Sales-Brain: A Grounded Verdict Path for CRM Agents

Consolidated Memory + Code Compute at Organization Scale

Rahul Balakavi
AmpUp Research
rahulb@ampup.ai

***Abstract*: We describe *Sales-Brain*, an operational memory-plus-compute architecture for language-model agents that answer questions over a company's customer-relationship-management (CRM) records. Sales-Brain is built to address a failure mode we characterize separately [1]: on operational CRM questions such as lead qualification, free-text call notes function as a sales pitch and the model is persuaded by them, clearing 29 of 31 deals the company's own records call unacceptable. The remedy is architectural: separate what the model reads from what decides the verdict. Sales-Brain implements this by pairing a consolidated memory with a knowledge base of exact stated facts, an inexpensive extractor that reads the verbatim layer into structured JSON, and a deterministic compute step that applies the policy in code. The memory is built on the open-source OpenViking store [2]. It is consolidated once at user, organization, deal, and meeting granularity and held as one verbatim layer and two distilled layers. Consolidation deduplicates and reconciles conflicts once at write time rather than on every read, and gives downstream systems a stable identity to attach preferences and history to. Its distilled per-account profile is bounded by design (about 2.3 KB at one call per account in our data); whether it stays that size as activity accumulates is a consequence of the design that we have not measured, and the full-text tier grows with events. We report a read-path comparison against retrieval at fixed reader, the layer ablation showing the verbatim tier is indistinguishable from raw text for the computation while the distilled tiers are not, the per-query and per-lead cost of each tier, and the preconditions under which the architecture ceases to apply. Reference numbers for the compute-step control and the cross-model persuasion sweep are cited from the companion paper [1]; here we focus on the system-level design and its operational cost.**

***Index Terms*: Sales-Brain, CRM agents, consolidated memory, extract-then-compute, agent operating cost, CRMArena-Pro.**

I. Introduction: What Sales-Brain Is

Sales-Brain is the memory-plus-compute layer AmpUp deploys underneath language-model agents that operate over a company's CRM. It exists to answer everyday operational questions correctly and cheaply: is this lead worth pursuing, does this quote follow policy, which deals are at risk. It works at the scale of an organization's ongoing activity stream (meetings, emails, cases, calls).

The design is a response to a specific failure mode. When a language-model agent reads a sales transcript end-to-end and reasons directly to a verdict, the representative's claims act as a sales pitch: on the CRMArena-Pro lead-qualification benchmark, a model reading only the transcript clears 29 of 31 deals the company's own records deem unacceptable, and the failure is consistent across seven models from four providers [1]. The remedy is architectural, not a larger model: separate what the model reads from what decides the verdict. The model becomes an extractor of plain facts; a deterministic compute step applies the company's policy in code; and both work over consolidated memory rather than against a fresh corpus each query.

Sales-Brain productizes that separation. Its four components are consolidated memory, a knowledge base of stated facts, an inexpensive extractor, and a deterministic compute step, shown in Figure 2. The rest of this paper is about the architecture: what each layer holds, why the tiering matters, what consolidation buys beyond storage economy, and what the system costs to run at a scale of about 50 representatives generating roughly 125,000 meetings and 2,000 deals per year. The general failure mode that motivates the split, and the cross-model sweep that shows the split fixes it, are quantified in the companion paper [1]; Figure 1 below gives the headline contrast that Sales-Brain was built to obtain.

The failure Sales-Brain addresses is not particular to one model. The cross-model sweep [1] runs the same 31-contradiction case set against seven models from four providers: two current-generation OpenAI models, Claude Sonnet 5 and Claude Fable 5.1, Kimi K2.6, and Qwen 3.8 Max, with prior-generation GPT-4o as reference: every model is misled on at least 27 of the 31 cases; two of them miss all but one (Table I). The grounded procedure recovers 25 to 28 of the cases on every model. Sales-Brain therefore targets a property of the task that no current model handles directly, whichever model is used. The next sections describe the architecture we built around that observation.

II. Sales-Brain Architecture

Sales-Brain is four components governed by one rule (the model reads, code decides) and one design decision

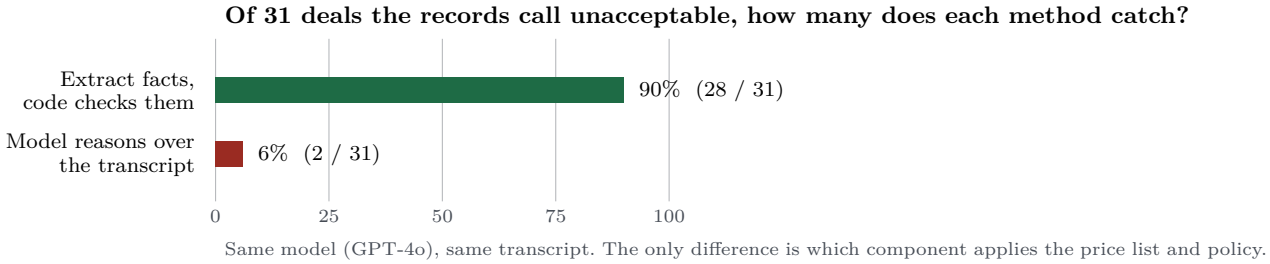

Fig. 1. Headline contrast on GPT-4o, from [1]; per-model figures in Table I. Reading the transcript, the model takes the representative's word and misses nearly every bad deal; the same model extracting facts for code to check catches most of them.

TABLE I
THE 31-CASE CONTRADICTION SET, PER MODEL, FROM [1]. "MISLED" IS THE COUNT OF DEALS THE MODEL CLEARS WHEN READING THE TRANSCRIPT DIRECTLY; "CAUGHT" IS THE COUNT THE EXTRACT-THEN-COMPUTE PATH FLAGS ON THE SAME MODEL. BRACKETS ARE 95% WILSON INTERVALS [15], [16]; AT n=31 THEY ESTABLISH A DIRECTION, NOT A RATE.

Model	Misled (direct)	Caught (extract → code)
GPT-4o (reference)	29/31 [79–98]	28/31 [75–97]
GPT-5	28/31 [75–97]	25/31 [64–91]
gpt-5.6-sol	28/31 [75–97]	28/31 [75–97]
Claude Sonnet 5	30/31 [84–99]	26/31 [67–93]
Claude Fable 5.1	29/31 [79–98]	27/31 [71–95]
Kimi K2.6	27/31 [71–95]	26/31 [67–93]
Qwen 3.8 Max	30/31 [84–99]	27/31 [71–95]

that matters more than any of the individual parts: the same activity stream is consolidated once into memory that is then read by every downstream question.

A. The four components

1) **Consolidated memory** (AmpUp's configuration of the open-source OpenViking context store [2]; all memory measurements in this paper are on that store). Every activity record, whether a sales-call transcript, email, support case, or CRM update, is ingested once, deduplicated against what is already known, reconciled where it conflicts with a prior record, and merged into consolidated records at four granularities: *user* (the individual representative and their history), *organization* (the account being sold to), *deal* (the specific opportunity), and *meeting* (the single call or event). Each granularity is held at three tiers: one verbatim and two distilled, an overview and a longer abstract. The verbatim tier retains the full text; the distilled tiers are summaries kept for read paths where fidelity to quantities is not required: one store that is never lossy, one that is compact. Zep [32] makes the same split between raw episodes and derived summaries.
2) **Knowledge base of stated facts.** A separate store for what the company *decides*, not what it *records*: catalog prices, written installation policies, routing rules, discount ladders. The distinction matters because these change on a different clock (a product manager updates them, not an activity stream), and they are the input against which the compute step adjudicates. Sales-Brain treats them as first-class references: exact, stated, and reused across every query rather than re-derived from context each time.
3) **Inexpensive extractor.** A small language model reading the verbatim tier into structured JSON: products, quantities, stated budget, required timeline, authority markers, need markers. It extracts one field at a time, so a bias in extracting one field cannot contaminate the others. Because the residual task is arithmetic against a catalog, model size buys almost nothing here: an inexpensive extractor at about \$0.0002/query is within 5 points of GPT-4o (84 versus 89, intervals overlapping; Table II).
4) **Deterministic compute step.** Ordinary code (Python, SQL) that applies the policy to the extracted facts: sums quantity×price against the stated budget, maps unit volume against the installation-tier table, enforces routing rules. This step never invokes a model. When the facts are already structured (a database row for a quote, an order line item), the extractor is omitted and the compute step reads directly.

The architecture is shown in Figure 2. The lead-qualification tasks reported here exercise the meeting granularity, so the reported accuracy numbers measure that tier; the user, organization, and deal records are populated by the same ingest but are not separately scored.

B. Memory operations at ingest

Each incoming activity record passes through four operations. They are analogous to Mem0's add/update/no-op [31] and Zep's edge invalidation [32], with two differences: the unit is the whole record rather than an extracted fact, and matching is by identifier where one exists rather than by an LLM judging similarity.

1) **Match.** Candidate existing records for the same user, account, deal, or meeting are found by identifier where one exists (CRM ids, calendar ids, email thread ids) and otherwise by a content signature over normalized participants, date, and the first few

hundred characters, in the sense of record linkage [38].

2) **Merge.** If a match carries no conflicting field, the new record's facts are folded into the existing consolidated record and the source is appended to its provenance list. Nothing is re-extracted.
3) **Supersede.** If a matched field conflicts, as when an earlier call states a budget of $4,000 and a later email revises it to $5,500, the later-dated value becomes current and the earlier one is retained with its validity closed. We adopt Zep's validity interval for this: each field value carries the time it held in the world (t_{valid}, t_{invalid}) and the time it entered the store. Our own rule, not Zep's, decides which value is current: event time first, ingest time second, so a late-arriving record about an earlier call does not overwrite a newer statement. We do not keep Zep's t_{expired}, so "what did the store believe in March" is not answerable, only "what did the customer say in March." Supersede edits the consolidated record's structured fields, the overlay from which the distilled tiers are generated. The verbatim text is never edited. Nothing is deleted.
4) **No-op.** An exact duplicate (the same call synced from two systems) is recorded as a second provenance entry and nothing else changes.

Reconciliation is therefore a *write-time* operation. By design the compute step takes each field's current value from the overlay rather than adjudicating among chunks, which is the failure mode Section III shows is unsafe; in every run reported here the field comes from the extractor's JSON over a single record, so the overlay path is unexercised. When a deal's verbatim record carries two conflicting statements, the extractor still sees both; the overlay is what settles the field. The operations run per granularity; a meeting-level merge does not update the deal or account record in the same transaction. And no number reported in this paper exercises Merge or Supersede: CRMArena provides one transcript per lead, so the multi-touch case is untested here (Section V).

C. Cost per Query

Answering one lead-qualification query costs a single extraction call. On an inexpensive model (GPT-4o mini) this is about $0.0002 per query at list prices [22], roughly two cents for all 100 tasks, and the ladder (Table II) shows the inexpensive reader (GPT-4o-mini) is within the interval of GPT-4o (84 versus 89), so there is no accuracy reason to pay GPT-4o per-query prices; at the top of the ladder the same single call is about $0.004. For the structured tasks of Section VIII the compute path calls no model at all. A tool-using agent [4], by contrast, answers by looping through plan, inspect the schema, write a query, read rows, re-query, and reason. It issues several large-model calls over context that accumulates across turns; a six-call trace is on the order of $0.08 per query (estimated), twenty to four hundred times the single-call cost depending on the reader, and, as Fig. 1 shows, an answer more likely to be wrong.

TABLE II
THE METHOD LADDER. TRAIN (n=80) AND TEST (n=20) ARE THE CRMARENA 80/20 SPLIT; "FULL" IS ALL 100 TASKS. BRACKETS: 95% WILSON INTERVALS. 84 AND 89 OVERLAP; 88 [78–93] AND 90 [70–97] BOTH CONTAIN 89.

Method	Reader	Tr.	Te.	Full
Direct from transcript	4o-mini	44	40	41 [32–51]
+ chain-of-thought	4o	56 [45–67]	–	–
+ records, model computes	4o	28 [19–38]	–	–
Extract → compute in code	4o	**88**	**90**	**89** [81–94]
same, inexpensive reader	4o-mini	–	–	**84** [76–90]

III. Grounded Verdict Evaluation

The design decisions above rest on measurements reported in the companion paper [1], which we summarize here at the level of detail needed to justify each choice; the full same-information control, compute-step control, and cross-model sweeps are in that document. Throughout, "strict" means the predicted set of failing factors equals the gold set exactly; the factors are the BANT four (Budget, Authority, Need, Timeline), and a lead that fails none is graded *None*. Brackets after a score are 95% Wilson intervals [15], [16].

The four rungs (Fig. 3, Table II). Direct-from-transcript scores 41% (GPT-4o-mini, full set). Chain-of-thought [23] lifts it to 56% and plateaus. Handing the model the catalog and policy and asking it to reason to a verdict *lowers* strict accuracy to 28% (GPT-4o, training split): recall of true failures rises, precision collapses. Extract-then-compute reaches 88–90% strict accuracy, and 84% on the inexpensive extractor, inside the larger reader's interval (Table II). What moves the result is who applies the policy, not which model reads.

Against retrieval (Table III). The natural alternative to a consolidated record is to chunk and embed everything and retrieve at query time. A fair comparison has to vary the substrate and the compute step separately, so we ran both. Retrieval with the reader reasoning to a verdict scores 42% on GPT-4o and 16% on GPT-4o-mini. Retrieval with the same extractor and code underneath scores 75% and 76% when the index holds the knowledge base and the transcript together, and 83% and 81% when it holds the transcript alone. The consolidated verbatim record with extractor and code scores 89% and 84%. The two steps separate cleanly. Most of the gap is the compute step: moving from "model reasons" to "extract → code" is worth 33 points on GPT-4o and 60 on GPT-4o-mini whichever store is underneath, the same finding as the "records to the model" rung of the ladder. The substrate matters much less, and most of its effect comes from index configuration. With the knowledge base and the transcript in one index, retrieval trails the consolidated record by

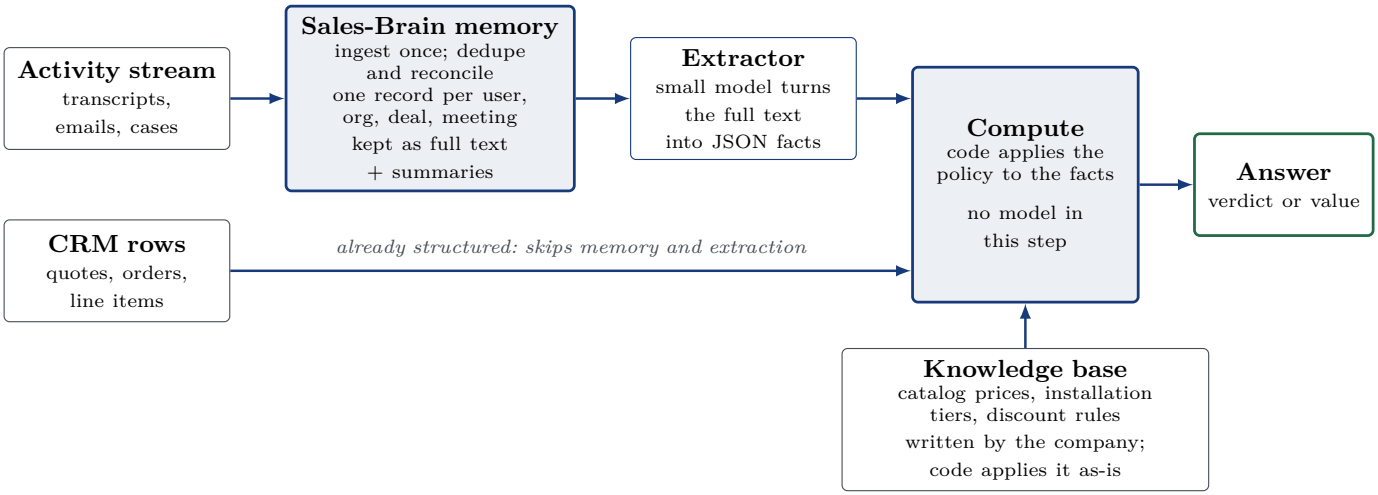

Fig. 2. The Sales-Brain data flow. Free text from the activity stream (transcripts, emails, cases) is ingested once into Sales-Brain memory, which deduplicates and reconciles it into one consolidated record per user, organization, deal, and meeting. Each record is kept in two forms: a *full-text* copy (the "verbatim layer"), which preserves everything the extractor might need for arithmetic, and *summaries* (a short "overview" and a longer "abstract," together the "distilled layers") that serve cheap reads where full fidelity is not needed; later sections use these short names. A small model reads the full text into JSON facts. Records that are already structured (quotes, orders, line items) skip both memory and extraction. Both paths end in the same compute step, where ordinary code applies the company's written policy (catalog prices, installation tiers, discount rules) to the facts. No model runs in that step, so the verdict cannot be argued with by the transcript.

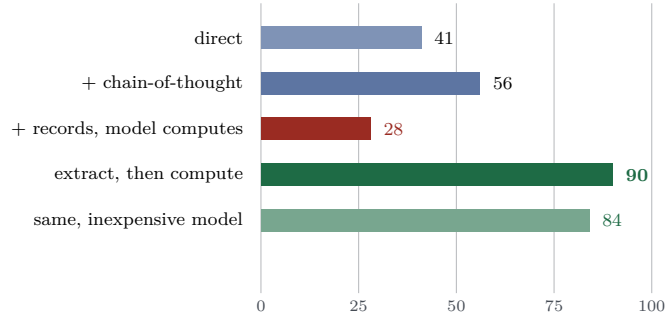

Fig. 3. Accuracy on lead qualification by method (strict; readers and splits as in Table II, so the bars are not all on the same n). Only the last rung, where code applies the policy, changes the picture. Giving the model the records to reason over scores below giving it nothing, because it over-flags.

TABLE III

SUBSTRATE AND COMPUTE STEP, SEPARATED, ON THE 100 LEAD-QUALIFICATION TASKS (STRICT). THE RAG ARMS EMBED THE KNOWLEDGE BASE AND THE LEAD'S TRANSCRIPT IN 600-CHARACTER CHUNKS AND RETRIEVE THE TOP 8 FOR THE BANT QUESTION; THE † ARMS INDEX THE TRANSCRIPT ALONE, SINCE THE CODE ALREADY HOLDS THE CATALOG AND TIERS; THE SALES-BRAIN ARMS READ THE CONSOLIDATED VERBATIM RECORD. "MODEL REASONS" LETS THE READER REASON TO A VERDICT; "EXTRACT → CODE" FEEDS THE SAME CONTEXT TO THE EXTRACTOR AND COMPUTES IN CODE. LATENCY FOR THE "MODEL REASONS" AND SALES-BRAIN ROWS IS THE MEDIAN OVER 8 LEADS IN ONE SESSION, A DIFFERENT SESSION FROM TABLE V, AND THE TWO DISAGREE BY ABOUT 30% FOR THE SAME CONFIGURATION, WHICH IS THE SESSION-TO-SESSION SPREAD OF ONE MODEL CALL; FOR THE RAG EXTRACT ROWS IT IS THE MEDIAN OF THE EXTRACTION CALL OVER ALL 100 LEADS, EMBEDDING ROUND-TRIPS EXCLUDED. COST IS AT LIST PRICES; THE RAG EXTRACT ROWS' COST IS ESTIMATED FROM THE OTHER ROWS' TOKEN COUNTS, NOT METERED. THE RAG MINI READER'S SLOWER READ IS UNEXPLAINED; WE DID NOT INVESTIGATE IT.

Read path	Reader	Strict	Lat.	$/q
RAG, model reasons	GPT-4o	42 [33–52]	0.6 s	0.0025
RAG, model reasons	GPT-4o-mini	16 [10–24]	2.1 s	0.0003
RAG, extract → code	GPT-4o	75 [66–82]	1.0 s	0.003
RAG, extract → code	GPT-4o-mini	76 [67–83]	1.2 s	0.0003
RAG†, extract → code	GPT-4o	83 [74–89]	1.1 s	0.003
RAG†, extract → code	GPT-4o-mini	81 [72–88]	1.5 s	0.0003
Sales-Brain, extract → code	GPT-4o	**89** [81–94]	1.5 s	0.004
Sales-Brain, extract → code	GPT-4o-mini	**84** [76–90]	1.3 s	**0.0002**

8 to 14 points: the 53 knowledge-base chunks compete with the transcript's 6–8 chunks for the eight slots, and on some leads the chunk carrying a quantity or a budget is the one that falls out. This is the selection failure the organizational-memory literature describes [35], a figure that is not the fragment most similar to the question going unretrieved. But in this arm the knowledge-base chunks are dead weight, since the code already holds the catalog and tiers; scoping the index to the transcript recovers 8 of the 14 points on GPT-4o and 5 of the 8 on GPT-4o-mini, and what remains (89 against 83, 84 against 81) is inside the intervals. The honest claim for consolidation on this benchmark is therefore narrow: it does not have a retrieval configuration to get wrong, and it reads the record whole. Whether a real substrate effect survives on transcripts long enough that k=8 cannot return all of them is untested here.

Same-information control. With the identical inputs (catalog + policy + transcript) and the identical model, sending the records to *code* scores 85% strict; sending them to the *model* scores 18% [1, Table IV]. (The ladder's 28% is the same "model applies the policy" arm under a different prompt on the training split; the two are not the

TABLE IV
PER-FACTOR RECALL AT THE TOP OF THE LADDER (GPT-4O). CODE COMPUTES BUDGET AND TIMELINE; AUTHORITY, NEED, AND *NONE* ARE THE EXTRACTOR'S BOOLEANS PASSED THROUGH. OVER-FLAGGING, WHICH CAUSES THE REMAINING STRICT MISSES, IS NOT SHOWN.

Factor	Correct	Answerable from...
Authority	27/27	transcript content
None (qualifies)	20/20	transcript content
Budget	18/19	catalog prices + computation
Need	12/15	transcript (subjective)
Timeline	9/11	install policy + computation

same measurement.) This is what justifies Sales-Brain's insistence on the deterministic compute step: the records are not what is missing, and adding them to the model's prompt makes things worse.

Compute-step control. With one extraction call, the same four fields, and the catalog in context, we graded the model's own boolean against code's recomputation from the very same extracted JSON. On GPT-4o-mini the gap is +42 points to code; on GPT-5 and o3-mini it pools to +14 to +19 over two seeds; on Claude Sonnet 5, Fable 5.1, gpt-5.6-sol, Kimi K2.6, and Qwen 3.8 Max it narrows to 0–+5 [1, Fig. 4]. What code guarantees on every model is a soundness property, not a uniform lift: it never lowered either metric. Two consequences follow. The correction narrows toward zero on models that already compute correctly, and the code path is *fail-safe against fabrication*, because a failed catalog lookup suppresses a Budget verdict instead of inventing one. These are why Sales-Brain routes the arithmetic to code even when a large model would answer correctly on its own.

Two encodings that mattered. Both affected the accuracy figures more than model choice. First, the installation policy assigns a timeline tier by *floor*: encoding it as a ceiling inverts small orders into false Timeline failures and costs about 50 strict points on the training split. Second, CRMArena's multi-factor gold answers are stored as a comma-joined string; splitting both sides before set comparison recovered several points that were never model errors. Both apply to any evaluator on this benchmark and are worth stating so readers do not misdiagnose extractor error.

Before measuring anything we sorted the 22 B2B task types by one question, fixed in writing in advance: can the answer be read off what the activity records say, with no arithmetic over rows, no numbered rulebook, and nothing outside the ingested data? Two types pass. We carry lead qualification through because it sits on the boundary between reading and computing. The other, `knowledge_qa`, is scored by token-overlap F1 [24] against short reference strings, which marks correct paraphrases wrong, so its score says little about answerability.

IV. Consolidation, Deduplication, and Conflict Resolution

The grounded-verdict argument in Section III explains why Sales-Brain uses code to compute; it does not explain why the substrate is a consolidated memory rather than a fresh corpus retrieved per query. Three properties fall out of consolidation that a per-query retriever cannot recover after the fact.

Deduplication at write time. A single deal typically generates overlapping records across systems: a Salesforce activity, a Gong call summary, an email thread, an Outreach touchpoint. Ingesting each into an append-only index leaves the reader, model or code, to weight or discount duplicates at read time. Sales-Brain resolves at write: candidate records for the same deal are matched by identifier plus content signature [38], and only the surviving canonical record enters the tier that downstream questions read. On the one corpus we ingested (the MediLux organization: 101 accounts, 14,134 activity records), the one deal we examined went from 219 raw activities to 164 consolidated records, a 25% reduction. That is one deal's rate, without ground truth for which merges were correct; a false merge across two deals is the case that would damage a verdict, and we have not measured its frequency. A retriever operating without this step would serve the duplicate top-k chunks; Sales-Brain removes them once and pays no inference-time cost for the choice.

Conflict reconciliation at write time. When two records make contradictory claims about the same field, the Supersede rule of Section II-B (event time first, ingest time second) is applied once, at ingest, and the earlier value is retained with its validity closed but does not enter the verdict path. This is what makes the consolidated record *trustable* for the compute step: the current value of each field is settled before any read, rather than left for the reader to choose between two statements. A retriever that returns both chunks and lets the model reconcile them is exactly the mode that Section III shows is unsafe.

Stable identity for personalization and history. The distilled per-account profile is bounded by design: its depth is fixed, so it should not grow with the number of interactions. We measured it at about 2.3 KB per account with one call per account, which shows it is small, not that it stays so; the growth curve is unmeasured. The verbatim tier is not bounded: it grows with events, less the deduplication rate, and the vector index over records grows with it (Table VI). This bounded profile is where preferences, prior objections, and per-account historical context attach. A stable identifier ties every downstream question (a coaching signal, a next-best-action suggestion, an alert on a stalled deal) to the same record, so downstream systems do not need to re-identify the account on every call and consolidations propagate to all consumers at once.

Dedup, conflict reconciliation, and a stable per-account

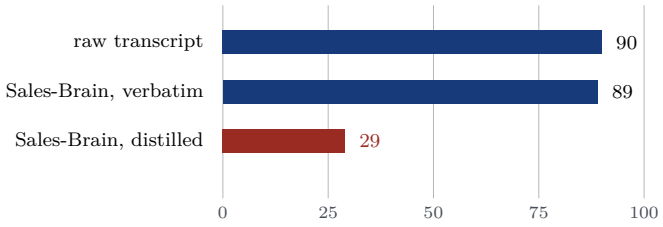

Fig. 4. Same pipeline, different context source (strict, n=100, GPT-4o). The verbatim layer loses nothing against raw text; the distilled layers lose the facts the computation requires, which is why Sales-Brain routes quantitative questions to it.

TABLE V
WHAT A QUERY COSTS BY CONTEXT SOURCE, GPT-4O-MINI READER. TOKENS ARE MEANS OVER THE 100 TASKS; LATENCIES ARE MEDIANS OVER 8 LEADS ON A FRESH SERVER, SPLIT INTO FETCHING THE CONTEXT AND THE EXTRACTION CALL. QUERY LATENCY IS BOUND BY THE MODEL CALL AND IS THE SAME ACROSS TIERS TO WITHIN NOISE; THE DISTILLED TIER'S NINE-FOLD TOKEN SAVING DOES NOT SHORTEN THE CALL AT THIS SIZE. BRACKETS: 95% WILSON INTERVALS.

Context source	Tokens	Fetch	LLM	Total	Strict
Raw transcript	886	4 ms	1.06 s	1.06 s	86 [78–91]
Sales-Brain, verbatim	964	95 ms	0.91 s	1.01 s	84 [76–90]
Sales-Brain, distilled	104	141 ms	0.93 s	1.07 s	31 [23–41]

identity together make the design case for the consolidated substrate. The verdict-path accuracy alone would be recoverable with a well-instrumented retriever; deduplication, reconciliation, and a stable identity are what a retriever does not provide, and Section VI shows they are not cost advantages at this scale.

V. Which Layer Produces the Result

Sales-Brain holds each record at three tiers, so the question is which one the verdict path should read. We ran the identical extract-then-compute pipeline while varying only the context source: the raw transcript from the database; the same transcript read from Sales-Brain's verbatim layer; and the distilled layers (abstract plus overview) (Fig. 4). Every controlled run in Section III reads the verbatim layer.

The verbatim layer is indistinguishable from raw text on this task: 90% [83–94] against 89% [81–94]. Ingesting into Sales-Brain costs no measurable accuracy, and the accuracy itself comes from the extract-then-compute path rather than from storage; the store does not get credit for a result the pipeline produced. Distillation, by contrast, is lossy for quantitative work: the distilled layers drop to 29% [21–39], because the products, quantities, and stated figures the deterministic step requires are what a summary discards. The two facts together are the case for tiering: one store serves both the compact profile reads of Section IV and the exact extraction of Section III, provided each question is routed to the layer that holds what it needs.

The natural objection is to skip the store and read raw records. Matching raw text is the entry condition that makes consolidation admissible. The benefits come from what consolidation adds on top. Once it holds, everything the store adds comes at no accuracy cost: each activity ingested once and deduplicated, conflicts reconciled at write time rather than re-litigated at every read, a distilled profile whose size is bounded (Section VI), and the same ingest yielding user, organization, and deal records alongside the meeting record scored here. What the raw path cannot offer is the routing: the distilled layers that make everyday reads inexpensive exist only because the store exists, and the verbatim layer guarantees they never cost a quantitative verdict. We claim the consolidation benefits as design; only the lossless floor and the one-call footprint are measured in this paper. The experiment that would test the rest is a multi-touch set with at least two activities per deal, a known revised field, true duplicates from two systems, and out-of-order arrival, run with Supersede on and off under the same extractor and code. We have not run it.

Table V puts the per-query cost next to the accuracy; the architecture is often sold on the opposite of what it shows. At one call per lead the verbatim record costs about 9% more tokens to read than raw text, because it wraps the transcript in a short header. Query latency is set by the model call and is the same for all three sources to within noise: the distilled tier sends nine times fewer tokens and is not faster. Its saving is in tokens, and therefore in cost at scale, not in wall-clock; and the verbatim fetch, while some twenty times slower than a SQLite read, is under a tenth of the total.

VI. Cost and Scaling

How the data is stored is separate from how it is answered, and the accounting does not favor the common claim. We compared a naive retrieval-augmented (RAG) index [5] against Sales-Brain's LLM-extracted records over the same corpus: 101 accounts, 14,134 activity records, and 21.8M characters.

Extraction is one model call per activity, so Sales-Brain costs roughly 200 times as much to build as an embedding index, and at one call per lead its full footprint (verbatim tier, distilled tiers, and the vector index over them) is about 72 times the raw text (Table VI). The claim that consolidated records are cheaper is false at write time and false at this scale in storage. What is true is narrower. The distilled profile is bounded by design: its depth is fixed, so it should not grow with the number of interactions on an account, whereas a retrieval index retains every chunk of every call [25]. The verbatim tier grows, at about three quarters the rate of raw text on the one deal we measured. If both hold, a crossover exists at which the bounded profile plus a deduplicated verbatim tier is smaller than an append-only index; by the design it sits at hundreds of interactions per account, far from the one call per lead

TABLE VI

What each store costs to build and to keep. Build cost is over the 14,134-record corpus; per-lead footprint is measured on the 100-lead set, where one activity per account makes "per lead," "per account," and "per activity" coincide. The distilled profile is the only component whose size is bounded by design; the verbatim tier and the vector index over it grow with events, as a retrieval index does. The build index is 12,519 chunks of 512 tokens with 15% overlap; the read-path RAG arms of Table III use 600-character chunks. Sales-Brain's index is denser than the RAG index because it embeds every tier of every record rather than one vector per chunk: per lead, about 116 KB of record files and 138 KB of index. †At about three quarters the rate of raw text on the one deal measured, because duplicates are removed at write.

Store	Build	Per lead	vs raw	Grows
Raw transcript text	∼$0	3.5 KB	1×	events
RAG index	$0.13	∼6 KB	∼2×	events
Sales-Brain, verbatim	∼$29	4.3 KB	1.25×	events†
Sales-Brain, distilled	incl.	∼2.3 KB	0.7×	**bounded**
Sales-Brain, all + index	incl.	∼250 KB	∼72×	records

this benchmark provides, and we have not observed it. And the profile that is bounded is the distilled layer that scored 29% on the quantitative task, so it is the verbatim-layer routing of Section V, not the compact profile, that carries any task requiring a specific figure.

A. Operating Cost at Scale

What the measured numbers imply at scale can be stated without a total-cost-of-ownership model. For the 50-representative organization of Section I, generating about 125,000 meetings a year, the build cost of Table VI scales to about $260 of extraction per year ($29 per 14,134 records), and the full footprint of about 250 KB per meeting scales to roughly 31 GB per year against about 750 MB for the embedding index: Sales-Brain is some forty times larger in storage and two hundred times more expensive to build, and both are small in absolute terms. The per-query cost is $0.0002 on the inexpensive reader. What these numbers do not include is the labor to keep either store trustworthy by deduplicating, resolving conflicts, and evicting stale entries. We believe that labor dominates, and we have not measured it; an earlier draft carried an illustrative labor model and we have removed it, because a savings figure resting on an assumed headcount is not a measurement and should not sit in a table beside ones that are.

VII. Where the Approach Ceases to Apply

Sales-Brain's compute step is only as good as the policy it encodes, so its reach ends where a policy cannot be written from the inputs. The companion paper [1] tests this directly with a prediction, recorded before any code ran, that `invalid_config` (audit a quote, name the violated policy article) would exceed 85%. It did not: identical inputs carry different answers across tasks, which caps every method near 68%. `policy_violation_identification` behaves the same way. Table VII states the resulting deployment rule for Sales-Brain.

TABLE VII

When to use Sales-Brain's compute step.

Use it only when the policy is exact *and* its inputs appear in the records. Lead qualification qualifies; quote auditing on this benchmark does not.

VIII. Structured Tasks: Trivial in Code, but Largely Already Solved

Lead qualification required the language model because the facts resided in a transcript. Many CRM task types are unlike it: the facts are already structured rows, so the extraction step vanishes and only the compute step remains. There, extract-then-compute reduces to a compiled solver that parses the parameters from the question, runs the computation in code, and uses no model. This is the setting text-to-SQL systems address with a model in the loop [41]. We wrote such solvers for the deterministic analytical and routing types, binding every parameter from the same question text the agent receives and never reading the gold answer, and evaluated them on the held-out test split.

Twelve deterministic types behave in this way (Table VIII). The comparison carries two caveats. The agent column is CRMArena-Pro's own figure on a different set, and each solver's logic was written offline, so the gap is an upper bound rather than a fair contest. More consequential, most of these types are functions a CRM platform already performs deterministically: `quote_approval` and `invalid_config` correspond to the price and product rules of a configure-price-quote system, and the analytical types are ordinary reports. The practical conclusion is therefore narrow: a language model should not be used to perform arithmetic that a query already performs. The novel contribution lies at the other end of the paper, in lead qualification, where the input is unstructured, the source is unreliable, and the platform is weak.

IX. Related Work

Sycophancy. Sharma *et al.* [6] and Perez *et al.* [7] document that assistants tailor their answers to a user's stated view; Wei *et al.* [8] show the effect persists even for objectively checkable arithmetic and can be reduced with synthetic fine-tuning data. Our setting differs in one respect. The persuasive party is a third party recorded in the context, a sales representative whose words sit in the CRM and whose incentive is optimism, and not the user. The model agrees with a witness, not with the person it is talking to. We know of no prior measurement of this incentive-misaligned-witness case; the companion paper [1] isolates it, and this paper builds the system around it.

Code as the calculator. Models are unreliable at the multi-step arithmetic a verdict needs [26], [27], and

TABLE VIII

COMPILED SOLVERS ON THE HELD-OUT TEST SPLIT (n=20 PER TYPE, SO EACH COMPILED FIGURE IS A MULTIPLE OF 5). THE AGENT COLUMN IS THE GPT-4O AGENT ACCURACY REPORTED BY CRMARENA-PRO ON ITS FULL TASK SET [3], REPRODUCED FOR REFERENCE; IT IS NOT A LIKE-FOR-LIKE CONTROL, BOTH BECAUSE IT IS MEASURED ON A DIFFERENT SET AND BECAUSE EACH SOLVER'S LOGIC WAS WRITTEN OFFLINE AGAINST THE TRAINING SPLIT. THE GAP IS AN UPPER BOUND ON WHAT COMPILING CAN PROVIDE.

Task type	Compiled (test, n=20)	GPT-4o agent [3]
monthly_trend_analysis	**100%**	15%
quote_approval	100%	8%
handle_time	100%	41%
case_routing	100%	52%
lead_routing	100%	0%
sales_cycle_understanding	100%	13%
top_issue_identification	100%	36%
transfer_count	100%	25%
sales_amount_understanding	95%	40%
conversion_rate_comprehension	95%	23%
best_region_identification	85%	57%
wrong_stage_rectification	85%	46%

handing that arithmetic to an interpreter is established: PAL [9] and Chain of Code [10] run model-written programs, and Toolformer [11] and ReAct [4] call tools mid-reasoning. Sales-Brain differs only in that the program is a fixed, reviewed policy and the model supplies facts, not code. The architecture is not our contribution, nor is the diagnosis, which is the companion paper's [1]. What this paper adds is the system evidence: the substrate-versus-compute separation of Table III, including that most of the retrieval gap is index configuration; the layer ablation; and the cost model.

Agent memory systems. Mem0 [31] extracts salient facts from conversation and applies one of add, update, delete, or no-op against the existing store; Zep [32] builds a temporal knowledge graph in three tiers (episodes, entities, communities) with bi-temporal validity on every fact; A-MEM [33] organizes notes Zettelkasten-style, links them at write time, and rewrites existing notes when a new one arrives; MemGuard [34] filters contaminating or stale facts at ingest rather than at retrieval. Sales-Brain's ingest operations (Section II-B) are analogous to these and adopt Zep's validity intervals; the memory half of the system is not a contribution over them, and we have not run either as the substrate under our extractor, which would be the direct comparison. The differences that matter are in what is evaluated. These systems are evaluated on conversational recall benchmarks such as LoCoMo [36] and LongMemEval [37], where the question is whether a fact from an earlier session can be recovered; Zep's authors note that no existing benchmark tests synthesis of conversation history with structured business data [32]. Lead qualification over CRMArena-Pro is exactly that: a transcript, a price list, and a policy table, graded against a verdict. And the operational-memory work closest in spirit, the process-atom architecture of [35], curates procedural rules from policy documents and shows that retrieval misses cross-cutting constraints that are not lexically similar to any one request; our knowledge-base component is that argument applied to prices and tiers; the 8–14 point cost of a mixed index in Table III, most of which a transcript-scoped index recovers, is a small instance of it.

Benchmarks and memory systems. Agent benchmarks set inside business software, such as CRMArena-Pro [3], τ-bench [12], WorkArena [13], and AppWorld [14], score end-to-end task success. Sales-Brain sits on the memory side of the literature: retrieval-augmented generation [5], [28] fetches chunks at query time, Generative Agents [29] and MemGPT [30] manage a growing stream or a paged window, and Mem0 [31] and OpenViking [2] store extracted, deduplicated facts. Our system-level finding (Sections IV and V) is about which tier of that store should feed a quantitative verdict: the verbatim one, and to code rather than to the model.

X. LIMITATIONS AND CONCLUSION

Everything here rests on one benchmark, one grader, and one task type carried to completion; Section VIII adds breadth only for deterministic computation, and the one pre-specified generalization test failed for reasons Section VII gives. The corpus is 101 synthetic B2B accounts with one call per lead, so the data is cleaner than a production CRM and favors a fixed pipeline over an adaptive agent. The per-model sweep (Table I) has $n = 31$ and fixes a direction, not a rate. Two system claims are design consequences we have not measured: that the distilled profile stays bounded as activity accumulates (at one call per lead the full footprint is about 72 times raw text), and the memory operations of Section II-B.

Before Sales-Brain acts without a human in the loop, three gaps must close. Queries must run under the caller's record permissions rather than directly against the store. The system needs an abstention signal [42]; point accuracy of 84–90% is not a routing policy. And extraction must be validated, because a misread quantity [43] becomes a

confidently wrong verdict: the design relocates error from reasoning to extraction rather than eliminating it.

The claim that survives is this. When a model answers a question about a deal, it is persuaded by a source with an incentive toward optimism, and clears deals that should be flagged; the remedy is a division of labor in which the model reads text into facts and ordinary code checks the facts against the company's own records. Sales-Brain is that division built into one system: consolidated user, organization, deal, and meeting records, ingested once and reconciled at write, with quantitative questions routed to the verbatim layer and answered by code. Under it an inexpensive model is indistinguishable from an expensive one (Table I, Table III). What the system's memory adds beyond that is, in this paper, design rather than measurement, and we have said where. The question to ask before reaching for a larger model is whether the policy is exact and the applicable rule is identifiable from the inputs. If it is, take the arithmetic out of the model and put the effort into extraction. If it is not, no model and no amount of in-context grounding will help; write the policy or accept what the inputs can support.

Appendix A
A Traced Response

The companion paper [1] traces one lead-qualification request through both paths in full. Briefly: on the extract-then-compute path an inexpensive model returns the product, quantity, stated budget, required days, and the Authority and Need booleans; code looks up the unit price and install tier and marks Authority, Budget, and Timeline as failing, in one call at about $0.0002. A tool-using agent instead inspects the schema, writes and re-runs queries, reasons over the rows, accepts the representative's assurances, and clears the lead, in six large-model calls.

Appendix B
The Compiled Skill

The validated algorithm for a task type, written once as a function that parses its parameters from the question text and runs a pre-written query. It never reads the gold answer and contains no model.

```
# audit a quote's discounts against the stated
# Volume-Based Discounts policy; the allowed
# discount tier is set by each line's quantity.
def expected_discount(qty):
    if qty >= 20: return 15.0
    if qty >= 10: return 10.0
    if qty >= 5:  return 5.0
    return 0.0

def solve_quote_approval(conn, task):
    qid = re.search(r"0Q0\w+",
        task["metadata"]["required"]).group()
    rows = conn.execute(
        "SELECT Quantity, Discount FROM "
        "QuoteLineItem WHERE QuoteId=?", (qid,))
    for qty, disc in rows:
        if float(disc) != expected_discount(float(qty)):
            return "ka0Wt000000Eq0MIAS"  # violated article
    return None                          # compliant

# no model in the loop
SOLVERS = {"quote_approval": solve_quote_approval, ...}
answer = SOLVERS[task["task"]](conn, task)
```

Reproducibility

Accuracy, footprint, and build-cost figures are measured; the agent per-query cost is an estimate and is marked as one. Two artifact sets back this paper. The lead-qualification harness, task classification, generalization-test pre-specification, and compute-step and cross-model result files are shared with the companion paper and ship as its ancillary files [1]. The Sales-Brain-specific scripts and results ship as ancillary files with this paper: memory ingestion and consolidation, the per-account event-stream builder, the read-path runs, footprint and cost models, and latency measurements. Model identifiers, temperatures, and the data split are as listed in the companion paper. This is an independent analysis of the public CRMArena-Pro B2B tasks and data, neither affiliated with nor endorsed by the benchmark's authors.

Data and license. The B2B records were read from the benchmark's offline SQLite materialization, redistributed in public GitHub mirrors of the CRMArena code, because the live Salesforce organization is not openly accessible; tasks are the Hugging Face release. CRMArena-Pro is CC BY-NC 4.0. This is non-commercial research and no benchmark records are redistributed here.

Conflict of interest. The author is a co-founder and employee of AmpUp, which sells Sales-Brain. All figures come from the public benchmark and can be reproduced from the released artifacts.

References

[1] R. Balakavi, "Persuaded, Not Informed: Incentive-Misaligned Witnesses Defeat In-Context Grounding," arXiv preprint arXiv:2609.28854, 2026 (companion paper).

[2] Volcengine, "OpenViking: A context database for AI agents," open-source software, 2026. [Online]. Available: https://github.com/volcengine/OpenViking

[3] K. Huang *et al.*, "CRMArena-Pro: Holistic assessment of LLM agents across diverse business scenarios," Salesforce Research, 2025, arXiv:2505.18878.

[4] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2023, arXiv:2210.03629.

[5] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020, arXiv:2005.11401.

[6] M. Sharma *et al.*, "Towards understanding sycophancy in language models," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2024, arXiv:2310.13548.

[7] E. Perez *et al.*, "Discovering language model behaviors with model-written evaluations," in *Findings of the Association for Computational Linguistics (ACL)*, 2023, arXiv:2212.09251.

[8] J. Wei, D. Huang, Y. Lu, D. Zhou, and Q. V. Le, "Simple synthetic data reduces sycophancy in large language models," 2023, arXiv:2308.03958.

[9] L. Gao, A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, J. Callan, and G. Neubig, "PAL: Program-aided language models," in *Proc. Int. Conf. Machine Learning (ICML)*, 2023, arXiv:2211.10435.

[10] C. Li *et al.*, "Chain of Code: Reasoning with a language model-augmented code emulator," in *Proc. Int. Conf. Machine Learning (ICML)*, 2024, arXiv:2312.04474.

[11] T. Schick *et al.*, "Toolformer: Language models can teach themselves to use tools," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023, arXiv:2302.04761.

[12] S. Yao, N. Shinn, P. Razavi, and K. Narasimhan, "τ-bench: A benchmark for tool-agent-user interaction in real-world domains," 2024, arXiv:2406.12045.

[13] A. Drouin *et al.*, "WorkArena: How capable are web agents at solving common knowledge work tasks?" in *Proc. Int. Conf. Machine Learning (ICML)*, 2024, arXiv:2403.07718.

[14] H. Trivedi *et al.*, "AppWorld: A controllable world of apps and people for benchmarking interactive coding agents," in *Proc. Annu. Meeting Assoc. Computational Linguistics (ACL)*, 2024, arXiv:2407.18901.

[15] E. B. Wilson, "Probable inference, the law of succession, and statistical inference," *J. Amer. Statist. Assoc.*, vol. 22, no. 158, pp. 209–212, 1927.

[16] L. D. Brown, T. T. Cai, and A. DasGupta, "Interval estimation for a binomial proportion," *Statistical Science*, vol. 16, no. 2, pp. 101–133, 2001.

[17] OpenAI, "GPT-4o system card," 2024, arXiv:2410.21276.

[18] OpenAI, "OpenAI o3-mini system card," Jan. 2025.

[19] OpenAI, "GPT-5 system card," Aug. 2025.

[20] Anthropic, "Claude Sonnet 5 and Claude Fable 5.1 model documentation," 2026.

[21] Kimi Team, "Kimi K2: Open agentic intelligence," 2025, arXiv:2507.20534.

[22] OpenAI, "API pricing," https://openai.com/api/pricing, accessed Sept. 2026.

[23] J. Wei *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022, arXiv:2201.11903.

[24] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proc. EMNLP*, 2016, arXiv:1606.05250.

[25] N. F. Liu *et al.*, "Lost in the middle: How language models use long contexts," *Trans. Assoc. Computational Linguistics*, vol. 12, 2024, arXiv:2307.03172.

[26] N. Dziri *et al.*, "Faith and fate: Limits of transformers on compositionality," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023, arXiv:2305.18654.

[27] K. Cobbe *et al.*, "Training verifiers to solve math word problems," 2021, arXiv:2110.14168.

[28] V. Karpukhin *et al.*, "Dense passage retrieval for open-domain question answering," in *Proc. EMNLP*, 2020, arXiv:2004.04906.

[29] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Proc. ACM Symp. User Interface Software and Technology (UIST)*, 2023, arXiv:2304.03442.

[30] C. Packer *et al.*, "MemGPT: Towards LLMs as operating systems," 2023, arXiv:2310.08560.

[31] P. Chhikara, D. Khant, S. Aryan, T. Singh, and D. Yadav, "Mem0: Building production-ready AI agents with scalable long-term memory," 2025, arXiv:2504.19413.

[32] P. Rasmussen, P. Paliychuk, T. Beauvais, J. Ryan, and D. Chalef, "Zep: A temporal knowledge graph architecture for agent memory," 2025, arXiv:2501.13956.

[33] W. Xu, Z. Liang, K. Mei, H. Gao, J. Tan, and Y. Zhang, "A-MEM: Agentic memory for LLM agents," 2025, arXiv:2502.12110.

[34] "MemGuard: Preventing memory contamination in long-term memory-augmented large language models," 2026, arXiv:2605.28009.

[35] "Organizational memory for agentic business process execution," 2026, arXiv:2607.03228.

[36] A. Maharana, D.-H. Lee, S. Tulyakov, M. Bansal, F. Barbieri, and Y. Fang, "Evaluating very long-term conversational memory of LLM agents," in *Proc. Annu. Meeting Assoc. Computational Linguistics (ACL)*, 2024, arXiv:2402.17753.

[37] D. Wu *et al.*, "LongMemEval: Benchmarking chat assistants on long-term interactive memory," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2025, arXiv:2410.10813.

[38] P. Christen, *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Berlin: Springer, 2012.

[39] B. A. Nosek, C. R. Ebersole, A. C. DeHaven, and D. T. Mellor, "The preregistration revolution," *Proc. Nat. Acad. Sci.*, vol. 115, no. 11, pp. 2600–2606, 2018.

[40] C. G. Northcutt, A. Athalye, and J. Mueller, "Pervasive label errors in test sets destabilize machine learning benchmarks," in *NeurIPS Datasets and Benchmarks Track*, 2021, arXiv:2103.14749.

[41] T. Yu *et al.*, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task," in *Proc. EMNLP*, 2018, arXiv:1809.08887.

[42] S. Kadavath *et al.*, "Language models (mostly) know what they know," 2022, arXiv:2207.05221.

[43] Z. Ji *et al.*, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, 2023, arXiv:2202.03629.