Think Fast, Talk Smart: Partitioning Deterministic and Neural Computation for Structured Health Text Generation

Kai-Chen Cheng [1] **Haejun Han** [1] **David Q. Sun** [1]

Abstract

Large language models (LLMs) are increasingly being used to generate health text from structured records such as wearable time series, biomarkers, vitals, and care-management logs. For recurring health outputs, fluency is not enough: systems must remain faithful to source data, ground explanatory claims in available evidence, follow stated policies, emit machine-readable outputs, and run cheaply enough for repeated use. We ask which responsibilities in structured health generation should be deterministic computation rather than runtime LLM prompting. We introduce Think Fast, Talk Smart (TFTS), a sleep-health insight pipeline in which deterministic code performs recurring analysis before one bounded LLM writer call. Across 280 user-nights and six models, TFTS achieves lower numeric error, lower instruction-compliance error, and lower end-to-end cost than structured zero-shot and few-shot one-call baselines. Layer replacement reveals contract-specific failures: LLM comparison raises numeric error, LLM ranking degrades policy selection, LLM attribution increases unsupported causal language, and an LLM-generated writer interface reintroduces errors even after upstream facts are deterministic. The results support a broader design rule: let code own recurring analysis, and let LLMs express verified facts within bounded interfaces.

1. Introduction

Many health and medical applications ask LLMs to turn structured records into user-facing text: wearable coaching summaries, lab results interpretations, medication guidance, vitals monitoring, and care-management reports (Thirunavukarasu et al., 2023; Singhal et al., 2023). These applications are promising precisely because language models can make technical records understandable. They are also risky because fluent health text must remain faithful to the source record, consistent with task instructions, and explicit about what evidence supports each claim (Ji et al., 2023; Van Veen et al., 2024). Recent wearable and personal-health systems show the promise of LLMs in this space (Kim et al., 2024; Khasentino et al., 2025; Merrill et al., 2026); they also sharpen a practical question: which parts of a recurring health-generation task should be learned generation, and which should be verified computation?

Prompting research offers several ways to improve model reasoning. Chain-of-thought elicits intermediate reasoning steps (Wei et al., 2022); iterative refinement trades more calls for improved outputs (Madaan et al., 2023); program-aided methods externalize computation (Chen et al., 2023; Gao et al., 2023); retrieval and tool-use systems augment LLMs at runtime (Lewis et al., 2020; Yao et al., 2023); and prompt-programming frameworks such as DSPy optimize LLM pipelines (Khattab et al., 2024). These methods improve what happens inside or around an LLM call. Our work asks a different systems question: when should a recurring health task be implemented outside the LLM call? For structured health workloads, some decisions recur with little change across inputs, such as comparing a measurement to a baseline, choosing what to surface, and deciding which explanatory claims are supported by logged evidence. When such steps are stable and verifiable, runtime prompting may be the wrong abstraction for the analytical work.

We study this partitioning problem through Think Fast, Talk Smart (TFTS), a sleep-health insight pipeline. Sleep-health is a useful case study because wearable records are longitudinal and heterogeneous, while the final output must be brief, grounded, and usable by non-expert readers. TFTS implements stable analytical steps as deterministic layers and uses an LLM only as a bounded final writer over a compact evidence interface. We compare this design against one-call baselines, then replace one analytical layer at a time with an LLM-generated typed artifact. This layer-replacement protocol tests whether a model preserves the intended meaning of an intermediate analytical artifact beyond producing syntactically valid JSON.

[1]AI/ML@Eight Sleep. Correspondence to: Kai-Chen Cheng <kevin.cheng@eightsleep.com>.

Proceedings of the Workshop on Structured Data for Health at the 43rd International Conference on Machine Learning, Seoul, South Korea. Copyright 2026 by the author(s).

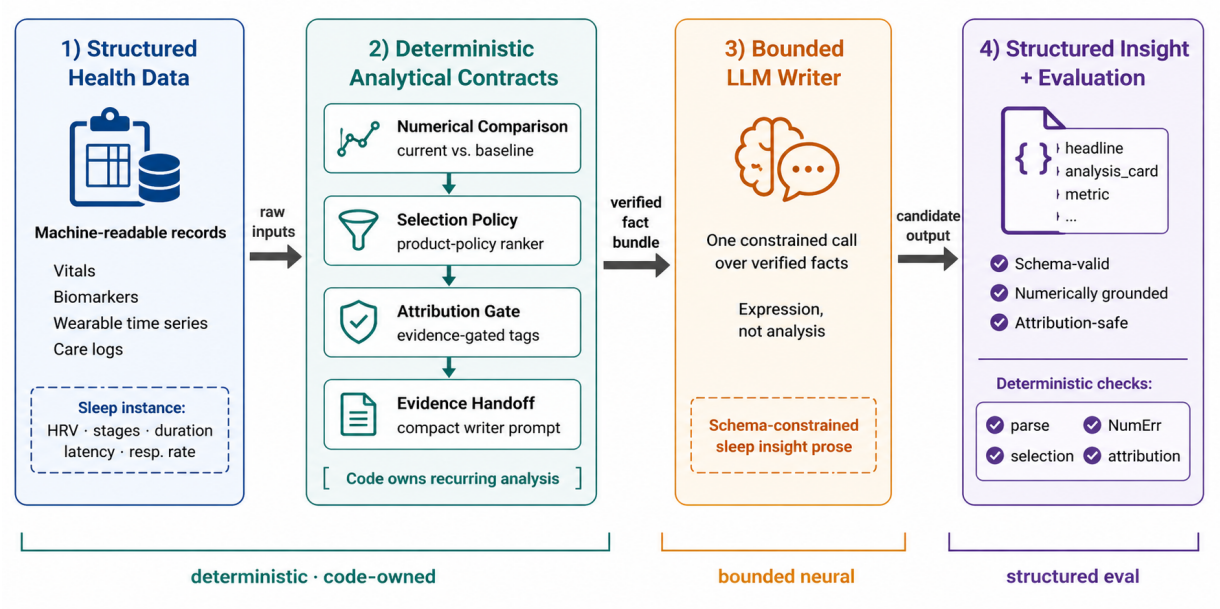

Figure 1. **The TFTS partitioning framework.** Recurring structured health-generation tasks are split into structured inputs, deterministic analytical modules, a bounded LLM writer, and deterministic evaluation of the structured output. The sleep-health callouts instantiate this partition in the evaluated pipeline.

Our contributions are threefold: we show that deterministic analytical layers plus a bounded writer achieve lower error and lower cost than strong prompting baselines on recurring health-insight generation tasks; introduce a layer-replacement protocol that isolates individual analytical responsibilities with typed LLM-generated intermediates; and identify the writer interface as a non-obvious failure point. Although our experiments use sleep-health data, the partitioning principle targets broader structured health settings where inputs are machine-readable, analytical rules recur, and intermediates can be checked.

2. Task and Pipeline

Figure 1 shows the four-stage partitioning pattern and the sleep-health instance we evaluate. For each user-night, the system ingests wearable-derived sleep, recovery, physiology, and behavior signals and returns a schema-constrained insight with a title, core insight, improvement suggestion, selected metric metadata, attribution tags, and chart data. The deterministic reference pipeline follows a classic data-to-text decomposition (Reiter, 2007; Wiseman et al., 2017): it formats data, computes numerical comparisons, ranks which metric should be surfaced, gates attribution by evidence, and compacts the selected facts into a deterministic prompt interface. The final LLM call then produces user-facing prose under a bounded schema, after which deterministic assembly normalizes and wraps the response.

TFTS decouples analytical reasoning from linguistic expression: deterministic code handles recurring, verifiable reasoning, while the LLM focuses on phrasing verified facts for the user. The experiment asks which recurring responsibilities can safely be moved from deterministic code into LLM-generated intermediates.

3. Baselines and Layer Replacement

One-call prompting baselines. We compare TFTS to two single-call alternatives. STRUCTURED ZERO-SHOT receives raw data, the output schema, and the same sleep insight generation rules in prompt form; it must analyze, select, attribute, and write in one call without demonstrations. STRUCTURED FEW-SHOT strengthens this baseline with structured metric tables, few-shot examples, explicit selection rules, and numeric-grounding instructions.

Layer-replacement protocol. For mechanistic analysis, we replace exactly one analytical layer with an LLM-generated typed artifact. All upstream reference layers remain deterministic; the generated artifact replaces the target layer's reference output. This protocol is stricter and fairer than removing a layer from the final writer prompt: the model is asked to perform one bounded intermediate task, not to recover missing context while writing the final JSON. The main conditions replaced COMPARISON, RANKER, ATTRIBUTION, and HANDOFF layers.

Data and models. The experiment uses 280 user-nights from 20 active users. We evaluate six models: GPT-5 nano,

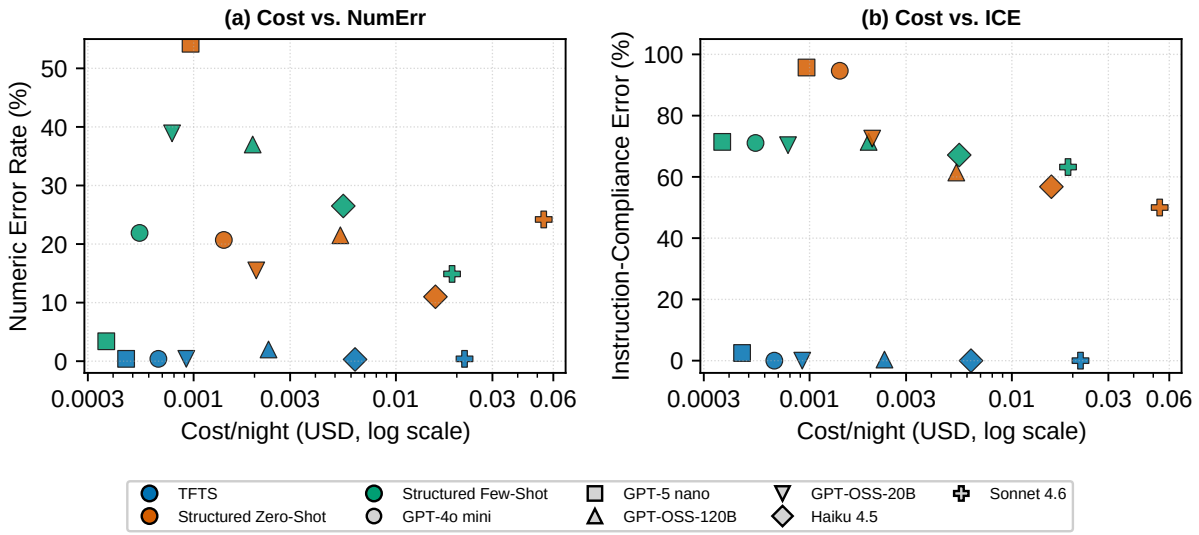

Figure 2. Baseline cost/error frontier across six models ($n = 280$ per cell). Both panels use end-to-end COST/NIGHT on a log scale. INSTRUCTION-COMPLIANCE ERROR is the union of SELERR and ATTRERR. TFTS remains lower on both error axes than the structured zero-shot and few-shot one-call baselines across the evaluated cost range.

GPT-4o mini, GPT-OSS-20B, GPT-OSS-120B, Haiku 4.5, and Sonnet 4.6. Replacement conditions use two LLM calls per night, one artifact call and one final writer call; TFTS and prompting baselines use one call. We therefore report end-to-end cost and latency per insight. d

4. Evaluation

All metrics are computed deterministically from saved per-night JSON traces. SCHEMAERR is the final JSON parse/schema error rate. NUMERR is the fraction of extracted numeric claims that are unsupported by, or inconsistent with, the source fact bank. Outputs with zero extracted numeric claims are excluded from NUMERR, so denominators vary across conditions. SELERR is the fraction of outputs whose selected metric contradicts the selection rule given to both the deterministic ranker and the LLM prompt. ATTRERR is the fraction of outputs that make an attribution not supported by the evidence-gated input.

5. Results

Error–cost frontier. Figure 2 shows the main comparison. Across all six models, TFTS occupies the low-cost, low-error region: average NUMERR is 0.7%, the worst model remains at 2.0%, and instruction-compliance error stays below 3%. The one-call baselines do not recover this frontier. STRUCTURED ZERO-SHOT often selects the wrong metric and makes unsupported numeric claims; STRUCTURED FEW-SHOT improves schema and attribution behavior, but aggregate SELERR remains 68.6% (see Table 1) and numeric errors remain high for several models. This should not be read as a negative result for prompting: prompting helps formatting and some safety language, but it does not reliably reproduce the recurring analytical decisions that drive the insight.

Layer replacements isolate failure modes. Table 1 complements the frontier by locating which analytical responsibility fails when delegated to an LLM. REPLACE COMPARISON delegates numerical observations and baseline comparisons to an LLM artifact; the aggregate NUMERR rises to 16.9%. REPLACE RANKER keeps numeric facts grounded but degrades adherence to selection rules, with an aggregate SELERR of 65.8%. REPLACE ATTRIBUTION preserves selection and numeric grounding but increases attribution errors, with aggregate ATTRERR of 24.5%. The pattern is more informative than a single end-to-end score: arithmetic, ranking policy, and evidence-gated attribution fail in different ways, and each is easier to verify as deterministic code than as prompted generation.

The writer interface matters. REPLACE HANDOFF tests a more subtle boundary. Upstream computation, ranking, and attribution remain deterministic, yet replacing the compact evidence-to-writer handoff raises aggregate NUMERR to 4.8% and ATTRERR to 10.7%. This suggests that the interface between verified facts and prose is part of the reliability mechanism, not just prompt formatting. A final writer can be useful, but the evidence packet that constrains what it may say should remain deterministic.

Representative examples clarify the metrics. Figure 3 shows representative examples for the four main error fami-

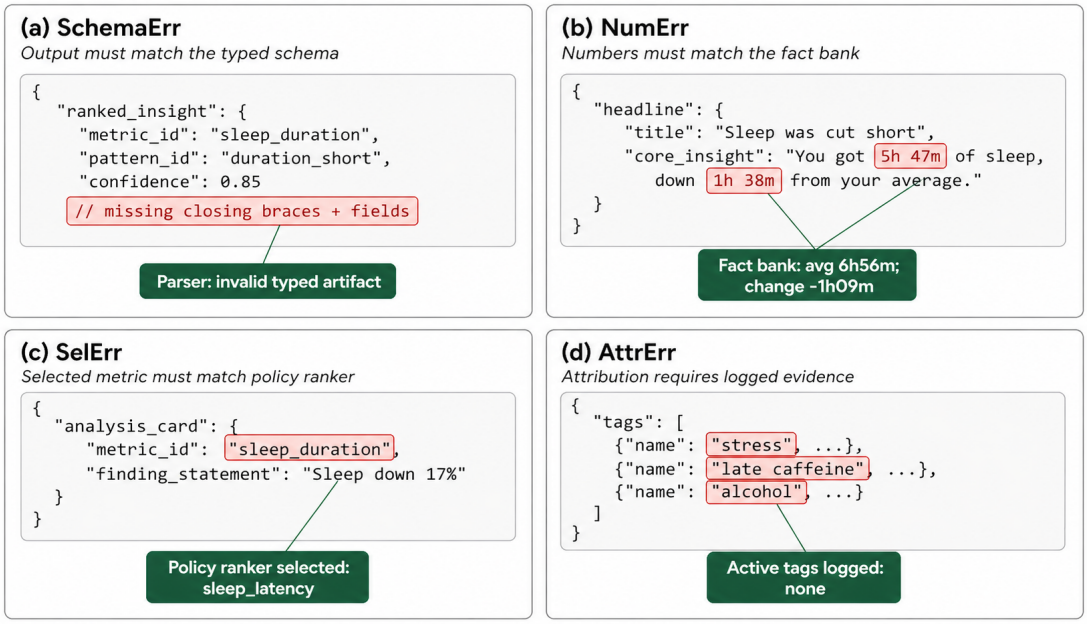

Figure 3. Annotated examples of error types. Red highlights mark the LLM output span that violates the target rule; green callouts show the deterministic reference used by the evaluator. SCHEMAERR captures parse/schema failures, NUMERR captures numeric hallucinations, SELERR captures selection-policy mismatch, and ATTRERR captures unsupported attribution.

lies. The examples are not used as evidence by themselves; they make the deterministic metrics in Table 1 interpretable. They also show why schema-valid output is not enough: a response can parse correctly while still using an unsupported number, selecting the wrong metric, or adding attribution with no evidence above threshold.

6. Discussion and Limitations

For recurring structured health outputs, deterministic code should own stable, verifiable responsibilities: arithmetic over records, selection policy, evidence-gated attribution, and the writer interface that controls which facts may enter prose. LLMs remain useful where their strengths match the task: converting verified facts into natural, empathetic, schema-constrained language. This framing complements RAG, tool use, and LLM-pipeline optimization (Lewis et al., 2020; Schick et al., 2023; Yao et al., 2023; Mialon et al., 2023): those methods can improve learned calls or retrieve better context, while our question is which repeated health-generation responsibilities should be removed from runtime prompting entirely. Another promising direction is build-time agentic AI, where LLM agents help humans construct and audit deterministic health-insight pipelines before deployment, closer to pipeline-compilation systems and wearable-health agent case studies than to the runtime prompting baselines evaluated here (Khattab et al., 2024; Merrill et al., 2026).

The partitioning principle should transfer most directly to domains with structured inputs, stable analytical rules, repeated deployment, and verifiable intermediates. Lab result explanations, vitals monitoring summaries, medication-adherence coaching, and chronic-disease reports often share this shape. We have not validated those domains here, and tasks requiring open-ended clinical reasoning, diagnosis, or treatment planning may need a different partition.

This study evaluates preservation of a deterministic reference system, not clinical optimality. SELERR is disagreement with an application selection rule rather than ground-truth health importance, and ATTRERR is attribution-policy compliance rather than biological causality. Human evaluation of helpfulness and perceived personalization is complementary future work.

7. Conclusion

Structured health generation requires both reliable analytical computation and natural user-facing expression. In our sleep-health case study, one-call prompting cannot match a deterministic analytical pipeline plus bounded writer, even with few-shot examples. Layer replacement shows why: numerical comparison, ranking policy, attribution gates, and writer handoff each encode responsibilities that schema-valid LLM artifacts often fail to preserve. TFTS supports a practical design rule for structured health systems with recurring analytical work: let code think fast over stable facts and policies, then let the LLM talk smart within verified bounds.

References

Chen, W., Ma, X., Wang, X., and Cohen, W. W. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023.

Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G. PAL: Program-aided language models. *International Conference on Machine Learning*, pp. 10764–10799, 2023.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y., Madotto, A., and Fung, P. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023. doi: 10.1145/3571730.

Khasentino, J., Belyaeva, A., Liu, X., Yang, Z., Furlotte, N. A., et al. A personal health large language model for sleep and fitness coaching. *Nature Medicine*, 31(10): 3394–3403, 2025. doi: 10.1038/s41591-025-03888-0.

Khattab, O., Singhvi, A., Maheshwari, P., Zhang, Z., Santhanam, K., et al. DSPy: Compiling declarative language model calls into self-improving pipelines. In *International Conference on Learning Representations*, 2024.

Kim, Y., Xu, X., McDuff, D., Breazeal, C., and Park, H. W. Health-LLM: Large language models for health prediction via wearable sensor data. In *Proceedings of the Fifth Conference on Health, Inference, and Learning*, volume 248 of *Proceedings of Machine Learning Research*, pp. 522–539. PMLR, 2024. URL https://proceedings.mlr.press/v248/kim24b.html.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2023.

Merrill, M. A., Paruchuri, A., Rezaei, N., Kovacs, G., Perez, J., et al. Transforming wearable data into personal health insights using large language model agents. *Nature Communications*, 17:1143, 2026. doi: 10.1038/s41467-025-67922-y.

Mialon, G., Dessì, R., Lomeli, M., Nalmpantis, C., Pasunuru, R., Raileanu, R., Rozière, B., Schick, T., Dwivedi-Yu, J., Celikyilmaz, A., et al. Augmented language models: a survey. *Transactions on Machine Learning Research*, 2023.

Reiter, E. An architecture for data-to-text systems. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*, pp. 97–104. Association for Computational Linguistics, 2007.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36, 2023.

Singhal, K., Tu, T., Gottweis, J., Sayres, R., Wulczyn, E., Hou, L., Clark, K., Pfohl, S., Cole-Lewis, H., Neal, D., et al. Towards expert-level medical question answering with large language models. *Nature Medicine*, 29:2899–2910, 2023. doi: 10.1038/s41591-023-02528-7.

Thirunavukarasu, A. J., Ting, D. S. J., Elangovan, K., Gutierrez, L., Tan, T. F., and Ting, D. S. W. Large language models in medicine. *Nature medicine*, 29(8):1930–1940, 2023.

Van Veen, D., Van Uden, C., Blankemeier, L., Delbrouck, J.-B., Aali, A., et al. Adapted large language models can outperform medical experts in clinical text summarization. *Nature Medicine*, 30:1134–1142, 2024. doi: 10.1038/s41591-024-02855-5.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837, 2022.

Wiseman, S., Shieber, S. M., and Rush, A. M. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2253–2263. Association for Computational Linguistics, 2017.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Appendix

A. Full Results

Table 1. Full condition-by-model results. Values are percentages except n, COST/NIGHT, and latency. SCHEMAERR is the final parse/schema failure rate; SELERR is disagreement with the selection rule; ATTRERR is unsupported attribution. Replacement conditions include both the artifact call and final writer call in COST/NIGHT.

Condition	Model	n	SchemaErr	NumErr	SelErr	AttrErr	Cost/night	Lat.
TFTS	GPT-5 nano	280	0.0	0.4	0.0	2.5	$0.0005	4.4s
	GPT-4o mini	280	0.0	0.4	0.0	0.0	$0.0007	8.1s
	GPT-OSS-20B	280	0.0	0.4	0.0	0.0	$0.0009	2.6s
	GPT-OSS-120B	280	0.0	2.0	0.0	0.4	$0.0023	4.5s
	Haiku 4.5	280	0.0	0.3	0.0	0.0	$0.0063	8.6s
	Sonnet 4.6	280	0.0	0.4	0.0	0.0	$0.0218	15.1s
Structured Zero-Shot	GPT-5 nano	280	0.0	54.2	95.0	18.6	$0.0010	4.4s
	GPT-4o mini	280	0.0	20.7	93.2	42.2	$0.0014	8.6s
	GPT-OSS-20B	280	0.4	15.5	71.8	2.5	$0.0020	4.8s
	GPT-OSS-120B	280	0.0	21.5	61.1	2.5	$0.0053	7.5s
	Haiku 4.5	280	20.7	11.0	77.5	0.4	$0.0157	12.0s
	Sonnet 4.6	280	32.5	24.2	82.5	1.1	$0.0538	16.8s
Structured Few-Shot	GPT-5 nano	280	0.0	3.4	71.1	3.6	$0.0004	3.8s
	GPT-4o mini	280	0.0	21.9	70.4	4.6	$0.0005	8.7s
	GPT-OSS-20B	280	0.0	38.9	68.9	2.1	$0.0008	2.4s
	GPT-OSS-120B	280	0.0	37.0	71.1	0.4	$0.0020	3.9s
	Haiku 4.5	280	0.0	26.5	67.1	0.7	$0.0055	6.3s
	Sonnet 4.6	280	0.0	14.9	63.2	0.7	$0.0189	9.9s
Replace Reference Report	GPT-5 nano	280	0.0	0.6	8.9	11.8	$0.0009	8.2s
	GPT-4o mini	280	0.0	0.4	4.6	11.4	$0.0012	14.5s
	GPT-OSS-20B	280	0.4	1.8	1.4	10.0	$0.0024	7.5s
	GPT-OSS-120B	280	0.0	3.4	0.7	10.7	$0.0060	12.9s
	Haiku 4.5	280	0.7	3.0	2.1	8.9	$0.0115	15.4s
	Sonnet 4.6	280	0.0	1.6	3.2	10.0	$0.0381	24.0s
Replace Comparison	GPT-5 nano	280	0.0	21.1	76.8	1.8	$0.0014	14.8s
	GPT-4o mini	280	0.0	20.9	68.2	5.7	$0.0021	31.0s
	GPT-OSS-20B	280	1.4	6.6	25.4	8.6	$0.0031	9.3s
	GPT-OSS-120B	280	0.0	8.8	15.0	10.0	$0.0077	16.3s
	Haiku 4.5	280	0.0	26.7	34.6	8.6	$0.0225	23.7s
	Sonnet 4.6	280	0.0	17.6	27.5	9.3	$0.0632	34.6s
Replace Ranker	GPT-5 nano	280	0.7	0.3	63.9	8.9	$0.0014	6.9s
	GPT-4o mini	280	0.0	0.0	71.8	9.6	$0.0022	13.5s
	GPT-OSS-20B	280	0.4	0.5	62.5	6.8	$0.0025	6.0s
	GPT-OSS-120B	280	0.0	3.1	67.1	10.7	$0.0070	10.2s
	Haiku 4.5	280	5.3	2.2	62.9	7.5	$0.0160	11.9s
	Sonnet 4.6	280	17.8	1.0	66.4	8.6	$0.0499	17.6s
Replace Attribution	GPT-5 nano	280	0.0	0.3	9.6	10.3	$0.0015	9.0s
	GPT-4o mini	280	0.0	0.4	2.9	21.1	$0.0020	12.7s
	GPT-OSS-20B	280	0.0	0.8	0.0	17.1	$0.0023	5.5s
	GPT-OSS-120B	280	0.7	5.0	0.7	22.1	$0.0058	8.0s
	Haiku 4.5	280	0.4	1.1	1.1	32.8	$0.0198	22.9s
	Sonnet 4.6	280	0.4	1.3	0.4	43.6	$0.0610	28.7s
Replace Handoff	GPT-5 nano	280	0.0	7.9	2.9	16.8	$0.0010	8.9s
	GPT-4o mini	280	0.0	1.5	2.9	11.4	$0.0014	14.2s
	GPT-OSS-20B	280	0.0	1.9	0.4	10.0	$0.0020	5.6s
	GPT-OSS-120B	280	0.0	3.8	0.4	9.3	$0.0046	8.1s
	Haiku 4.5	280	0.4	8.1	0.4	8.6	$0.0133	17.9s
	Sonnet 4.6	280	0.0	5.8	0.4	7.8	$0.0423	27.0s

B. Example Pipeline Trace

Input record and deterministic facts


```
{
"date": "2026-02-23",
"history_window": "2026-02-09..2026-02-23",
"sleep": {
"score": 84,
"duration": "7h 50m",
"deep": "1h 26m", "rem": "1h 40m",
"light": "4h 44m"
},
"vitals": {
"hrv": "34.2 ms",
"heart_rate": "59.2 bpm",
"resp_rate": "15.8 brpm"
},
"snore_percent": "6.0%",
"tag_candidates": [
"Alcohol", "Stress", "Sick", "Fever"
],
"selected_metric": "hrv",
"comparison": "34.2 vs 41.3 ms (-17%)"
}
```


Bounded writer output


```
{
"headline": {
"title": "Your recovery looks different",
"core_insight": "Your HRV dropped to
34 ms, down 17% from your baseline.",
"how_to_improve": "Do a 10-minute
progressive muscle relaxation tonight..."
},
"analysis_card": {
"metric_id": "hrv",
"finding_statement": "HRV down 17%",
"tags": [
{"name": "Alcohol", ...},
{"name": "Stress", ...},
{"name": "Sick", ...},
{"name": "Fever", ...}
]
}
}
```


Deterministic handoff. The pipeline selects `hrv` as the surfaced metric and gives the writer the fixed comparison "34.2 vs 41.3 ms (-17%)." The writer may copy the pre-generated attribution tags shown in blue, but it may not add new tags or recalculate numbers.

Figure 4. Representative TFTS trace for one user-night (Sonnet 4.6, 2026-02-23). The figure shows the compacted structured input, deterministic selected facts, and final schema-constrained output. Blue spans mark tag fields copied through the bounded writer interface.