DRiPP: Discrete Representation Protocols for VRAM-Constrained Local LLM Pipelines

Sachin Mahesh
University of California, Berkeley
sachinmahesh@berkeley.edu

Abstract

Multi-stage LLM pipelines on consumer hardware face a fundamental tension: passing verbose intermediate reasoning states between stages consumes tokens, increases latency, and can exceed VRAM budgets when stages must be temporally scheduled (loaded and unloaded). We study the *inter-stage communication protocol*—the format in which one model's output is passed to the next—as a first-class design variable.

We implement DRiPP (**D**iscrete **R**epresentation **i**nter-stage **P**ipeline **P**rotocols), a two-stage temporally-scheduled pipeline on an RTX 5060 Ti (16 GB), and benchmark six discrete communication protocols—full natural language, truncation, prose summary, structured JSON, symbolic triples, and a codebook—plus two causal ablations (key shuffling, value randomization) across three model families (Qwen3-4B-Instruct-2507, Llama-3.2-3B-Instruct, Phi-4-mini-Instruct) on 30 passages (SQuAD + custom), measuring latency, intermediate token count, peak VRAM, GB-seconds, token F1, recall, exact match, and numeric hallucination rate.

Prose summary achieves the highest factual utility per intermediate token across all model configurations, reducing token count by 44–56% and GB-seconds by 37–49% vs. full natural language, with no significant F1 loss on Qwen3-4B ($p = 0.74$) or Llama-3.2-3B ($p = 0.35$). On Phi-4-mini, full NL achieves statistically higher raw F1 ($p < 0.001$), but prose summary remains the Pareto-optimal choice under any token budget. Symbolic and codebook formats degrade factual recall and introduce hallucinations (22.7% and 8.7% numeric hallucination rates on Qwen3; 14.7% and 0% on Llama) while offering no token-efficiency advantage. A causal ablation—randomizing symbolic values while preserving format—drops F1 from 0.453 to 0.021, confirming that content, not format scaffolding, carries factual signal. In a heterogeneous cross-family pipeline (Qwen3-4B synthesizer → Llama-3.2-3B refiner), prose-summary inter-stage messages achieve $F1 = 0.705$, outperforming same-model Llama-only pipelines ($F1 = 0.551$) by 28%.

1 Introduction

Running capable AI pipelines locally—without cloud APIs—requires fitting multiple LLMs into a fixed VRAM budget. A common solution is *temporal model scheduling*: stages are loaded just-in-time, executed, and immediately unloaded, so that the memory footprint at any moment equals only the currently active model rather than the sum of all models [2]. This strategy works, but introduces a design question that has not been studied: what format should be used to pass information from one stage to the next?

We call this the **inter-stage communication protocol**. In practice, most multi-stage systems pass the previous stage's raw text output verbatim—an implicit choice of *full natural language*. But full NL may be unnecessarily verbose: a 600-token synthesizer output fed into a 3B-parameter refiner occupies a significant fraction of the refiner's context window, increases prefill time, and grows GB-seconds (VRAM × wall-time), a resource metric that matters when VRAM is the binding constraint. Alternative formats—JSON, symbolic notation, compressed summaries—promise shorter messages but may degrade factual fidelity or introduce hallucinations.

The prior art on this question is sparse. CIPHER [15] passes embedding-level representations between agents, bypassing token sampling entirely, but requires all models to share the same architecture and vocabulary space—inapplicable to heterogeneous local pipelines where stages may be different model families or quantization levels. LLMLingua [7] and LLMLingua-2 [9] compress *input* prompts to a single model; compressing *intermediate reasoning states transferred between two separate models* is a different problem formulation with different failure modes. FrugalGPT [3] cascades models to reduce cost but

treats inter-stage messages as fixed natural language. To our knowledge, no prior work systematically compares discrete text-format communication protocols for inter-stage transfer, with causal ablations isolating whether format structure or content carries factual signal.

This paper's contribution is an initial controlled empirical study of the question: *which inter-stage communication protocol best preserves downstream factual utility under a fixed token budget?* Concretely we contribute:

1. A taxonomy of six inter-stage communication protocols spanning the verbosity–structure space.
2. A controlled benchmark (30 passages $\times$ 5 Q&A pairs, SQuAD + custom) with eight metrics covering latency, VRAM efficiency, factual recall, and hallucination.
3. Evaluation across three model families and a heterogeneous cross-family pipeline reflecting the actual DRiPP temporal architecture.
4. Two causal ablations—JSON key shuffling and symbolic value randomization—that identify what carries factual signal in structured formats.

2 Related Work

Multi-model pipelines and cascades. FrugalGPT [3] learns which model combination minimizes cost for a target accuracy, but treats each model's output as opaque natural language. Speculative decoding [8] uses a draft model to accelerate a larger target model but operates within a single generation pass, not across separately loaded models. Mixture-of-Experts [11, 6] routes tokens to specialist subnetworks within a single model and requires all experts simultaneously in VRAM. DRiPP's temporal scheduling—loading and unloading models per stage—is closest to FlexGen [12] in motivation but operates at the pipeline level rather than the operator level.

Prompt and context compression. LLMLingua [7] uses a small auxiliary LM to drop low-perplexity tokens from *input* prompts, achieving up to 20$\times$ compression. LLMLingua-2 [9] extends this with data distillation. RECOMP [14] applies abstractive or extractive compression to RAG-retrieved context. All of these compress *inputs* to a single model. Our target—the output of stage k consumed as input to stage k+1 across different models—is a different problem: the intermediate state must faithfully encode reasoning produced by one model for consumption by another that shares no context, KV cache, or activation space.

Embedding-level inter-agent communication. CIPHER [15] bypasses token sampling by passing probability-weighted vocabulary embeddings between agents in a debate setup, outperforming natural-language debate by 0.5–5.0% on reasoning tasks. These embedding-level approaches require homogeneous model architectures sharing the same embedding space. They cannot be applied when Stage A and Stage B are different model families loaded sequentially on one GPU—which is precisely the case for temporally scheduled local pipelines. Our work occupies the complementary regime: text-format protocols that work across any models.

Structured outputs from LLMs. Constrained decoding [4] guarantees syntactically valid JSON or schema-conformant output but does not guarantee semantic correctness. Our JSON protocol relies on prompted (unconstrained) generation; the numeric hallucination rates we observe arise precisely from semantic errors despite syntactically plausible outputs.

3 The DRiPP Pipeline Architecture

3.1 Temporal Model Scheduling

DRiPP implements a two-stage inference pipeline in which no two models occupy GPU memory simultaneously:

1. **Stage A (Synthesizer).** A model M_A is loaded, given a system context and task description, and generates an intermediate state S in protocol format $\mathcal{P}$. M_A is then unloaded and its VRAM reclaimed.
2. **Stage B (Refiner).** A model M_B (which may differ from M_A in family, size, or quantization) is loaded and receives only S—not the original context—and produces the final output.

The key constraint: Stage B has *no access* to the original source beyond what S encodes. This makes the information content of S the binding constraint on downstream factual quality, and makes the choice of protocol $\mathcal{P}$ a first-class engineering decision.

Peak VRAM at any moment is $\max(\text{VRAM}(M_A), \text{VRAM}(M_B))$ rather than

VRAM(M_A) + VRAM(M_B), enabling configurations that would otherwise exceed the hardware budget.

3.2 Protocol Taxonomy

We define six protocols spanning a verbosity–structure space:

Full-NL. Stage A generates an unconstrained natural-language summary of all key facts. Maximum token budget ($\leq$ 700 new tokens). Serves as the upper-bound reference.

Truncated. Same prompt as Full-NL, but the output is hard-truncated to 150 tokens post-generation. Tests information loss from position-based, non-semantic compression.

Summary. Stage A is prompted to write a *3-sentence* factual summary prioritizing specific numbers, dates, and named entities. Semantic compression by instruction.

JSON. Stage A is prompted to extract all key facts as a flat JSON object (descriptive string keys, exact values). No surrounding text.

Symbolic. Stage A encodes facts as `ENTITY[value|attribute]` triples, one per line. Inspired by knowledge-graph notation.

Codebook. Stage A defines a short-code dictionary (`K1=value|K2=value|...`) then writes a compressed message using only the codes. Maximum compression, maximum parsing overhead.

3.3 Causal Ablations

Two ablation conditions test *what* in a structured format carries factual signal:

JSON-Shuffled. The JSON output from JSON is parsed and its keys are randomly permuted before being passed to Stage B. Tests whether key *order* carries signal.

Symbolic-Rand. Values inside each `[...]` triple from SYMBOLIC are replaced with random alphanumeric strings of equal character length. Tests whether the symbolic *format* alone scaffolds retrieval, independent of content.

4 Experimental Setup

4.1 Hardware and Models

All experiments run on a single NVIDIA GeForce RTX 5060 Ti (16 GB GDDR7, CUDA 13.0) under Ubuntu 24.04 LTS. No model quantization is applied; all models run in BF16. Models are loaded and unloaded between stages using `transformers` 5.3.0 with `device_map=cuda` and `torch.cuda.empty_cache()` between stages. No Ollama or external serving infrastructure is used.

Three model families are evaluated:

- **Qwen3-4B-Instruct-2507** [13] (4B parameters, 8.07 GB VRAM). Thinking mode disabled via `enable_thinking=False`.
- **Llama-3.2-3B-Instruct** [5] (3B parameters, $\sim$6 GB VRAM).
- **Phi-4-mini-Instruct** [1] (3.8B parameters, $\sim$8 GB VRAM).

A cross-family pipeline (Qwen3-4B as Stage A, Llama-3.2-3B as Stage B) evaluates the heterogeneous case.

4.2 Dataset

The benchmark consists of 30 factual passages with 5 questions each (150 Q&A pairs total):

- **10 custom passages** covering diverse factual domains (space exploration, particle physics, computing history, biology, engineering, oceanography). Questions target specific numerical values, dates, and named entities.
- **20 SQuAD passages** [10] sampled from the SQuAD v1.1 validation split (Wikipedia passages, $\leq$400 words, $\geq$5 questions per passage with short extractive answers).

All passages are presented to Stage A; Stage B receives *only* the intermediate state S—not the original passage.

4.3 Metrics

For each (passage, protocol, model) triple we measure:

- **Intermediate token count**: tokenized length of S.
- **Latency**: wall-clock time for Stage A + Stage B generation.

- **Peak VRAM**: PyTorch peak memory per stage (`max_memory_allocated()`).
- **GB-seconds**: $\text{VRAM}_A \times t_A + \text{VRAM}_B \times t_B$. Captures the VRAM-time integral—the resource cost of inference.
- **Token F1**: token-overlap F1 between Stage B prediction and ground truth answer, averaged over 5 questions.
- **Recall**: fraction of ground-truth tokens present in prediction.
- **Exact Match (EM)**: ground-truth string is a substring of the prediction.
- **Numeric numeric hallucination rate**: fraction of questions where Stage B's response contains a specific numerical value not present in the original source passage.

Statistical significance is assessed via paired t-tests (per-passage, two-tailed) with Prose Summary as the reference condition. We report $p < 0.05$ as significant; no multiple-comparison correction is applied given the exploratory nature of the comparison.

4.4 Generation Details

All generation uses greedy decoding (temperature = 0) for reproducibility. Stage A maximum new tokens: 700 for FULL-NL, 400 for all others. Stage B maximum new tokens: 300. System prompts are identical across protocols within each stage (Appendix A).

4.5 Implementation Details

Stage B question delivery. All five questions for a passage are presented to Stage B in a single prompt, concatenated as a numbered list. Stage B generates answers for all five in one forward pass, with answers parsed by line. The same intermediate state S is reused for all five questions; Stage B is not given the passage or Stage A's instructions.

Token counting. Intermediate token counts use Stage A's tokenizer throughout, including in the cross-family pipeline. In the Qwen3-4B → Llama-3.2-3B pipeline, Stage A's (Qwen3) tokenizer may produce a slightly different token count than Stage B's (Llama) tokenizer for the same text. We report Stage A token counts for consistency; actual Stage B prefill cost may vary by ±5–10% depending on the receiving model's vocabulary.

Protocol transparency. Stage B does *not* receive the protocol name or any format-parsing instructions. Its system prompt is identical across all protocols: "Answer each question based ONLY on the intermediate state above." This is the harder, more realistic test: a downstream model receives structured or symbolic output without being told how to interpret it. Protocols that require active parsing (Codebook, Symbolic) are therefore at a genuine disadvantage relative to natural-language formats.

5 Results

5.1 Single-Model Protocol Comparison

Table 1 shows results across all three model families. The primary findings are consistent across models:

Prose Summary is token-optimal; Full-NL is not significantly better. On Qwen3-4B (30 passages), Full-NL achieves the highest raw F1 (0.666) but is not significantly better than Prose Summary (F1 = 0.655; $p = 0.74$). Prose Summary uses 56% fewer tokens (138 vs. 315) and 48% fewer GB-seconds (42.2 vs. 81.9). Prose Summary is significantly better than all other protocols: vs. Truncated ($p = 0.001$), vs. Symbolic ($p = 0.001$), vs. Codebook ($p < 0.001$). The same pattern holds on Llama-3.2-3B: Prose Summary achieves F1 = 0.551 (best of all protocols, $p < 0.001$ vs. Truncated, JSON, Symbolic, Codebook), using 48% fewer tokens and 41% fewer GB-seconds than Full-NL.

Phi-4-mini: Full-NL is the raw-F1 winner, but not token-efficient. On Phi-4-mini, Full-NL (F1 = 0.775) is statistically significantly better than Prose Summary (F1 = 0.646; $p < 0.001$, $\Delta = +0.129$). This reversal holds across all Phi-4 protocol comparisons: JSON and Truncated also approach Full-NL performance ($p > 0.1$). However, Prose Summary remains the most *token-efficient* protocol (F1/token = 0.00653 vs. 0.00439 for Full-NL): among the evaluated protocols, prose summary delivers the most factual utility per token on Phi-4 as well. This likely reflects Phi-4-mini's training distribution favouring longer, denser context in the 177-token Full-NL regime.

Symbolic and Codebook formats are harmful. Symbolic notation produces 22.7% numeric hallucination rate on Qwen3 (30p), 14.7% on Llama, and 12.7% on Phi-4, while delivering the lowest F1 of

Table 1: Protocol comparison across model families on the DRiPP benchmark (30 passages, 5 Q&A pairs each). Bold = best non-ablation F1 per model. All experiments on RTX 5060 Ti (16 GB VRAM), BF16 weights.

Model	Protocol	Tokens	Lat.(s)	GB·s	F1	Recall	EM	Num.Halluc.
Qwen3-4B-Inst-2507 (30p)	Full NL	315	10.02	81.9	**0.666**	0.874	0.773	0.000
	Truncated NL	150	8.98	73.2	0.507	0.664	0.613	0.000
	Prose Summary	138	5.19	42.2	0.655	0.792	0.673	0.007
	Struct. JSON	251	8.01	65.3	0.598	0.708	0.660	0.120
	Symbolic	227	7.30	59.5	0.453	0.507	0.427	0.227
	Codebook	304	9.43	77.0	0.428	0.547	0.487	0.087
	JSON shuffled†	253	8.20	66.9	0.568	0.721	0.653	0.107
	Symbolic rand†	457	6.53	53.3	0.021	0.038	0.033	0.007
Llama-3.2-3B-Inst	Full NL	228	5.14	33.5	0.495	0.550	0.473	0.027
	Truncated NL	147	4.98	32.4	0.336	0.377	0.313	0.013
	Prose Summary	118	3.06	19.9	**0.551**	0.621	0.553	0.007
	Struct. JSON	213	4.77	31.1	0.384	0.405	0.367	0.127
	Symbolic	216	4.88	31.8	0.322	0.316	0.253	0.147
	Codebook	368	7.86	51.3	0.088	0.097	0.093	0.000
	JSON shuffled†	213	4.76	31.0	0.422	0.447	0.387	0.113
	Symbolic rand†	415	4.64	30.3	0.049	0.052	0.040	0.007
Phi-4-mini-Inst	Full NL	177	4.99	38.7	**0.775**	0.893	0.827	0.000
	Truncated NL	140	4.94	38.3	0.697	0.796	0.727	0.000
	Prose Summary	99	3.15	24.4	0.646	0.715	0.620	0.007
	Struct. JSON	198	5.40	41.9	0.694	0.757	0.660	0.047
	Symbolic	157	4.49	34.8	0.587	0.615	0.547	0.127
	Codebook	247	6.49	50.5	0.541	0.585	0.520	0.000

† Causal ablation conditions.

any non-truncation protocol. Codebook collapses to F1 = 0.088 on Llama ($p < 0.0001$ vs. Prose Summary) and produces the highest GB-seconds in all configurations—worst on both quality and efficiency dimensions. The GB-seconds penalty is explained by Stage A token output: on Llama, the codebook format generates 368 tokens on average, *60% more* than Full-NL (229 tokens). The dictionary preamble (`K1=value|K2=value|...`) is itself verbose, and the model's "compressed" message using K-codes provides little actual compression. Stage B generates fewer tokens in response (33 vs. 38 for Prose Summary) but mostly "NOT FOUND"—the parsing overhead falls entirely on Stage A's generation cost, not Stage B's.

Truncation is a poor compression strategy. Hard-truncating Full-NL to 150 tokens loses 24% F1 vs. Full-NL on Qwen3 (0.507 vs. 0.666) and 39% vs. Prose Summary on Llama (0.336 vs. 0.551), despite a similar token budget to Prose Summary. Positional compression discards the most factually dense content at the end of Stage A's generation.

5.2 Causal Ablations

JSON key order is irrelevant; content is causal. Shuffling JSON keys changes F1 by +0.021 on Qwen3 and +0.038 on Llama (not significant, $p > 0.3$). Semantic key labels matter; their order does not. The slight positive delta on Llama may reflect positional reading bias in smaller models.

Symbolic format carries negligible signal without correct content. Replacing symbolic values with random equal-length strings collapses F1 to 0.021 on Qwen3 and 0.049 on Llama ($p < 0.001$ vs. Symbolic in both), with hallucination dropping to near-zero (0.007 and 0.007 respectively). Stage B predominantly outputs "NOT FOUND" when values are randomized, confirming the model correctly identifies the intermediate state as uninterpretable. The residual F1 of 0.021 reflects occasional lucky token overlaps on short answers, not genuine retrieval. The implication: symbolic notation's elevated numeric hallucination rate (22.7% on Qwen3) arises from Stage B *attempting* to decode plausible-looking but inaccurate triples, not from format scaffolding providing useful structure.

5.3 Cross-Family Heterogeneous Pipeline

Table 2 shows results for the Qwen3-4B → Llama-3.2-3B pipeline—Stage A and Stage B are loaded and unloaded sequentially.

Table 2: Cross-family pipeline: Stage A = Qwen3-4B-Instruct-2507 (synthesizer), Stage B = Llama-3.2-3B-Instruct (refiner). Models are loaded and unloaded sequentially (DRiPP temporal scheduling). 30 passages × 5 Q&A pairs, RTX 5060 Ti.

Protocol	Tokens	Lat.(s)	GB·s	F1	Recall	EM	Num.Halluc.
Full NL	315	9.25	81.1	**0.736**	0.848	0.780	0.000
Prose Summary	138	4.52	41.8	0.705	0.773	0.680	0.000
Struct. JSON	251	7.46	65.9	0.636	0.701	0.647	0.120
Symbolic	241	7.15	63.0	0.526	0.551	0.467	0.213

Prose Summary achieves F1 = 0.705 and Full-NL achieves F1 = 0.736, both *exceeding* same-model Llama-3.2-3B (F1 = 0.495–0.551 depending on protocol). This demonstrates two things: (1) temporal scheduling with heterogeneous models is viable—factual utility is preserved across the load/unload boundary; and (2) a stronger synthesizer improves the downstream refiner's output even when the refiner is weaker. The 28% F1 improvement over same-model Llama (0.705 vs. 0.551) directly justifies DRiPP's staged architecture.

Prose Summary remains the most token-efficient protocol in the cross-family setting (F1/token = 0.00509 vs. 0.00234 for Full-NL), using 56% fewer intermediate tokens (138 vs. 315) and 48% fewer GB-seconds (41.8 vs. 81.1).

6 Discussion

6.1 Why Prose Summary Wins on Token Efficiency

Prose summary exploits the fact that instruction-tuned LLMs are trained to produce coherent, prioritized natural-language text in response to summarization prompts. The 3-sentence constraint forces *semantic* compression: the model must rank facts by importance and discard peripheral detail, rather than truncating by position (TRUNCATED) or expanding a fixed-schema structure (JSON). The resulting text sits squarely in Stage B's training distribution, requiring no format-parsing overhead.

On Qwen3-4B and Llama-3.2-3B, the raw-F1 gap between Full-NL and Prose Summary is statistically indistinguishable ($p > 0.3$ in both cases), meaning prose summary incurs *no* factual utility penalty while saving 44–56% of tokens. On Phi-4-mini, Full-NL achieves significantly higher raw F1 ($p < 0.001$), suggesting this model's training distribution favours longer, denser input context—but Prose Summary is still the token-efficient choice when context budget is the binding constraint.

In contrast, structured formats demand two things simultaneously: accurate fact extraction *and* format adherence. Small instruction-tuned models (3B–4B) frequently trade one for the other, producing well-formed JSON or symbolic triples with hallucinated values.

6.2 Implications for Pipeline Design

For practitioners building temporally-scheduled local LLM pipelines:

- Use **prose summarization** as the default inter-stage protocol. On most 3B–4B models it achieves Full-NL accuracy at 44–56% lower token cost with minimal hallucination.
- **Avoid symbolic and codebook formats** unless the synthesizer model is specifically fine-tuned for structured output. The hallucination cost outweighs any token savings.
- In **cross-family pipelines**, use the stronger model as Stage A (synthesizer): even if the refiner is weaker, it benefits from a higher-quality intermediate state.
- **Token efficiency** (F1/token) is a more useful design metric than raw F1 when operating under a context-window or latency budget.

6.3 Limitations

Dataset scale. Our benchmark uses 30 passages and extractive short-answer questions. Performance on longer, multi-hop reasoning tasks may differ—structured formats might fare better when the relationship between facts matters as much as individual values.

Numeric hallucination metric. Our detector counts specific numerical values in Stage B's output that do not appear in the source passage. This is a lower bound on false claims: it misses wrong names, wrong dates formatted differently, wrong causal claims, and hallucinated entities. We name it *numeric* hallucination rate throughout to reflect this scope. The metric is also sensitive to passage composition: our 10 custom passages are number-dense (distances,

dates, masses) and yield higher numeric hallucination rates (e.g., Symbolic: 48% on custom vs. 10% on SQuAD passages for Qwen3; see Appendix B). Readers should interpret the hallucination rates as indicative of structured-format reliability rather than absolute false-claim frequencies.

Dataset composition. Our 30 passages comprise 10 custom (number-dense) and 20 SQuAD (Wikipedia text). Table 1 aggregates both; the custom/SQuAD split in Appendix B shows that headline results are consistent in direction across both subsets, though effect sizes differ.

Greedy decoding. All experiments use temperature 0. Sampling-based generation would introduce variance and may change the relative ranking of protocols on borderline cases.

Single GPU. Results are from a single RTX 5060 Ti. VRAM profiles and GB-seconds ratios will differ on GPUs with different memory bandwidth and compute characteristics.

7 Conclusion

We studied inter-stage communication protocols in temporally-scheduled local LLM pipelines as a first-class design variable. Across three model families, 30 passages, and 8 protocols with causal ablations, prose summary is consistently the most token-efficient protocol and matches or approaches full natural-language accuracy (no significant difference on Qwen3 or Llama; statistically lower but more efficient on Phi-4) while reducing intermediate token count by 44–56% and GB-seconds by 37–49%. Symbolic and codebook formats degrade factual recall and introduce hallucinations; causal ablation confirms that format structure alone carries no factual signal—content is everything. In a heterogeneous cross-family pipeline, temporal model scheduling with prose-summary inter-stage messages achieves 28% higher F1 than a same-model pipeline with the weaker refiner, justifying the DRiPP staged architecture.

Reproducibility. All code, datasets, and result files are available at `[repository URL]`. Experiments run on publicly available pretrained models with no fine-tuning.

Appendix A: Prompts

Stage A system prompt (all protocols). *"You are a precise information synthesizer. Your output will be passed verbatim to another language model that has NO access to the original source text. Capture ALL specific facts accurately."*

Stage B system prompt (all protocols). *"You are a factual question answering model. Answer each question using ONLY information present in the provided intermediate state. Do NOT use external knowledge. If a fact is not present, say 'NOT FOUND'."*

Full per-protocol Stage A instructions are provided in the supplementary code repository.

Appendix B: Custom vs. SQuAD Split (Qwen3-4B, selected protocols)

Table 3: F1 and numeric hallucination rate broken down by passage subset (Qwen3-4B-Instruct-2507, 30 passages total: 10 custom, 20 SQuAD).

Protocol	Subset	F1	Num.Halluc.
Full NL	Custom (10p)	0.821	0.000
Full NL	SQuAD (20p)	0.589	0.000
Prose Summary	Custom (10p)	0.859	0.000
Prose Summary	SQuAD (20p)	0.553	0.010
Symbolic	Custom (10p)	0.475	0.480
Symbolic	SQuAD (20p)	0.442	0.100
Codebook	Custom (10p)	0.480	0.220
Codebook	SQuAD (20p)	0.403	0.020

Direction of results (Summary $\geq$ Symbolic $\approx$ Codebook) is consistent across both subsets. The numeric hallucination metric is substantially more sensitive on number-dense custom passages: Symbolic hallucination is 48% on custom passages vs. 10% on SQuAD. This reflects the hallucination metric's design (numerical value detection), not a qualitative difference in model behaviour.

References

[1] Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awasthi, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

[2] Reza Yazdani Aminabadi, Samyam Rajbhandari, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Olatunji Ruwase, Shaden Smith,

Minjia Zhang, Jeff Rasley, and Yuxiong He. DeepSpeed-Inference: Enabling efficient inference of transformer models at unprecedented scale. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. IEEE, 2022.

[3] Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. In *ICLR 2024 Workshop on Practical Machine Learning for Low Resource Settings*, 2023.

[4] Yixin Dong, Lianmin Zheng, Zihao Ni, Ruihang Shao, Han Liu, Yilong Yang, Beidi Chen, Ion Stoica, and Eric P. Xing. XGrammar: Flexible and efficient structured generation engine for large language models. *arXiv preprint arXiv:2411.15100*, 2024.

[5] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[6] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[7] Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Luo. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 13358–13376. Association for Computational Linguistics, 2023.

[8] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 19274–19286. PMLR, 2023.

[9] Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 963–981. Association for Computational Linguistics, 2024.

[10] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2383–2392. Association for Computational Linguistics, 2016.

[11] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[12] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. FlexGen: High-throughput generative inference of large language models with a single GPU. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 31094–31116. PMLR, 2023.

[13] Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[14] Fangyuan Xu, Weijia Shi, and Eunsol Choi. RECOMP: Improving retrieval-augmented LMs with compression and selective augmentation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[15] Chau Pham Zhang, Weizhe Yuan, and James Glass. Let models speak ciphers: Multiagent debate through embeddings. *arXiv preprint arXiv:2310.06272*, 2023.