt families, a single random seed), naive LLM-maintained wikis do not outperform a structured RAG baseline. It does not support a general claim that the pattern is without value. We did not test frontier models, a human curator in the loop, longer time horizons, or real plant data, and any of these could change the picture. The result also does not speak to structured-retrieval alternatives such as GraphRAG or

agent-memory systems such as A-MEM, which differ from the markdown-wiki pattern in both artifact and maintenance loop.

Why the baseline is hard to beat here

Two properties of our setup probably contribute. First, the corpus is small (ten SOPs) and a BM25-style retriever over the raw sources already places the relevant pages in context for most queries, leaving little knowledge gap for a wiki to fill. Second, the Wiki systems and the baseline share the same candidate-generation and exception-screening prompt, isolating the Wiki's contribution to retrieval structure; at this corpus size, retrieval structure is not the binding constraint. We expect the gap between a naive Wiki and RAG to widen on much larger corpora or on queries that require cross-document synthesis, but did not test that regime.

LIMITATIONS

Model: glm-4-flash is a competent but not frontier model; its tendency toward append-style edits may be weaker in stronger models.

Sampling: one trajectory per configuration at temperature 0; variation estimates come from test-case bootstrap only, not from sampling across LLM runs.

Corpus: ten synthetic SOPs plus 40 synthetic training cases in Chinese.

Metric assumptions: the fuzzy-match scorer for diagnosis uses normalized substring match, which is lenient; the forbidden-action recall metric depends on the GT list being reasonably complete.

Downstream module: the original research program also includes a symbolic constraint engine downstream of the Wiki; this paper evaluates only the Wiki component.

CONCLUSIONS

We set out to test whether the LLM Wiki pattern exceeds a strong RAG baseline in an exception-dense decision-support task. Across three implementation iterations, four comparison systems, and a 50-case test set stratified by six exception subtypes, no Wiki

configuration exceeded the baseline at the scale tested. We characterized three failure modes of the LLM-as-maintainer loop: fragmentation, atomization, and over-merging, that any future implementation of the pattern in a similar domain should expect to encounter.

Future work

The most natural next steps are: extending the ingest loop with a human curator in a lightweight review role, enlarging the corpus beyond ten SOPs to see whether the retrieval-structure advantage materializes at scale, testing frontier models, and connecting the Wiki to the symbolic constraint engine foreshadowed in our research program.

DECLARATIONS

Authors' contributions

The author contributed solely to the article.

Availability of data and materials

Code, prompts, the synthetic SOP corpus, the 40-case training stream, the 50-case stratified test set, all four run outputs (snapshots and per-batch evaluation results), and analysis scripts will be released as a reproducibility archive alongside the published version of this paper.

Financial support and sponsorship

To be declared on final submission.

Conflicts of interest

All authors declared that there are no conflicts of interest.

Ethical approval and consent to participate

Not applicable. The study uses synthetic operational scenarios and does not involve human subjects, human material, or human data.

Consent for publication

Not applicable.

REFERENCES

1. Lewis P, Perez E, Piktus A, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Adv Neural Inf Process Syst* 2020;33:9459-74.[DOI: 10.48550/arXiv.2005.11401]

2. Edge D, Trinh H, Cheng N, et al. From local to global: a graph RAG approach to query-focused summarization. *arXiv* 2024;arXiv:2404.16130.[DOI: 10.48550/arXiv.2404.16130]

3. Xu W, Liang Z, Mei K, Gao H, Tan J, Zhang Y. A-MEM: agentic memory for LLM agents. *Adv Neural Inf Process Syst* 2025.[DOI: 10.48550/arXiv.2502.12110]

4. Karpathy A. LLM Wiki [Internet]. GitHub Gist; 2026 Apr [cited 2026 Apr 20]. Available from: https://gist.github.com/karpathy/442a6bf555914893e9891c11519de94f