

Language Model Memory Flattens the Temporal Form of User Facts

Sugam Panthi, Muhaiminul Yeamin, Siyan Luo, Rabab Abdelfattah

AIMS Lab, The University of Southern Mississippi

{sugam.panthi, muhaiminul.yeam, siyan.luo, rabab.abdelfattah}@usm.edu

Abstract

Long-term memory systems store conversations as short notes that later models read. A note can turn progressive “I am working at Corvida” into simple-present “The user works at Corvida,” dropping the cue that the job may be temporary. We call this *aspectual flattening* and measure it with LAPSE, a benchmark of matched user statements that differ only in temporal form. Three writer models flattened the progressive statement but kept its simple-present match in 210 of 381 pairs, never the reverse. This one-way loss held in all 11 models tested and in the memory systems mem0, Graphiti, and Letta. In exploratory tests, reader models judged a fact less likely to still hold when only its stored verb was progressive. In a planned test, the reader GLM-5.2 acted without asking the user on 64% of “works” notes but 40% of “is working” notes. Gemini 3 Flash showed a smaller gap, one reader added later showed the reverse, and others showed none. In a further exploratory test, free-text notes erased this difference and notes that kept the user’s words restored it. Memory systems should keep the user’s wording as evidence for later readers. Code and data: <https://github.com/aimsresearchlab/lapse>.

1 Introduction

Consider a user who says, “I am working at Corvida.” A memory writer, the model or pipeline step that turns a conversation into stored notes, may store *The user works at Corvida*. The person and employer survive, but the cue that the job may be temporary does not. The model that later reads the note never sees the conversation, so the note is the only message it receives. Months later, that reader has less evidence for deciding whether to check the fact first. We call this transformation *aspectual flattening*.

Memory systems already address storage, retrieval, decay, contradictions, and timing (Packer et al., 2023; Wu et al., 2025; Chhikara et al., 2025). A timestamp records when the user spoke, but not whether the user presented the job as temporary. A writer can keep the timestamp while erasing that distinction.

Aspect is how a sentence presents the internal temporal structure of a situation, for example as ongoing or as a standing state (Vendler, 1957; Comrie, 1976; Friedrich et al., 2023). Moens and Steedman (1988) argued that aspect can change the temporal category of a proposition, and that any usable temporal database queried in natural language must represent such categories. A long-term memory store is one such database, and its notes are messages from one model to another. We therefore treat memory writing as communication between models and ask whether the note keeps the aspectual evidence that its reader would need.

Testing this question requires a comparison that changes the wording without changing the situation. We build LAPSE (Linguistic Aspect Persistence and Stability Evaluation), a controlled benchmark of matched user statements (Table 1). Within a pair, the surrounding words, situation, and elapsed time stay fixed; only the temporal form changes. Controls test two other explanations for an apparent effect: a model’s general tendency to act on any stored fact, and content that sounds temporary on its own. We fixed the main test and its analysis before collecting data. After a human check of the labels, we stopped counting notes that keep the progressive but add “currently” as a loss; the test passes either way (Section 4.3).

Flattening is a miscommunication in which nothing false is said. “The user works at Corvida” is true when written, but it no longer shows how the

user presented the job, and the reader cannot ask the writer what was meant. A miscommunication can arise on the sender side or the receiver side, so we ask two separate questions (Figure 1): does the writer keep the source cue, and does the reader use the cue that survives? Scoring the reader’s final behavior alone could not show which side lost the cue. The form of the message is also a design choice, so we compare a free-text note with a structured record that keeps the user’s words (Section 7.4).

Our contributions are:

- **Construct and benchmark.** We name *aspectual flattening* and release LAPSE, a benchmark of matched user statements that differ only in temporal form, with its stimulus builder, model outputs, labels, and analysis programs (<https://github.com/aimsresearchlab/lapse>).
- **Write-time result.** Memory writers flatten the progressive while keeping the matched simple present in 210 of 381 pairs, and never the reverse. The pattern holds in all 11 models tested and in the released memory systems mem0, Graphiti, and Letta.
- **Read-time evidence.** In exploratory tests, changing only the stored verb to the progressive lowers readers’ estimates that a fact still holds. In a planned test, GLM-5.2 and Gemini 3 Flash then act without asking less often, and one reader added later does the reverse. In an exploratory test, free-text notes erased this difference and records that kept the user’s words restored it.

2 Related Work

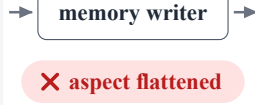
Long-term agent memory. Generative Agents, MemoryBank, MemGPT, RecallM, THEANINE, A-MEM, Mem0, and Graphiti store and retrieve long-lived information in different forms (Park et al., 2023; Zhong et al., 2024; Packer et al., 2023; Kynoch et al., 2023; Ong et al., 2025; Xu et al., 2025; Chhikara et al., 2025; Rasmussen et al., 2025). Their temporal mechanisms include timestamps, recency, decay, graph chronology, and consolidation. Recent systems also build semantic timelines or attach lifecycle relations and policies to extracted facts (Su et al., 2026; Yacoubi et al., 2026; Mullick and Tüzün, 2026). These methods reason over stored events, metadata, or relations among memories. LAPSE asks an earlier question: does memory writing preserve temporal evidence already present in the user’s words?

Long-context and temporal evaluation. Lo-CoMo and LongMemEval test long-horizon recall, temporal reasoning, updates, and abstention (Maharana et al., 2024; Wu et al., 2025). Their user turns include temporary-form statements, but we found no question that requires noticing that form (Appendix G). Which stored form receives retrieval credit can also change conclusions drawn from these benchmarks (Panthi and Abdelfattah, 2026b). TVCP predicts validity from historical text and timestamps (Wenzel and Jatowt, 2024), while Chronocept evaluates temporal-concept understanding (Goel et al., 2026). Most closely, STALE studies stale personalized knowledge after a later observation changes the world (Chao et al., 2026), and StateAuditor repairs downstream responses from timestamped old-to-new transitions (Sun and He, 2026). Memora evaluates obsolete or invalidated memories as user circumstances change (Uddin et al., 2026), while MemStrata retires a stale value after a newer assertion contradicts it (Yadav, 2026). Those settings provide a contradiction or later update. LAPSE instead holds the history fixed and changes only the source wording, so any lost cue is removed by the write step itself.

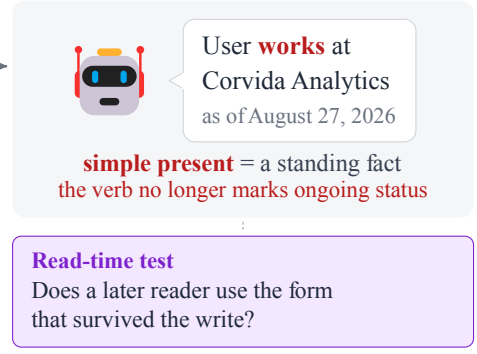
Qualification and relation loss. Manufactured Confidence shows that consolidation can strip epistemic hedges and turn tentative reports into facts that guide later action (Kwon, 2026). TANGLE likewise finds that end-to-end memory extraction can lose relations needed to interpret unresolved conflicts, and separates extraction from downstream reasoning with pipeline and oracle tracks (Yang et al., 2026). Both show that rewriting can erase decision-relevant qualification. Outside memory, fixed RAG compression also drops details that stronger readers would have used (Panthi and Abdelfattah, 2026a). The memory studies’ cues are hedge words such as “probably” and “reportedly” or relations between facts; LAPSE tests a cue carried by the verb form alone. Losing that cue is not the same as losing lexical qualifiers: one writer we test, GPT-5.6 Luna, kept “for now” and “at the moment” more often than the progressive (Table 15).

Aspect as a model diagnostic. Ma and Miyao use the imperfective paradox to test whether models infer that a past progressive event reached its endpoint (Ma and Miyao, 2026). There, the original premise is available when the model reasons. LAPSE asks whether a memory writer keeps present-progressive evidence before a later reader

1 · SOURCE UTTERANCE



2 · STORED MEMORY



separate tests
no mediation assumed

Figure 1: LAPSE separates what a writer stores from how a reader uses it. Panels 1–2 show a note written by the released memory system mem0: the writer replaces progressive *working* with simple-present *works* but keeps a date.

Factor	Levels	Values (main test highlighted)	Varied to test
Form	5	progressive simple present bounded perfect progressive perfect simple	the manipulated variable
Frame	8	all eight: lodging workplace vehicle class household equipment affiliate role project	whether it holds across kinds of fact
Carrier	6	life update after a request bare statement mid-message formal note list item	whether the surrounding conversation matters
Gap	6	fresh (2 days) near (42 days) boundary expired-soon stale (245 days) stale-long	fresh and boundary serve as positive controls
Task	7	<i>write:</i> memory write guided memory write <i>use:</i> direct use memory use guided memory use <i>ask:</i> validity question send-or-check choice	what is stored versus how it is used

Main test: progressive vs. simple present in all eight frames, with the life-update carrier, the stale gap, and the memory-write task: 128 matched pairs per configuration.

Table 1: The LAPSE design. An example pair is “I am working at Corvida” vs. “I work at Corvida”; both members use the same verb. Appendix A gives every template.

sees the note. A recent reanalysis attributes some apparent culmination errors to label mapping and lexical variation, and uses lexically matched minimal pairs (Han and Sun, 2026). Our same-lexeme control addresses the same concern, but our outcome is the stored note rather than a natural-language-inference judgment.

3 Temporal Form as Memory Evidence

Progressive form typically presents an event as ongoing. Simple present often supports a habitual or stative reading, and perfect constructions relate an earlier event to a later reference point (Comrie, 1976; Dowty, 1979). “I am working at Corvida” presents the job as ongoing and possibly temporary; “I work at Corvida” presents it as a standing state. Both cues are defeasible: context can make a progressive long-lived and a simple state short-lived. Our claim is narrow. When content and context are

held constant, a memory writer should not erase a form distinction that bears on how long the fact will hold.

A rewrite loses the cue when it removes the source’s overt marking of ongoingness or boundedness, or when it adds an unsupported fact that cancels that marking. A paraphrase may change person, drop politeness, or compress syntax and still keep the source’s temporal meaning, which is all the test asks.

4 The LAPSE Evaluation

4.1 Factorial design

The unit of analysis is a matched pair of user statements, each given to a model with the same memory-writing instruction. Within a pair, the topic, main verb, surrounding conversation (the *carrier*), and elapsed time are identical; only the temporal form changes. Table 1 summarizes the

design. Besides form, it varies the *frame*, the kind of fact a statement reports (for example, a workplace or a vehicle), along with the carrier, the time gap, and the task. These factors check that a result is not tied to one wording or setting.

The main test uses progressive–simple pairs in the life-update carrier at a stale gap of 245 days. Every source has an absolute timestamp. No form receives an extra temporal adverb such as “at the moment,” and an automatic check rejects pairs that differ outside the intended marking. Two gaps serve as positive controls: at the fresh gap (2 days) and the boundary gap (60 days, before any stated end month), a reader should still use the fact.

4.2 Write and read tasks

In the memory-write task, a model receives a short conversation and writes one stored note. Guided memory writing adds an instruction to preserve temporal qualification. In the memory-use task, a later model receives the resulting note and a user request. Two further tasks give the model the conversation itself. The validity question asks whether the fact is still safe to assume. The send-or-check choice asks whether the model’s reply can go out as written or whether it would first check a detail with the user; its results were null or degenerate (Appendix C.6).

We call each tested system a *model configuration* because the serving provider and settings can change a model’s outputs. Each complete configuration receives 6,008 requests. Seven complete configurations and four smaller frontier-model subsets yield 45,000 requests. The three confirmatory configurations were chosen before data collection; the other configurations test breadth and are not part of the confirmatory tests.

4.3 Scoring

Write outputs receive one of five labels. A FAITHFUL note preserves the source’s temporal meaning, a COERCED-STATIVE note renders a progressive or bounded source as a simple-present state, and a MANUFACTURE note adds “currently” or an undated progressive. The main outcome is an *aspect flip*: a progressive source flips when its note is COERCED-STATIVE, which is flattening, and a simple-present source flips when its note adds a progressive. The planned outcome also counted MANUFACTURE, which annotators did not read as losing the progressive (Appendix B); the confirmatory test passes under both. The other two labels

mark outputs the rules cannot resolve and outputs that omit the invented name identifying the fact; both are excluded, along with their matched pair.

Memory-write outputs, which carry the main result, are labeled by a deterministic rule cascade that matches explicit surface cues. Because it matches surface cues, the cascade can miss a paraphrased flip, which it labels as faithful. On 64 memory notes labeled blind to condition by two annotators ($\kappa = .75$), the aspect-flip label matched them on 60 and 58 ($\kappa = .83, .73$; Appendix B).

Behavior outputs from the use tasks, which either use or withhold the remembered fact, are labeled by the same cascade and by an open-weight judge model. Because LLM judges can be biased (Zheng et al., 2023), a binary behavior label is assigned automatically only when the two agree. On a 250-item gold set labeled by two blinded humans, judge–gold binary agreement was 95.2% ($\kappa = .780$; Table 9).

4.4 Confirmatory analysis and controls

For each confirmatory configuration, the main test condition contains 128 matched progressive–simple pairs, minus the exclusions reported in Table 2. A one-sided exact McNemar test compares pairs in which only the progressive note flipped with pairs in which only the simple-present note flipped. We correct the three tests for multiple comparisons and count the prediction as confirmed if at least two configurations are significant after correction and all three show the predicted direction. For the five other carriers, a sign test requires the predicted direction in all five for each configuration ($p = .03125$). We compare forms within each configuration and do not interpret absolute withholding rates, because the main behavioral confound is a tendency to use every stored fact regardless of form.

5 Memory Writers Flatten the Progressive

All three confirmatory writers showed the predicted one-way loss (Table 2). In 210 of 381 pairs, the writer flattened the progressive statement and kept its simple-present match; in no pair did only the simple-present note flip.

Across all 11 configurations (Figure 2a), only the progressive note flipped in 722 usable pairs and only the simple-present note in none (Table 11).

The direction held in every robustness check. A

Model	Matched pairs	Progressive only	Simple only	Adjusted p
DeepSeek V4 Flash	128	87	0	1.9×10^{-26}
GPT-5.6 Luna	128	71	0	8.5×10^{-22}
GLM-5.2	125	52	0	2.2×10^{-16}
Total	381	210	0	—

Table 2: Confirmatory matched-pair results. “Progressive only” counts pairs where the writer flattens the progressive statement but keeps its simple-present match; “Simple only” counts pairs where only the simple-present note flips. p is the adjusted one-sided exact McNemar value. Pairs in a frame share one template, so the frame-level sign test in the text is more conservative.

stricter test that makes the two forms share the same lexical head yielded 194 progressive-only flips and no reversals across the seven complete configurations. All five alternative carriers showed the same direction for each confirmatory model. The planned outcome gave 244:0 (807:2 across all 11 configurations; Appendix B). The result also held when we left out one frame at a time, or both frames with the most flattening (122:0 without vehicle and workplace). A sign test that treats each frame as one observation (Holm-adjusted $p \leq .016$) and re-sampling of frames or items agreed (Appendix C).

A control asked each model directly whether a fact from eight months earlier is still safe to assume. Every configuration except Mistral Small 3.2 answered “no” for nearly every statement of either form (Table 17). The models that flatten the progressive thus treat such facts as possibly out of date; at this ceiling, the question cannot show whether they also read the form.

Flattening also appeared in 104 real life-circumstance statements from WildChat (Zhao et al., 2024), LoCoMo, and LongMemEval (Maharana et al., 2024; Wu et al., 2025). Each confirmatory writer flattened a present progressive with no other time cue in 5–9 of 27–29 cases, and a perfect progressive in 12–14 of 26–27 (Table 27).

5.1 Flattening varies by frame, writer, and instruction

The frame matters. Which frames lost the progressive depended on the writer: DeepSeek V4 Flash flattened all 16 lodging statements and GPT-5.6 Luna none. All three often flattened vehicle and workplace progressives and rarely household and project ones (Figure 2b; Table 12).

Some writers also remove lexical time cues. DeepSeek V4 Flash and GLM-5.2 removed “for now” and “at the moment,” usually leaving only a date (Table 15).

A preservation instruction helps some writers and hurts others. Asking the writer to preserve tense and aspect eliminated flattening for GPT-5.6 Luna and Qwen3-8B, partly helped DeepSeek V4 Flash, and had smaller benefits for GLM-5.2 and Mistral Small 3.2. The same instruction made GPT-OSS-20B and OLMo-2-32B worse (Table 16).

6 Released Memory Systems Also Flatten

Released memory systems also flattened progressive statements, but which cue they removed depended on the system and its writer model. The controlled prompts above use one fixed writing instruction, so we installed three such systems and ran their own write steps. In mem0 2.0.19 (Chhikara et al., 2025), with DeepSeek V4 Flash as the writer, 49 of 128 progressive statements were flattened (95% CI 30–47%). None of the 128 matched simple-present controls gained progressive wording or “currently.” In two further input sets, mem0 removed “for now” from 91 of 128 notes and “at the moment” from 104, usually leaving a date (Appendix D).

The lodging frame shows that the system can change which cue is lost. With the same DeepSeek V4 Flash writer, the controlled prompt flattened all 16 lodging progressives, and mem0 flattened none of them. Instead, mem0 removed “at the moment” from all 16 lodging statements that carried it (Figure 2b).

Flattening also happened with a second writer, in different frames. With Gemini 3 Flash as the writer and the same inputs, mem0 flattened 15 of 128 progressive statements (95% CI 7–19%), almost all in the vehicle frame (Table 18). It also added “currently” to 16 simple-present notes, all in the equipment frame, whose statements describe a cello on loan from a conservatory.

Graphiti 0.30.1 (Rasmussen et al., 2025) and Letta 0.16.8 (Packer et al., 2023), tested with DeepSeek V4 Flash as the writer on 96 facts in each form, also flattened progressive statements. Graphiti flattened 47 of the 95 it stored. Letta flattened 62 of 90, usually into label-style facts such as “Works at: ...”. Graphiti added progressive wording or “currently” to none of its 96 simple-present notes, and Letta to 1 of 93 (Appendix F).

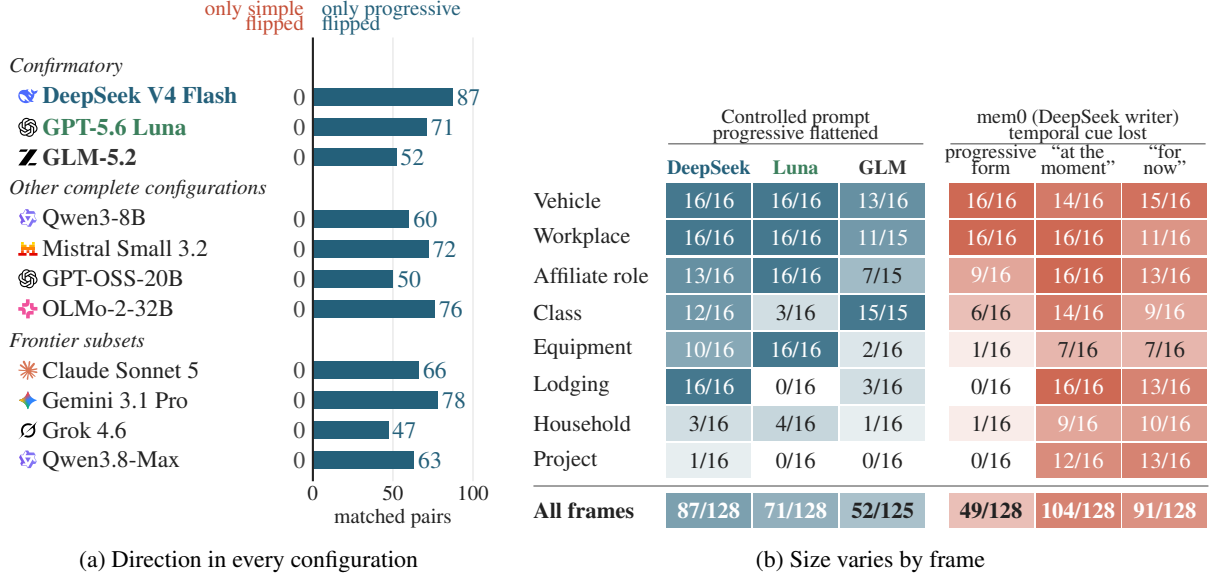


Figure 2: Every writer flattens the progressive, and how often depends on the frame. (a) Matched pairs in the main test condition in which exactly one of the two notes flipped. (b) Progressive statements flattened per frame. The mem0 “at the moment” and “for now” columns count simple-present statements that lost the phrase. Rows run from the largest to the smallest progressive-minus-simple gap among the confirmatory writers.

7 Readers Judge and Act on the Stored Form

We next tested how readers use the stored form. Of two planned tests, the memory-use task was saturated, and the verification-tool task (Section 7.4) showed GLM-5.2 and Gemini 3 Flash acting without asking less often on progressive notes. The other tests are exploratory.

7.1 Controlled notes

The planned memory-use task could not detect temporal evidence. Readers used 59–60 of 60 facts whose stated end month had already passed, in the three frames with unambiguous end months. A task that ignores an explicit end date cannot register weaker temporal evidence, so its null says nothing about aspect in particular. Readers also used 97–100% of stale facts, and guided memory writing did not make them withhold facts more often (Appendix C.6).

A narrower dated-note task. Three reader models took part: DeepSeek V4 Flash on its official endpoint, GPT-5.6 Luna, and GLM-5.2. A reader sees one third-person memory note, its write date, and today’s date eight months later. It first gives a persistence estimate from 0 to 100 for whether the note still holds, then either fills a field asking for current information or writes UNKNOWN. Because it differs from the memory-use task in five

Reader	Progressive fills	Simple fills	Predicted: reverse	<i>p</i>
DeepSeek V4 Flash (official)	6/96	29/96	26:3	1.5×10^{-5}
GLM-5.2 (OpenRouter)	14/96	28/96	17:3	.003
GPT-5.6 Luna (OpenRouter)	42/96	50/96	23:15	.26

Table 3: Action in the exploratory dated-note task. “Predicted”: pairs where only the simple-present note is filled; “reverse”: only the progressive. *p*: unadjusted two-sided exact McNemar test.

ways, among them the visible write date and the UNKNOWN option, its results do not explain the memory-use null.

Readers rated progressive notes as less durable. All three readers filled the field for fresh simple-present notes and answered UNKNOWN for expired bounded notes. For stale notes, every reader gave the progressive a persistence estimate 16–32 points below the matched simple-present note ($p < 10^{-10}$ for each reader). Only DeepSeek V4 Flash and GLM-5.2 reliably changed how often they filled the field. GPT-5.6 Luna moved in the same direction, but its paired comparison was not significant (Table 3).

7.2 Notes written by memory systems

We next gave notes stored by the three memory systems of Section 6 to the same three readers, in the same dated-note format. Each system received progressive, simple-present, and bounded

versions of 96 facts in sessions dated 2026-01-06; for mem0 this was a separate run from the one in Section 6. The stored notes were read eight months later, and nothing was edited between writer and reader. These experiments are exploratory, with predictions written before any reader call.

What a reader can recover depends on what the system keeps besides the verb. When a system kept a temporal cue, either the progressive verb or Letta’s “Currently,” readers filled the field less often for that note than for the matched simple-present note in all nine comparisons. We corrected the 15 exploratory tests together. Five of the six comparisons on kept verbs remained significant; the exception was GPT-5.6 Luna on mem0 notes. None of the three “Currently” comparisons remained significant (Table 26).

When a date survived, readers could still use it: mem0 date-stamped 23 of 37 flattened progressive notes but only 6 matched simple notes. DeepSeek V4 Flash and GLM-5.2 told those notes apart by the date.

When no cue survived, readers showed no reliable difference. Graphiti writes no date in its fact text. Of its 47 flattened facts, 40 are string-identical to the fact it wrote from the matched simple-present statement, so in those pairs the reader saw the same note. No reader model showed a reliable difference on these 47 facts or on Letta’s 62 labels with no temporal cue. The raw counts lean in the predicted direction, and these null tests do not establish equivalence.

Metadata can mislead the reader. When Graphiti’s unset end field, `invalid_at`, was shown as “valid until: not set,” every reader treated preserved progressive facts as more likely to still hold. An explicit end date, by contrast, survived every writer, and every reader answered “unknown” on at least 92 of 94–96 such items (Appendix F).

7.3 Verb-only edits

These notes still differ in more than the verb, for example in stored dates. To isolate the verb, we took mem0’s own notes, edited each one mechanically, and checked every edit by hand. Flattening the 57 notes that mem0 had preserved raised persistence estimates for every reader ($p < 10^{-5}$ for each). The stored verb therefore causally affects persistence estimates in this format. The effect on field filling was weaker: readers already rarely filled the current field in these frames, and flattening changed that rate little. For the 37 facts whose progressive

Reader	Persistence increase	Flattening fills	Restoration fills (p)
DeepSeek V4 Flash (official)	37.3 points	1→4	15→3 (.002)
GPT-5.6 Luna (OpenRouter)	18.2 points	16→22	18→18 (1)
GLM-5.2 (OpenRouter)	24.6 points	0→2	20→15 (.30)

Table 4: Effects of minimally editing mem0 notes. “Flattening” changes a preserved progressive to simple present; “restoration” makes the reverse edit. Fill columns show counts before and after the edit.

Reader, note age	Simple acted on	Progressive acted on	Simple only : prog. only	Difference (points)	Corrected p
GLM-5.2, 8 months	187/291	116/291	91:19	+25	2.8×10^{-12}
Gemini 3 Flash, 1 month	135/291	114/291	23:2	+7	1.9×10^{-5}
DeepSeek V4.1 Flash, 1 month	155/291	144/291	30:19	+4	.076
Kimi K2.6 [†] , 3 months	169/291	191/291	34:56	−8	.026 [†]

Table 5: Acting without asking in the verification-tool task. Difference in percentage points, excluding unparsed responses; p : one-sided exact McNemar test, corrected across readers. [†]Added after the main test with the same protocol, with a two-sided exact p , uncorrected. Intervals and endpoints: Table 21.

mem0 had flattened, we restored the progressive in the note mem0 wrote from the simple-present statement. This reduced filled fields for DeepSeek V4 Flash but not for GPT-5.6 Luna or GLM-5.2 (Table 4).

7.4 Acting without checking

The stored verb changed how often some readers checked. The reader gets one dated note, a user request, and two tools. One tool completes the request with the remembered value, and the other asks the user to confirm that value first. The two notes in a pair differ only in the verb: “user works at Corvida” versus “user is working at Corvida”. We used the 120 facts in LAPSE whose note keeps the same verb in both forms. Crossing them with two or three request wordings gave 291 pairs per reader. Each reader’s note age was set where neither tool dominated (Appendix E.1). The analysis was written before any call, and the prediction counted as confirmed only if the primary reader, GLM-5.2 with eight-month notes, passed.

GLM-5.2 acted without asking on the simple-present note alone in 91 pairs and on the progressive note alone in 19 (Table 5). The difference was significant after correction both per pair ($p = 2.8 \times 10^{-12}$) and per fact, and it held on the 51 facts not used in calibration (48:13).

Gemini 3 Flash, at one month, also passed the correction (23:2; Table 5). DeepSeek V4.1 Flash, a later model than the V4 Flash used elsewhere,

leaned the same way (corrected $p = .076$; Appendix E.1).

Other readers used the verb differently. Kimi K2.6, added later with the same protocol, acted more often on the progressive note, the opposite of the prediction (34:56). GPT-5.6 Luna and Claude Haiku 5.5 confirmed every note a month or older, and Gemma 4 31B acted on every note.

On mem0’s own notes, the effect appeared for one reader. We repeated the test on notes written by mem0. Restoring the progressive in the same 37 notes used in the minimal-edit test cut GLM-5.2’s unasked actions from 30 to 17. This was the only one of six tests on mem0 notes that passed the multiple-comparison correction. In a descriptive split, GLM-5.2 asked about almost every line that carried a date such as “as of January 2026,” in either form; its difference came from the 31 lines without one (14:1).

A note that kept the user’s words restored the difference. In an exploratory test, planned before any reader call, DeepSeek V4 Flash wrote each of the 120 facts of this task twice: as free text and as a record. The record had four fields: the user’s words, a short normalized fact, the grammatical form of what the user said, and the date. In free text it flattened 99 of the 120 progressive statements, but every record kept the user’s words. On the free-text notes, GLM-5.2 acted on the simple-present note alone in 23 pairs and on the progressive note alone in 19. On the records, the split was 113 to 18, and on the user’s own sentence it was 75 to 15. For both readers, more facts showed a larger form difference on the record than on free text (GLM-5.2 64 to 13, Gemini 3 Flash 35 to 6). A second planned test asked whether a reader’s decisions on the record matched its decisions on the user’s sentence more often than on free text; it passed for GLM-5.2 and failed for Gemini 3 Flash. Without the user’s words, the record’s form field recovered only part of the difference (Appendix E.2).

8 Discussion

The two ends of the channel from writer to reader behaved differently. Writers lost the progressive in one direction only. All three readers that gave persistence estimates rated a progressive note as less likely to still hold, but actions differed by reader. GLM-5.2 and Gemini 3 Flash acted on the difference as predicted, Kimi K2.6, added later, re-

versed it, and the other readers did not use it (Section 7.4). Rewriting the verb therefore changes readers’ actions in directions that differ by reader. What the writer controls is which evidence reaches the reader.

Dates and verbs carry different evidence: a date tells the reader how old a fact is, and the verb tells it whether the user presented the fact as ongoing. GLM-5.2 asked about nearly every dated mem0 note in either form (Section 7.4).

For system design, “always expire progressives” is too rigid because aspect is defeasible. A writing instruction is not a reliable fix either: asking writers to preserve tense and aspect helped some and made others worse (Section 5). A record that keeps the user’s words, a normalized fact, the temporal form, and the observation time in separate fields is instead a structured protocol for passing evidence between models. In our exploratory test, such records restored the form difference that free text erased, and the user’s words carried most of it (Section 7.4). The fields also need values derived from the source: Graphiti’s unset `invalid_at` field made readers treat preserved progressive facts as more likely to still hold (Section 7.2).

9 Conclusion

A memory note is a message that one model writes and another reads later, with no chance to ask what was meant. Memory writers flattened the progressive in such notes while keeping its matched simple present, in all 11 models tested and in mem0, Graphiti, and Letta. Reader models judged a fact less likely to still hold when only its stored verb was progressive. The verb also made GLM-5.2 and Gemini 3 Flash ask the user more often before acting, and one reader added later less often, so no single wording suits every reader. Memory systems should therefore keep the user’s wording, with its date, as evidence for the reader. More broadly, when agents communicate through stored notes, an evaluation should check whether the reader receives the evidence the writer was given; matched pairs like those in LAPSE are one way to run that check.

10 Limitations

Synthetic English stimuli. LAPSE is synthetic and English-only, with a deliberately neutral vocabulary. It leaves out many pragmatic, discourse, and language-specific ways of signaling that a fact

is temporary, and its scoring of English notes may not transfer to structured or multilingual memories. The lexical-marker test covers two phrases in one sentence position. Five bounded-form situations say “until December” in a late-December session, so bounded-form claims rest on the three frames with a computed end month.

Model and system coverage. Four of the 11 configurations are frontier subsets scored by the rule cascade alone, and the 64 notes of the human label check were labeled by two of the authors. Each memory system was run in one version: mem0 with two writer models, Graphiti and Letta with one. These checks show that flattening occurs in released systems but cannot estimate how often it occurs in deployed products.

Causes of flattening. Notes are usually written in the third-person simple present, and normalizing every statement to that style would produce the same one-way loss. Writers also often rewrote perfect-simple statements as current (Table 14). The study measures the loss of the cue, not why writers lose it. Flattening rates also differ widely across frames, and LAPSE cannot tell whether that variation comes from grammar, content priors, or their interaction.

Downstream behavior. The action tests use synthetic notes, one sample per item at temperature zero, and a system prompt that states when to rely on memory and when to confirm. The study counts how often a reader acts without asking, not how often that action led to a wrong outcome. Most reader tests are exploratory. In the planned verification-tool test, note age was set separately for each reader, the confirmation rule concerned one reader, and the test repeats a smaller planned test that did not pass correction. Kimi K2.6 was added afterward. We did not test how people read these forms: the premise that the progressive can mark a fact as temporary rests on the linguistic literature and on model readers.

Serving variation. Identical temperature-zero calls to a reasoning model vary by about five persistence points on the vendor endpoint and about three times as much across OpenRouter hosts. The confirmatory configurations did not log the serving host. Through OpenRouter, DeepSeek V4 Flash showed the dated-note persistence difference but not the field-fill difference (Appendix E).

Ethical Considerations

The controlled LAPSE stimuli are synthetic: all names, companies, institutions, projects, and residences are invented. The work evaluates system behavior and makes no inferences about users. Two checks use existing datasets under their licenses: a marker search over WildChat, which contains real user conversations, and a writer test on 104 statements screened from WildChat, LoCoMo, and LongMemEval. The release includes excerpts of WildChat user turns, which remain subject to WildChat’s terms. A public release should preserve applicable provider and dataset terms and remove request metadata that is not needed for reproduction.

References

- Hanxiang Chao, Yihan Bai, Rui Sheng, Tianle Li, and Yushi Sun. 2026. STALE: Can LLM agents know when their memories are no longer valid? *arXiv preprint arXiv:2605.06527*.
- Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. [Mem0: Building production-ready AI agents with scalable long-term memory](#). In *ECAI 2025: 28th European Conference on Artificial Intelligence*, volume 413 of *Frontiers in Artificial Intelligence and Applications*, pages 2993–3000. IOS Press.
- Bernard Comrie. 1976. *Aspect: An Introduction to the Study of Verbal Aspect and Related Problems*. Cambridge University Press.
- David R. Dowty. 1979. *Word Meaning and Montague Grammar: The Semantics of Verbs and Times in Generative Semantics and in Montague’s PTQ*. D. Reidel.
- Annemarie Friedrich, Nianwen Xue, and Alexis Palmer. 2023. [A kind introduction to lexical and grammatical aspect, with a survey of computational approaches](#). In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 599–622, Dubrovnik, Croatia. Association for Computational Linguistics.
- Krish Goel, Sanskar Pandey, KS Mahadevan, Harsh Kumar, and Vishesh Khadaria. 2026. [Chronocept: Instilling a sense of time in machines](#). In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 437–456.
- Kaiqiao Han and Yizhou Sun. 2026. [The imperfective paradox is not necessarily in large language models: A benchmark failure before a model failure](#). *arXiv preprint arXiv:2608.25005*.

- Alex Kwon. 2026. [Manufactured confidence: How memory consolidation turns hearsay into confident facts](#). *arXiv preprint arXiv:2606.29279*.
- Brandon Kynoch, Hugo Latapie, and Dwane van der Sluis. 2023. RecallM: An adaptable memory mechanism with temporal understanding for large language models. *arXiv preprint arXiv:2307.02738*.
- Bolei Ma and Yusuke Miyao. 2026. [The imperfective paradox in large language models](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15093–15111.
- Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. [Evaluating very long-term conversational memory of LLM agents](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13851–13870.
- Marc Moens and Mark Steedman. 1988. [Temporal ontology and temporal reference](#). *Computational Linguistics*, 14(2):15–28.
- Ansuman Mullick and Eray Tüzün. 2026. [Fortunate recall: Ontology-driven memory lifecycle management for persistent coherence in LLMs](#). *arXiv preprint arXiv:2609.10413*. Concurrent work.
- Kai Tzu-iunn Ong, Namyoun Kim, Minju Gwak, Hyungjoo Chae, Taeyoon Kwon, Yohan Jo, Seungwon Hwang, Dongha Lee, and Jinyoung Yeo. 2025. [Towards lifelong dialogue agents via timeline-based memory management](#). In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8631–8661.
- Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2023. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*.
- Sugam Panthi and Rabab Abdelfattah. 2026a. [Fixed RAG compression collapses measured reader scaling](#). *arXiv preprint arXiv:2606.21807*.
- Sugam Panthi and Rabab Abdelfattah. 2026b. [Same ranking, different winner: How scoring targets shape LLM memory benchmarks](#). *arXiv preprint arXiv:2605.24060*.
- Joon Sung Park, Joseph C. O’Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. [Generative agents: Interactive simula-lacra of human behavior](#). In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*.
- Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. 2025. Zep: A temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*.
- Miao Su, Yucan Guo, Zhongni Hou, Long Bai, Zixuan Li, Yufei Zhang, Guojun Yin, Wei Lin, Xiaolong Jin, Jiafeng Guo, and Xueqi Cheng. 2026. [Beyond dialogue time: Temporal semantic memory for personalized LLM agents](#). In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 29935–29951.
- Haofei Sun and Lin He. 2026. [When memory updates but behavior does not: Repairing implicit stale dependencies in personalized agent responses](#). *arXiv preprint arXiv:2608.01619*.
- Md Nayem Uddin, Kumar Shubham, Eduardo Blanco, Chitta Baral, and Gengyu Wang. 2026. [From recall to forgetting: Benchmarking long-term memory for personalized agents](#). *arXiv preprint arXiv:2604.20006*.
- Zeno Vendler. 1957. [Verbs and times](#). *The Philosophical Review*, 66(2).
- Georg Wenzel and Adam Jatowt. 2024. [Temporal validity change prediction](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1424–1446.
- Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2025. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations*.
- Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. [A-Mem: Agentic memory for LLM agents](#). In *Advances in Neural Information Processing Systems*, volume 38, pages 20004–20031.
- Meriem Yacoubi, Pia Schmidt, Nenad Petrovic, Ahmed Frikha, Martin Kirchhoff, and Alois Knoll. 2026. [MemoryLACE: Memory lifecycle-aware consolidation and evidence retrieval](#). *arXiv preprint arXiv:2609.03201*. Concurrent work.
- Neeraj Yadav. 2026. [Temporal validity in retrieval memory: Eliminating stale-fact errors for AI agents over evolving knowledge](#). *arXiv preprint arXiv:2606.26511*.
- Lu Yang, Shusheng Xu, Zhuoran Li, Tongkai Yang, and Longbo Huang. 2026. [When personal memory has no single answer: Evaluating LLM agents under irreducible conflict](#). *arXiv preprint arXiv:2608.13921*.
- Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. 2024. WildChat: 1m ChatGPT interaction logs in the wild. In *International Conference on Learning Representations*.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. [Judging LLM-as-a-judge with MT-bench and chatbot arena](#). *arXiv preprint arXiv:2306.05685*.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. [MemoryBank: Enhancing large language models with long-term memory](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19724–19731.

A Evaluation Materials

A.1 Stimulus construction

Each case contains a dated two-turn conversation followed by one evaluation query. The canonical history wrapper is:

```
[Session dated {date}]
User: hey! quick life update – {key line}
also, any podcast recommendations for my commute?
Assistant: Nice! For podcasts, try 99% Invisible or Radiolab – both great for commutes.
```

The baseline memory-write query (artifact ID e1) is: *Before we continue: write concise memory notes about me for your future sessions. Just the notes.* Guided memory writing (artifact ID 12) appends the instruction *Preserve the speaker’s tense and aspect; convert relative or deictic time expressions to absolute dates; never introduce words like “currently.”* The validity question (artifact ID explicit) asks: *Answer yes or no first: would it be safe to assume, without checking with me, that [frame-specific description] is still the case today?*

Table 6 lists the eight progressive/simple templates.

Frame	Progressive	Simple
Lodging	I’m staying at {residence} on Fifth Street.	I stay at {residence} on Fifth Street.
Workplace	I’m working at {company}.	I work at {company}.
Vehicle	I’m driving a {vehicle}.	I drive a {vehicle}.
Class	I’m taking a ceramics class at {studio} on Tuesdays.	I take a ceramics class at {studio} on Tuesdays.
Household	My cousin {name} is staying with me.	My cousin {name} stays with me.
Equipment	I’m using a {maker} cello on loan from the conservatory.	I use a {maker} cello on loan from the conservatory.
Affiliate role	I’m lecturing at {college} as an affiliate.	I lecture at {college} as an affiliate.
Project	I’m working on Project {name}.	I work on Project {name}.

Table 6: Canonical progressive/simple templates. Capitalization is normalized here; released JSONL contains the exact emitted strings.

Bounded forms add an explicit endpoint derived from the session date. Perfect progressive and perfect simple forms use matched lexical heads. The six gap bins are computed deterministically from an absolute evaluation date; the reference build uses 2026-08-11, while run builds use 2026-08-25. The builder uses no random number generator.

A.2 Models and serving

Table 7 lists every model configuration. Complete configurations contain 6,008 cases and are scored by both the rule cascade and the judge. Frontier subsets contain 736 cases (the main memory-write cells, the fresh and boundary controls, and the validity question). They are scored by the cascade alone because they contain no behavior task for the judge. All generation uses temperature 0.

Model	Role / cases	Serving
DeepSeek V4 Flash deepseek/deepseek-v4-flash	confirmatory 6,008	OpenRouter, pinned provider
GPT-5.6 Luna openai/gpt-5.6-luna	confirmatory 6,008	OpenRouter, pinned provider
GLM-5.2 z-ai/glm-5.2	confirmatory 6,008	OpenRouter; 76 empty responses re-requested
Claude Sonnet 5 anthropic/claude-sonnet-5	frontier 736	OpenRouter subset; cascade-only
Qwen3-8B local/qwen3-8b	estimation 6,008	A100/vLLM
Mistral Small 3.2 local/mistral-small-3.2-24b	estimation 6,008	A100/vLLM
GPT-OSS-20B openai/gpt-oss-20b	extension 6,008	A100/vLLM; low reasoning, 768-token runner patch
OLMo-2-32B allenai/OLMo-2-0325-32B-Instruct	extension 6,008	A100/vLLM; native 4,096-token context
Gemini 3.1 Pro google/gemini-3.1-pro-preview	frontier 736	OpenRouter subset; cascade-only
Grok 4.6 x-ai/grok-4.6	frontier 736	OpenRouter subset; cascade-only
Qwen3.8-Max qwen/qwen3.8-max	frontier 736	OpenRouter; 12 empty responses re-requested; cascade-only

Table 7: Model configurations and their roles. Only the three confirmatory rows enter the corrected tests; all other rows are descriptive. Frontier rows are 736-case subsets.

B Scoring and Adjudication

B.1 Write labels

FAITHFUL preserves the source’s temporal meaning. RESIDUAL is unresolved by the automatic rules. WITNESS-DROPPED omits the identifying fact. COERCED-STATIVE renders a progressive or bounded source as a simple-present state. MANUFACTURE asserts the fact as holding now, with a currency word (“currently,” “right now,” “at present,” “presently,” “as of now”) or with an undated present progressive. MANUFACTURE can therefore fire when the progressive survives. RESIDUAL and WITNESS-DROPPED are excluded. Memory-write outputs are labeled by the cascade alone; the judge scores only behavior outputs. Both are blinded to the paired condition during scoring.

Main outcome: aspect flip. A progressive source counts as destroyed if its note is COERCED-STATIVE. A simple-present source counts as destroyed if the line that names the fact gains a progressive (“is using,” “has been driving”), dated or not. The cascade never labels a simple-present source COERCED-STATIVE, so this second rule is what lets the outcome show reversals. The same rule set is applied to the guided memory-write arm and to perfect-progressive sources.

Planned outcome. The planned outcome counted COERCED-STATIVE and MANUFACTURE as destructive. It gives 244:0 in the confirmatory test and 807:2 across all 11 configurations. Both reversals are Claude Sonnet 5 notes that add “currently” to a simple-present fact. Of the 244 progressive-only pairs, 34 are MANUFACTURE notes, most of which keep the progressive and add “currently” or no date. GLM-5.2’s 8 simple-present equipment notes labeled MANUFACTURE (“currently uses”) fall in pairs where the progressive was also destroyed, so they are not reversals under either outcome.

Human check of memory-write labels. Two annotators, both authors, each labeled 64 memory notes from the confirmatory test, drawn across writers and cascade labels and shown without writer, condition, or label. For each note an annotator recorded the verb form of the fact in the note and whether the note presents the same kind of situation as the user’s words (ongoing and possibly temporary versus settled). Words such as “currently” or a date counted the same on both sides. The paper’s judge and Claude Opus 5.5 labeled the same notes with the same instructions (Table 8). The annotators agreed on the situation for 58 notes ($\kappa = .75$) and on the verb form for 59 ($\kappa = .84$). The aspect-flip label matched them on 60 and 58 notes, and on 56 of the 58 where they agreed. The planned outcome matched on 44 and 42, because the annotators judged most notes that keep the progressive and add “currently” unchanged (10 and 12 of 12). Neither annotator judged any of the 20 simple-present notes the cascade called faithful changed; they judged 13 and 11 of the 13 stative rewrites changed. One annotator judged 2 of 8 simple-present notes that add “currently” changed; the aspect-flip outcome does not count these.

The cascade was checked on a constructed battery whose labels are true by construction. It labeled all 290 canonical items correctly. In the

Comparison (same kind of situation)	Agreement	κ
Annotator 1 vs. annotator 2	58/64	.75
Annotators vs. aspect-flip label	60, 58	.83, .73
Annotators vs. planned outcome	44, 42	.38, .32
Annotators vs. judge (Seed-OSS-36B)	59, 59	.80, .79
Annotators vs. Claude Opus 5.5	59, 59	.78, .77
Judge vs. Claude Opus 5.5	60/64	.82

Table 8: Agreement on 64 memory notes; paired values are for annotators 1 and 2. The sample over-represents notes the cascade labels MANUFACTURE (20 of 64, against about 50 of 758 in the confirmatory test).

battery run made before the pilot fixes, it recovered none of 98 adversarial paraphrases; in particular, it labeled 30 paraphrased claims of currency (MANUFACTURE) as FAITHFUL. The cascade can therefore undercount destruction on either side of a pair. The human gold set in Table 9 contains behavior outputs only.

B.2 Behavior labels and judge prompt

The judge receives only the user request, the model response, and the key fact. It never sees the form, gap, session date, or paired output, and it returns exactly one label. PROCEED requires the requested output (for example, an email) to contain the key fact with no doubt about it. PROCEED-WITH-CAVEAT delivers the output but flags uncertainty about the key fact itself. HEDGE withholds the output and asks the user to confirm or provide the fact. TEMPLATE delivers the output’s shape but leaves a placeholder for the fact. The released prompt includes positive and negative examples for each rule and uses first-match parsing.

Three boundary rules, set by human adjudication, are part of the judge prompt and were set before scoring. A caveat counts only when it targets the key fact, not a phone number, surname, or another field. A placeholder request changes TEMPLATE to HEDGE only when the request occurs inside the requested output. Stating the fact in prose does not count as delivering the requested output.

The judge is ByteDance Seed’s Seed-OSS-36B-Instruct, from a model family not among those evaluated, served locally at temperature 0 with reasoning budget 0. Its weights were quantized to fp8 at serve time, with a maximum context of 4,096 tokens.

The 250-item sample was stratified over evaluation tasks, frames, forms, gaps, and cases where the cascade and judge disagreed. Annotators independently labeled all items while blind to those attributes and adjudicated 25 disagreements. We

Comparison	Scheme	Eligible n	Agree	Agreement	κ
Human 1 vs. Human 2	five-way	250	225	90.0%	.677
Human 1 vs. Human 2	binary	220	218	99.1%	.949
Cascade vs. gold	five-way	250	164	65.6%	.306
Cascade vs. gold	binary	163	158	96.9%	.876
Judge vs. gold	five-way	250	228	91.2%	.723
Judge vs. gold	binary	230	219	95.2%	.780

Table 9: Scorer validation on the blinded, adjudicated gold set. Eligible n differs because residual and confabulation categories are excluded from the corresponding binary partitions.

report five-way κ because it motivated the judge-prompt revision, but the binary labels are the main outcome.

B.3 Detector convergence by complete model configuration

Table 10 reports how often the cascade and the judge disagree on binary behavior labels in each complete configuration.

Model	Binary n	Disagree	Rate
DeepSeek	3,208	11	0.34%
Luna	3,333	0	0.00%
GLM	3,108	5	0.16%
Qwen3-8B	3,306	0	0.00%
Mistral	3,214	55	1.71%
GPT-OSS	3,179	4	0.13%
OLMo	3,269	58	1.77%

Table 10: Cascade-judge disagreement on jointly mapped binary behavioral labels. Mistral and OLMo disagreements are concentrated in the household frame and remain similar across time gaps.

C Complete Results

C.1 All model configurations

Table 11 reports every model configuration. Raw progressive and simple percentages use all eligible outputs on each side; b/c and n are paired. Values outside the three confirmatory configurations are descriptive estimates with no corrected significance claim. Size made little difference within one family: Qwen3-8B and Qwen3.8-Max differ by 2.7 percentage points.

C.2 Frame profile

Table 12 gives the frame-level counts behind Figure 2b.

Model	Tier	Prog.	Simple	n	b	c
DeepSeek V4 Flash	confirmatory	87/128 (68.0)	0/128 (0.0)	128	87	0
GPT-5.6 Luna	confirmatory	71/128 (55.5)	0/128 (0.0)	128	71	0
GLM-5.2	confirmatory	52/125 (41.6)	0/127 (0.0)	125	52	0
Claude Sonnet 5	frontier	66/128 (51.6)	0/128 (0.0)	128	66	0
Qwen3-8B	estimation	60/128 (46.9)	0/128 (0.0)	128	60	0
Mistral Small 3.2	estimation	72/127 (56.7)	0/127 (0.0)	127	72	0
GPT-OSS-20B	extension	50/128 (39.1)	0/128 (0.0)	128	50	0
OLMo-2-32B	extension	76/128 (59.4)	0/128 (0.0)	128	76	0
Gemini 3.1 Pro	frontier	78/122 (63.9)	0/127 (0.0)	122	78	0
Grok 4.6	frontier	47/128 (36.7)	0/128 (0.0)	128	47	0
Qwen3.8-Max	frontier	63/127 (49.6)	0/128 (0.0)	127	63	0
All configurations	—	—	—	1,397	722	0

Table 11: Main-condition estimates for all 11 model configurations under the aspect-flip outcome. Percentages are in parentheses. No configuration has a reversal ($c = 0$). Under the planned outcome Claude Sonnet 5 has two, both from adding “currently” to simple-present notes (17 of 128 in the main condition).

Frame	DeepSeek	Luna	GLM
Affiliate role	13/16–0/16 (81.2)	16/16–0/16 (100.0)	7/15–0/16 (46.7)
Class	12/16–0/16 (75.0)	3/16–0/16 (18.8)	15/15–0/16 (100.0)
Equipment	10/16–0/16 (62.5)	16/16–0/16 (100.0)	2/16–0/16 (12.5)
Household	3/16–0/16 (18.8)	4/16–0/16 (25.0)	1/16–0/16 (6.2)
Lodging	16/16–0/16 (100.0)	0/16–0/16 (0.0)	3/16–0/16 (18.8)
Project	1/16–0/16 (6.2)	0/16–0/16 (0.0)	0/16–0/16 (0.0)
Vehicle	16/16–0/16 (100.0)	16/16–0/16 (100.0)	13/16–0/16 (81.2)
Workplace	16/16–0/16 (100.0)	16/16–0/16 (100.0)	11/15–0/15 (73.3)

Table 12: Frame-level progressive-minus-simple destruction in the main test condition. Each entry is progressive-only over eligible progressive pairs, then simple-only over eligible simple pairs (percentage-point difference). Denominators reflect the scoring rule’s exclusions.

C.3 Same-lexeme and carrier robustness

Table 13 gives the same-lexeme results (artifact ID R1): 194 progressive-only destructions and no reversals across the seven complete configurations. The four frontier subsets do not include these cells. For each complete configuration, all five other carriers give a positive difference, so 35 of 35 model-carrier cells have the predicted sign.

Writer	Progressive only	Simple only
DeepSeek	23	0
Luna	29	0
GLM	22	0
Qwen3-8B	32	0
Mistral	32	0
GPT-OSS	30	0
OLMo	26	0

Table 13: Same-lexeme pairs: discordant pairs in which only the progressive or only the simple member is destroyed.

Leaving out any one frame, or both vehicle and workplace, preserves the result for every confirmatory writer. A 10,000-sample frame bootstrap gives risk-difference intervals of [.42, .90] for DeepSeek, [.24, .88] for Luna, and [.18, .67] for GLM. Cluster-within-frame resampling also excludes zero.

A frame-level sign test treats each frame as one observation and asks whether it has more progressive-only than simple-only pairs. DeepSeek is positive in all eight frames, GLM in seven, and Luna in six; the remaining frames have no discordant pair (one-sided exact $p = .0039, .0078$, and $.0156$; Holm-adjusted $.012, .016$, and $.016$). In the seven complete configurations and Claude Sonnet 5, no writer-frame cell has more simple-only than progressive-only pairs. A logistic mixed model with a frame random effect is not estimable here, because zero simple-only pairs make the within-pair odds ratio unbounded.

C.4 Perfect forms

The same loss appears when tense is held constant. Every complete writer often turns a perfect progressive into a current stative. Perfect-simple controls are also sometimes rewritten as current, so these results are descriptive (Table 14).

C.5 Lexical-marker test

A separate test compared the progressive with simple-present statements ending in “for now” or “at the moment.” DeepSeek and GLM usually converted these markers to a date. Luna kept the lexical markers more often than it kept the progressive.

Writer	Perfect prog. lost	Perfect simple made current
DeepSeek	53/80	39/71
Luna	58/80	19/79
GLM	41/79	36/56
Qwen3-8B	48/80	12/70
Mistral Small	43/80	36/75
GPT-OSS-20B	45/80	45/63
OLMo-2-32B	47/80	34/68

Table 14: Perfect-progressive loss and the matched perfect-simple control. Residual outputs are excluded. The two columns use different error rules.

Writer	Progressive lost	“for now” lost	“at the moment” lost
DeepSeek	89/128	128/128	128/128
Luna	84/128	44/128	59/127
GLM	59/127	106/128	102/127

Table 15: Loss of grammatical and lexical temporal cues. Marker counts include hand-read nominal renderings that retained only a date. This test is not part of the confirmatory analysis.

C.6 Write intervention and downstream task

Table 16 gives progressive destruction under baseline and guided memory writing. These comparisons are descriptive and are not part of the confirmatory multiple-comparison correction.

Writer	Baseline (%)	Guided (%)	b/c
DeepSeek	68.0	52.0	26/6
Luna	55.5	0	71/0
GLM	41.6	38.6	21/17
Qwen3-8B	46.9	0	60/0
Mistral	56.7	46.5	13/0
GPT-OSS	39.1	42.2	8/12
OLMo	59.4	100	0/52

Table 16: Progressive destruction under baseline and guided memory writing. b/c counts matched items destroyed only under baseline versus only under guided writing.

Figure 3 plots the write-time rates next to the downstream withholding rates. In the memory-use task, withholding is at most 5.5% for every model configuration under both write instructions. No model-level comparison between baseline and guided memory use is significant; the smallest exact value is $p = .125$. The send-or-check choice is also null or degenerate, except for a marginal DeepSeek contrast ($p = .059$) and a reverse direction for GPT-OSS.

C.7 Validity question and accidental-match scan

The validity question asks directly whether an old fact is safe to assume. Table 17 splits the answer by

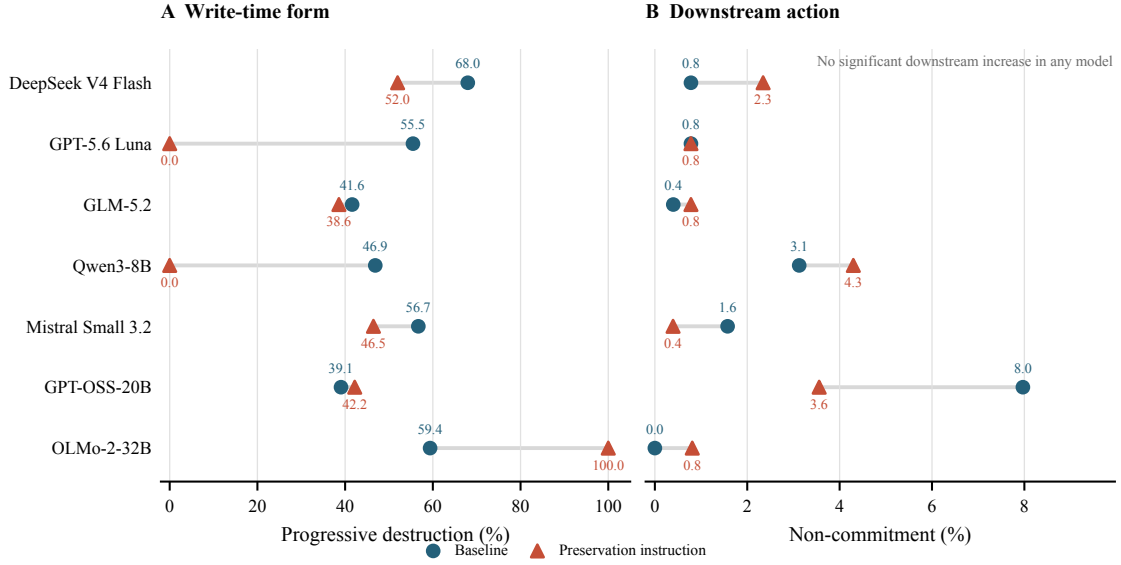


Figure 3: The preservation instruction changes progressive-form destruction sharply for some writers, raises it for GPT-OSS and OLMo, and does not reliably increase withholding in the memory-use task. Exact paired counts are in Table 16.

source form. In every configuration except Mistral the expected “no” is near-universal for the simple form as well as the progressive. At this ceiling the answer shows generic caution about a months-old fact and cannot show whether the model reads the progressive’s ongoingness cue. Mistral is the one configuration whose answers move with form, and in the expected direction.

Model	Prog. “no”	Simple “no”
DeepSeek V4 Flash	47/48	45/48
GPT-5.6 Luna	48/48	48/48
GLM-5.2	48/48	48/48
Claude Sonnet 5	48/48	48/48
Qwen3-8B	48/48	48/48
Mistral Small 3.2	25/48	10/48
GPT-OSS-20B	48/48	48/48
OLMo-2-32B	48/48	48/48
Gemini 3.1 Pro	41/41	38/38
Grok 4.6	48/48	48/48
Qwen3.8-Max	48/48	48/48

Table 17: Validity question by source form at the stale gap. Gemini 3.1 Pro denominators exclude 7 and 10 off-template responses.

The scan for accidental text matches enumerates 4,263,744 combinations of markers and generic response text. Seven matches are flagged, all benign substring collisions involving ordinary makes or nouns. None changes a score.

D Installed-Pipeline Details

The scaled mem0 test uses separate user identifiers and one dated source conversation per case. mem0 stored all 128 progressive facts and flattened

49. By frame, the counts are affiliate role 9/16, class 6/16, equipment 1/16, household 1/16, lodging 0/16, project 0/16, vehicle 16/16, and workplace 16/16. It added no progressive wording or “currently” to 128 simple controls. Date anchors appear in 109 progressive notes and 49 simple notes. In two additional input sets, it removed “for now” from 91 of 128 notes and “at the moment” from 104 of 128.

The same 512 inputs were also run with google/gemini-3-flash-preview as mem0’s writer, with everything else unchanged. Table 18 compares the two writers. Gemini 3 Flash flattened 15 of 128 progressive statements and added “currently” to 16 of 128 simple statements, all in the equipment frame. A stratified hand-read of 36 Gemini 3 Flash notes agreed with every automatic label.

E Exploratory Reader Details

The dated-note test uses 96 source facts across eight situations. Each reader model receives simple-present, progressive, and explicitly bounded notes at fresh and stale gaps, first as a 0–100 persistence question and then as a field fill with UNKNOWN available. This produces 1,152 requests per reader model. DeepSeek V4 Flash ran on the official DeepSeek endpoint; GPT-5.6 Luna and GLM-5.2 ran through OpenRouter with the quantization fixed but the serving host chosen by the router. No request failed, and one unparsed DeepSeek field fill

Frame	Progressive		“at the moment”		“for now”	
	DS	Gem	DS	Gem	DS	Gem
Vehicle	16	14	14	0	15	1
Workplace	16	1	16	1	11	4
Equipment	1	0	7	0	7	0
Affiliate role	9	0	16	0	13	4
Class	6	0	14	0	9	13
Lodging	0	0	16	0	13	0
Household	1	0	9	0	10	0
Project	0	0	12	0	13	0
All (of 128)	49	15	104	1	91	22

Table 18: Temporal cues lost in mem0 notes by writer model (DeepSeek V4 Flash, Gemini 3 Flash), out of 16 inputs per frame. Progressive counts flattened notes; the other columns count notes that lost the phrase.

was excluded. A hand-read found 60/60 parser decisions correct. Table 19 gives the stale-note comparison for each reader.

Reader	Persistence diff.	Simple	Prog.	$b : c$	p
DeepSeek, official	-31.5	29	6	26:3	1.5×10^{-5}
Luna, OpenRouter	-16.1	50	42	23:15	.26
Z GLM, OpenRouter	-18.5	28	14	17:3	.003

Table 19: Stale progressive versus simple-present notes in the dated-note test ($n = 96$ pairs per reader model). Persistence diff. is the progressive minus the simple-present persistence estimate. Simple and Prog. count filled fields; $b : c$ counts pairs filled only for the simple versus only for the progressive note.

Fresh simple-present notes receive mean persistence 94.8–97.4 and are used in 87–96 of 96 cases. Expired bounded notes receive means 0–2.4 and are refused in all 96 cases by each reader model. The stale simple-present form is itself used in only 28–50 of 96 cases, so the displayed date and UNKNOWN option drive much of the abstention. Repeating the persistence question reproduces the stale progressive-minus-simple-present contrast (-29.3, -15.6, and -18.7). DeepSeek’s mean absolute change between identical calls is 4.9 points on the official endpoint but 15.7 points between the official and OpenRouter runs. On OpenRouter, DeepSeek’s field-fill contrast is null despite a similar persistence contrast.

The minimal-edit test isolates the stored verb while holding the note format, date, reader prompt, key fact, and response options fixed. We mechanically restore progressive form in notes written from simple-present statements, or flatten progressive form in notes where mem0 preserved it, then hand-check every edit. Table 20 reports the paired results.

The persistence p -values use two-sided Wilcoxon signed-rank tests; the field-fill values use exact two-sided McNemar tests.

Edit	Reader	n	ΔP (p)	Fills before → after	$b : c$	p
Restore prog.	DeepSeek	37	-13.9 (.00034)	15 → 3	13:1	.0018
	Luna	37	-6.1 (.00045)	18 → 18	6:6	1.0
	Z GLM	37	-9.4 (1.3×10^{-6})	20 → 15	10:5	.30
Flatten prog.	DeepSeek	57	+37.3 (2.6×10^{-8})	1 → 4	1:4	.375
	Luna	57	+18.2 (1.5×10^{-6})	16 → 22	7:13	.263
	Z GLM	57	+24.6 (7.9×10^{-10})	0 → 2	0:2	.50

Table 20: Minimal edits to mem0’s stored notes. ΔP is the persistence estimate after minus before. For field fill, $b : c$ counts items filled only before versus only after the edit. Endpoints follow Table 19; tests are exploratory.

E.1 Verification-tool task

The reader receives one dated note, a user request, and two tools: one completes the request with the remembered value, and one asks the user to confirm it first. The system prompt tells the reader to act if the stored value can be relied on as still current and to ask for confirmation if it may no longer be current. Only the first tool call is scored, and a call counts as acting only if it uses the stored value exactly. In each reader’s main-test cell, all 30 notes written the same day were acted on and all 30 notes that had explicitly expired were confirmed.

The main test crosses the 120 same-verb contents of LAPSE with two or three request wordings, for 291 pairs per reader. Lodging and household are excluded because their edits change the verb. The analysis was written before any target call. It uses a one-sided exact McNemar test per pair and a one-sided sign test per content, each corrected across the three readers. The prediction counted as confirmed only if GLM-5.2 at eight months passed both. GLM-5.2 and DeepSeek V4.1 Flash were tested at note ages set from the calibration runs below. The eight-month GLM row of Table 23 was itself a planned test on 69 pairs; it was not significant after correction ($p = .051$), and the main test is its larger repetition. The age choice used only the rate of acting on simple-present notes, but the same runs produced the form contrast that motivated the repetition. The 69 facts of those runs are among the 120 used here, with new request wordings. Gemini 3 Flash was admitted by a sweep on 30 separate contents, which placed it at one month. Three candidates could not be tested: GPT-5.6 Luna confirms at every age, Mistral Small 3.2 confirmed only 12 of 30 expired notes, and Qwen3.8-Max rejects a required tool call.

After the main test we added three readers. Each used the same items, gates, age rule, and analysis, and each is analyzed on its own. Claude Haiku 5.5 and Gemma 4 31B passed admission but fell outside the age rule. Haiku confirmed every simple-present note at one, three, and eight months, and Gemma acted on every one. Kimi K2.6, on the Moonshot AI endpoint, was placed at three months, where it acted on 18 of 30 simple-present notes, and it passed its cell gate. DeepSeek V4.1 Flash ran on the official endpoint; it is a later model than the DeepSeek V4 Flash used elsewhere in the paper. Table 21 gives the result.

Reader	Months	Simple acted on	Progressive acted on	Simple only : prog. only	Contents + : -	Difference (95% CI)	Corrected <i>p</i>
Z GLM	8	187/291	116/291	91:19	54:9	+25 (18, 32)	2.8×10^{-12}
Gemini 3 Flash	1	135/291	114/291	23:2	19:0	+7 (4, 11)	1.9×10^{-5}
DeepSeek 4.1	1	155/291	144/291	30:19	23:13	+4 (-1, 8)	.076
Kimi K2.6 [†]	3	169/291	191/291	34:56	23:36	-8 (-15, -0.4)	.026 [†]

Table 21: Verification-tool main test (291 pairs, 120 contents per reader). “Contents + : -” counts contents whose pairs lean toward acting on the simple-present or the progressive note. The difference is in percentage points, with a bootstrap interval over contents. *p* is the one-sided exact McNemar test, Holm-corrected across the three readers. GLM ran on Baidu (fp8) through OpenRouter, Gemini 3 Flash on Google AI Studio, DeepSeek on its official endpoint. Three GLM pairs with a truncated value are excluded. [†]Added after the main test, with a two-sided exact *p*, uncorrected. Three unparsed Kimi pairs are excluded.

GLM’s per-pair splits by kind of fact are 27:1 for affiliate role, 31:0 for class, and 12:4 for workplace. They are 12:7 for project and 6:1 for vehicle. Equipment reverses at 3:6, and both of its forms are rarely acted on. Every request wording leans the same way. Gemini 3 Flash never acts on a class note in either form. DeepSeek’s project notes go the other way (4:10). Kimi K2.6 leans toward the progressive note for projects (5:20) and affiliate roles (6:16). It leans toward the simple note for workplaces (8:3) and acts on every class note in both forms.

The mem0 test uses notes written by installed mem0 with DeepSeek V4 Flash as the writer (Section 6). In the preserved subset, mem0 kept the progressive and we flatten the verb by hand (34 pairs). The simple-source subset covers the contents where mem0 flattened the progressive statement; we take the note mem0 wrote from the matched simple-present statement and restore the progressive (37 pairs). The six tests are corrected as one family (Table 22). For GLM, the verb matters on stored

lines without a date, which split 14:1 in the simple-source subset. Lines with a date split 1:1 in both subsets. Gemini 3 Flash’s four reversed pairs in the preserved subset are all equipment or project notes.

Reader	Subset	Simple acted on	Progressive acted on	Simple only : prog. only	Corrected <i>p</i>
Z GLM, 8 mo	preserved simple source	9/34 30/37	7/34 17/37	4:2 15:2	1 .007
Gemini 3 Flash, 1 mo	preserved simple source	10/34 19/37	14/34 14/37	0:4 5:0	1 .16
DeepSeek 4.1, 1 mo	preserved simple source	16/34 14/37	11/34 13/37	6:1 4:3	.25 1
Kimi K2.6 [†] , 3 mo	preserved simple source	21/34 28/37	19/34 26/37	3:1 5:3	.63 .63

Table 22: Verification-tool test on mem0’s stored notes. “Simple” is the flattened or simple-source note. *p* is the one-sided exact McNemar test, Holm-corrected across the six tests. One DeepSeek preserved pair is unparsed. [†]Added after the main test; Holm-corrected across its own two tests.

Before the main test, the same task ran on 69 pairs whose notes came from Graphiti, at one, three, and eight months (Table 23). Its eight-month GLM cell was a planned test and did not pass the correction (raw *p* = .017). These runs set the note ages of GLM and DeepSeek in the main test, and 69 of its 120 contents appear in them with other request wordings.

Reader	Months	Simple acted on	Progressive acted on	Simple only : prog. only	<i>p</i>	Note
DeepSeek 4.1	1	33/69	28/69	5:0	.062	
	3	20/69	14/69	9:3	.146	
	8	5/69	2/69	5:2	.453	bound.
Z GLM	1	61/69	66/69	1:4	.375	bound.
	3	59/69	50/69	14:3	.013	
	8	38/69	28/69	17:5	.017	
Ⓢ Luna	1	1/69	0/69	1:0	1.000	bound.
	3	1/69	0/69	1:0	1.000	bound.
	8	0/69	0/69	0:0	1.000	bound.

Table 23: Verification-tool calibration runs by note age (69 same-verb pairs per cell). “Acted on” counts first tool calls that complete the request with the remembered value instead of asking for confirmation. *p* is an unadjusted two-sided exact McNemar test. “Bound.” marks a floor or ceiling cell (an arm under 10% or over 90%). DeepSeek 4.1 is DeepSeek V4.1 Flash.

E.2 Free-text note and structured record

This test was added after the main experiments and is exploratory. Its analysis was written before any reader call. DeepSeek V4 Flash wrote notes for the 120 contents of Appendix E.1 in progressive and simple-present form, from the memory-write session of Appendix A. Each session received the

baseline memory-write query, and separately the same query with this instruction appended: *Write each note as a record with exactly these four fields, one per line: said: the user’s own words, quoted exactly; fact: the fact in a short normalized form; form: the grammatical form of what the user said (simple present, present progressive, past, future, or other); observed: the session date.* The reader saw the line or record that names the fact.

In free text, the writer rendered 99 of the 120 progressive sources as simple-present states. In records, the said field contained the user’s sentence for all 240 notes. The form field named the progressive for 112 of the 120 progressive sources and the simple present for all 120 simple-present sources. The fact field kept the progressive in 24 notes.

Each reader saw four versions of every note in the verification-tool task: the free-text line, the full record, the record without the said field, and the user’s sentence alone (“user said on [date]: ...”). Gemini 3 Flash read one-month notes and GLM-5.2 eight-month notes, the ages of Table 21, on the same endpoints. Both passed a gate of 30 same-day and 30 expired notes written as records. Two tests were fixed before the reader calls. The first asks whether the form contrast is larger for the record than for the free-text note (one-sided sign test over contents). The second asks whether, on progressive sources, the reader’s decision matches its decision on the user’s sentence more often with the record than with free text (one-sided exact McNemar test). We ran Gemini 3 Flash first and decided to run GLM-5.2 after seeing its result.

The first test passed for both readers. For Gemini 3 Flash, 35 contents leaned toward a larger contrast for the record and 6 the other way ($p = 2.4 \times 10^{-6}$). For GLM-5.2 the split was 64 to 13 ($p = 1.5 \times 10^{-9}$).

The second test passed for GLM-5.2. Its decisions on records matched its decisions on the user’s sentence 218 times, against 176 for free text ($p = 3.2 \times 10^{-5}$). Asked about the user’s sentence a second time, GLM-5.2 repeated its first decision 221 times, so the record matched about as often as a repeat of the same input. The second test failed for Gemini 3 Flash (206 matches against 212). Each version set its own overall rate of acting (Table 24), which limits how often any version can match the user’s sentence. On simple-present sources, the record matched the user’s sentence less often than free text for both readers.

Reader	Note shown	Simple acted on	Progressive acted on	Simple only : prog. only	p
GLM-5.2	Free text	160/278	156/278	23:19	.32
	Record	177/287	82/287	113:18	2.6×10^{-18}
	Record without words	157/285	133/285	49:25	.0035
	User’s sentence	137/279	77/279	75:15	4.6×10^{-11}
Gemini 3 Flash	Free text	129/291	128/291	11:10	.50
	Record	180/291	138/291	45:3	6.6×10^{-11}
	Record without words	161/291	153/291	17:9	.084
	User’s sentence	94/291	53/291	47:6	2.9×10^{-9}

Table 24: The verification-tool task on notes written by DeepSeek V4 Flash as free text or as records. GLM-5.2 read eight-month notes and Gemini 3 Flash one-month notes. Pairs with an unparsed response are excluded. p is a one-sided exact McNemar test, uncorrected.

F Pipeline-to-Reader Details

Each pipeline receives 288 isolated inputs: 96 key facts (witnesses) in progressive, simple-present, and explicitly bounded form. The writer clock is fixed to 2026-01-06, and readers evaluate the stored witness-bearing line on 2026-09-08. A line is labeled preserved when it contains a progressive verb or an explicit ongoing marker such as “currently”; nominal and simple-present rewrites are labeled flattened. Missing witness lines are dropped and counted. Table 25 gives the writer outcomes, and Table 26 how often each reader used the stored facts.

Pipeline/form	Stored	Preserved	Flattened	Dropped
mem0/progressive	94	57	37	2
mem0/simple	96	0	96	0
mem0/bounded	94	76	18	2
Graphiti/progressive	95	48	47	1
Graphiti/simple	96	0	96	0
Graphiti/bounded	96	54	42	0
Letta/progressive	90	28	62	6
Letta/simple	93	1	92	3
Letta/bounded	95	18	77	1

Table 25: Writer outcomes in the pipeline-to-reader experiments. Bounded lines can be labeled flattened after losing progressive morphology while still retaining their explicit end date.

mem0 2.0.19 and Graphiti 0.30.1 use DeepSeek on the official endpoint; Letta 0.16.8 uses DeepSeek through OpenRouter. Reader endpoints match Appendix E. The mem0 test contains 1,064 rows per reader with no unparsed outputs; the Graphiti and Letta test contains 1,704 rows per reader with four unparsed outputs in total. Independent spot-checks found 45/45 field fills correctly parsed in each test.

The mem0 prediction that flattened notes would produce no reader-model contrast fails in two readers because mem0 moves the cue into a date stamp.

Reader	Pipeline	Cue	<i>n</i>	Prog. used	Simple used	Simple only	Prog. only	Raw <i>p</i>
DeepSeek	mem0	verb	57	1	12	12	1	.003
	Graphiti	verb	48	4	19	18	3	.0015
	Graphiti	none	47	18	25	13	6	.17
	Letta	“Currently”	28	0	6	6	0	.031
	Letta	none	62	16	22	13	7	.26
Luna	mem0	verb	57	16	21	12	7	.36
	Graphiti	verb	48	16	29	15	2	.0024
	Graphiti	none	47	22	28	9	3	.15
	Letta	“Currently”	28	10	12	7	4	.55
	Letta	none	62	40	42	12	9	.66
GLM	mem0	verb	57	0	10	10	0	.002
	Graphiti	verb	48	1	13	13	1	.0018
	Graphiti	none	47	22	28	12	6	.24
	Letta	“Currently”	28	0	8	8	0	.0078
	Letta	none	62	23	32	16	7	.093

Table 26: Reader use of each pipeline’s stored facts. “Simple only” and “Prog. only” are discordant matched pairs. Values are exact McNemar tests. Under Holm correction across these 15 tests, five remain below .05: the mem0 verb rows for DeepSeek and GLM and the Graphiti verb rows for all three readers.

For Graphiti, the predicted null on its 47 cue-absent facts holds in all three readers, as does the predicted contrast on its 48 verb-preserved facts. Letta’s pooled null prediction fails in two readers because 28 lines retain a “Currently” prefix; the 62 bare-label lines are null in all three. These nulls do not prove equivalence: their raw counts lean in the predicted direction, and only 40 of 47 Graphiti pairs and 11 of 61 comparable Letta pairs are string-identical. Graphiti sets `valid_at` to the write time on all 287 witness edges and never sets `invalid_at` for a progressive input. It does extract an explicit endpoint from 91 of 96 bounded inputs, contrary to our prediction.

G Benchmark Screen

The LongMemEval input is deduplicated to 1,744 user-turn segments; LoCoMo contributes 2,486 turns. A high-recall marker search followed by contextual inspection yields 80 unique utterances containing progressive or perfect marking. The search was designed to find examples and its recall is unknown, so this count is not a rate. We read the benchmark questions about these utterances and found none that can be answered only by noticing that the statement was in temporary form: when a fact changes, a later turn says so, or the most recent mention answers the question. This is the authors’ reading of a set selected by the search, not a blind count.

To check that the marked forms occur outside curated benchmarks, we also ran the same marker search over 809,281 English user turns of WildChat-1M (Zhao et al., 2024). A hit is a regular-expression match and nothing more. We read all

339 progressive and perfect-progressive hits under a fixed three-question rule (the user’s own first-person statement; a current state with a plausible end; a life circumstance rather than the tool stack of the current question). This yields 57 distinct life-circumstance facts in temporary form, 14 of which match a LAPSE frame almost exactly (“from yesterday I’m staying at my parent’s place”; “I have been taking fluoxetine for approximately 21 months”), and about 200 tool-in-use statements. The bounded and adverbial marker classes had near-zero precision and are excluded. This is an occurrence check, not a rate: recall is unknown, the read was performed by the authors rather than blind annotators, and WildChat users state such facts only when a task requires them. The rule and every hit excerpt are released.

G.1 Writer test on screened utterances

We also passed 104 screened life-circumstance statements through the three confirmatory writers. The set contains WildChat utterances and dialogue from LoCoMo and LongMemEval. All outputs labeled dropped or flat were hand-read. Table 27 gives the results.

Source form	DeepSeek	Luna	GLM
Bare present progressive	9/27	6/28	5/29
Bare perfect progressive	13/27	14/27	12/26
Duration-bearing	5/45	4/45	8/44

Table 27: Loss of temporal qualification in stored real-utterance facts. Denominators include only stored facts. The regex screen is not a prevalence sample, and labels were read by one analyst.

H Reproducibility and Release

The confirmatory protocol was frozen internally on 2026-08-12, before scored outcomes were opened. It was not deposited in a third-party registry, so we describe it as fixed before data collection, not as preregistered. The freeze is commit `a60dcc5b5c0f555b05c3d766e8f50cf324250c3a`; the frozen specification has SHA-256 `8521448b3f012ebac0f131bd2c200e7f14041fb4816e7307a137ec8b86053deb`. The released specification also includes later dated amendments. The estimation rows and exploratory experiments they added are not part of the confirmatory tests.

The release (<https://github.com/aimsresearchlab/lapse>) includes the de-

terministic stimulus builder, reference and run manifests, raw response traces, and cascade and judge outputs. It also includes the blinded gold labels and adjudication rules, the two analysis programs with their expected byte outputs, the mem0 traces, and a claim-to-artifact hash ledger. The generator is v2.0.1 (source SHA-16 58fb635d3b996e71); the reference stimulus file has SHA-16 87561af0713a9fff. Analysis programs are rerun before any figure build, and the build aborts unless their textual outputs reproduce the committed SHA-16 values exactly. Every plotted count is then derived from hash-verified artifacts and serialized to one figure-data JSON file.

All names, companies, institutions, projects, and residences in LAPSE are invented. A contamination screen found no strong web collisions after rotating three weak names. Model outputs can still contain provider-generated text, so the public release should preserve applicable provider terms and redact any request metadata not needed for reproduction. The benchmark evaluates memory systems, not users, and contains no personal conversations.