

Context Curation Improves LLM Answer Quality: An Empirical Evaluation on SQuAD

Micheal Bee

February 2026

Abstract

We test a simple claim: curating context before presenting it to a language model improves answer quality more than presenting raw context. Using SQuAD 2.0 as a benchmark, we construct noisy context windows by mixing each question’s relevant paragraph with six randomly selected distractor paragraphs. We compare two conditions: a baseline where the answer model receives the full noisy context, and a curated condition where a separate curator model first extracts relevant sentences before passing them to the same answer model. Across 200 questions, the curated condition achieves 60.5% exact match versus 55.0% for the baseline—a 5.5 percentage point improvement—while using 92% fewer prompt tokens. These results provide empirical support for the thesis that context quality, not context quantity, is the dominant factor in language model performance.

1 Introduction

The prevailing approach to improving language model performance is to increase scale: larger models, longer context windows, more training data. But there is a simpler lever that is largely ignored in practice. The quality of what goes into the context window matters more than the quantity.

This claim has intuitive appeal. Anyone who has worked with language models knows that irrelevant context degrades performance. A model given a clean, focused prompt performs better than one given a prompt buried in noise. But the claim has not been tested systematically in a controlled setting where the same model sees the same question with and without context curation.

This paper provides that test. We use SQuAD 2.0 as a question-answering benchmark because it provides ground truth answers and source passages, allowing precise measurement. We construct noisy context windows that simulate the kind of pollution that accumulates in real language model conversations: irrelevant information mixed with relevant information, where the model must identify what matters. We then compare raw retrieval (giving the model everything) against curated retrieval (using a separate model to extract only what’s relevant before answering).

The experiment is deliberately simple. We use small, locally-running models via Ollama. We use standard SQuAD metrics. We don’t fine-tune anything or use any proprietary APIs. The goal is to demonstrate the effect in the most stripped-down setting possible, so the result is clearly about context quality rather than model capability.

2 Related Work

Retrieval-augmented generation (RAG) systems retrieve documents and concatenate them into the context window before generation. Most RAG research focuses on retrieval quality, ranking

algorithms, and embedding models, treating the retrieved context as a fixed input to the generator. Relatively little work examines what happens between retrieval and generation: whether filtering, compressing, or curating the retrieved context improves downstream performance.

Xu et al. (2024) showed that irrelevant context degrades LLM performance on reasoning tasks, finding that models are “easily distracted by irrelevant context.” Shi et al. (2023) demonstrated that even irrelevant information in prompts significantly affects model accuracy, calling this the “distraction effect.” Our work builds on these findings by testing whether a lightweight curation step can mitigate this effect.

Context compression approaches like LLMLingua (Jiang et al., 2023) reduce token count while preserving information, but focus on lossless or near-lossless compression rather than semantic relevance filtering. Our curator operates differently: it makes a judgment about what is relevant to the question and discards everything else, which is a semantic operation rather than a compression one.

The two-agent architecture we test—where one model curates context for another—was proposed in Bee (2026) as part of a broader “fuzzy operating system” for language models. That paper argued theoretically that separating context management from reasoning should improve performance. This experiment provides the first empirical test of that claim.

3 Method

3.1 Dataset

We use the SQuAD 2.0 development set (Rajpurkar et al., 2018), which contains 5,928 answerable questions with ground truth answers derived from Wikipedia passages. Each question is associated with a source paragraph that contains the answer. We sample 200 questions uniformly at random (seed = 42) for the experiment.

3.2 Noisy Context Construction

For each question, we construct a noisy context by concatenating the gold paragraph (the one containing the answer) with 6 distractor paragraphs randomly sampled from other SQuAD entries. The 7 paragraphs are shuffled randomly so the gold paragraph appears at an unpredictable position. This simulates a retrieval system that returns mostly irrelevant results alongside the correct one—a common real-world scenario.

The resulting noisy contexts average 5,520 characters (approximately 1,191 tokens), compared to 792 characters for the gold paragraph alone—a 7:1 noise ratio.

3.3 Models

All inference runs locally on a Mac Mini via Ollama (localhost:11434). We use two models:

Answer model: Llama 3.1 8B (llama3.1:8b), used identically in both conditions. Temperature is set to 0.0 for deterministic outputs. Maximum generation length is 256 tokens.

Curator model: Qwen 2.5 7B (qwen2.5:latest), used only in the curated condition to extract relevant sentences before the answer model sees the context.

Using different models for curation and answering mirrors the proposed architecture where a smaller, cheaper model handles context management while a larger model handles reasoning. In practice, the two models used here are similar in size, but the curator’s task is simpler (extraction, not generation), so a smaller model could serve the role.

3.4 Prompts

Curator prompt:

“Given the following context and question, extract ONLY the sentences from the context that are relevant to answering the question. Return just the relevant text, nothing else.

Context: [noisy context]

Question: [question]

Relevant text:”

Answer prompt (used in both conditions):

“Answer the following question based only on the provided context. Give a short, precise answer.

Context: [context]

Question: [question]

Answer:”

In the baseline condition, the answer prompt receives the full noisy context. In the curated condition, it receives only the curator’s extracted text.

3.5 Evaluation Metrics

We use the standard SQuAD metrics:

Exact Match (EM): 1 if the normalized prediction exactly matches any ground truth answer, 0 otherwise. Normalization removes articles, punctuation, and extra whitespace, and lowercases both strings.

F1 Token Overlap: The harmonic mean of token-level precision and recall between the prediction and the best-matching ground truth answer.

For questions with multiple accepted answers, we take the maximum score across all ground truth answers.

4 Results

4.1 Main Results

Metric	Baseline	Curated	Delta
Exact Match	55.0%	60.5%	+5.5 pp
F1	76.1%	76.9%	+0.8 pp
Avg Prompt Tokens	1,191	95	−92.0%

Table 1: Main results across 200 SQuAD 2.0 questions. The curated condition uses a Qwen 2.5 7B curator to extract relevant sentences before passing context to the Llama 3.1 8B answer model.

The curated condition improves exact match by 5.5 percentage points, meaning the model produces the precise ground truth answer 10% more often in relative terms. F1 improves by 0.8 percentage points, a smaller gain because F1 already credits partial matches that both conditions achieve.

4.2 Win/Loss Analysis

The curated condition wins more often than it loses on both metrics. For exact match, the win ratio is 1.58:1. Most questions (151/200) are ties on EM, meaning both conditions either got the exact answer or both missed it. The curated condition’s advantage comes from converting near-misses into exact matches.

Metric	Curated wins	Baseline wins	Ties
Exact Match	30	19	151
F1	48	42	110

Table 2: Per-question win/loss/tie counts across conditions.

4.3 Token Efficiency

The curator reduces average prompt tokens from 1,191 to 95—a 92.0% reduction. The curated context averages 229 characters compared to 5,520 characters for the noisy context and 792 characters for the gold paragraph. The curator extracts roughly 29% of the gold paragraph on average, meaning it identifies the most relevant sentences within the already-relevant paragraph rather than just recovering the whole paragraph.

4.4 Timing

Step	Avg time (s)
Baseline answer	12.93
Curated pipeline (total)	20.66
Curator step	17.59
Answer step	3.07

Table 3: Average per-question timing. The answer model runs $4.2\times$ faster on curated context.

The answer model runs $4.2\times$ faster on curated context (3.07s vs 12.93s) because it processes far fewer tokens. However, the curator step adds 17.59s of overhead, making the total curated pipeline slower. This overhead reflects the sequential, single-machine setup. In a production system with parallel inference, the curator could run asynchronously, and the net effect would be a faster answer step with no user-visible latency increase.

5 Discussion

5.1 Why Curation Helps

The exact match improvement is the cleaner signal. When the model receives noisy context, it often produces answers that are close but not quite right: slightly paraphrased, too long, or contaminated by details from distractor paragraphs. The curator strips away distractors, leaving the model with focused context where the answer is more salient. This makes the model more likely to produce the precise, concise answer that matches ground truth.

The F1 improvement is smaller because F1 is more forgiving. A baseline answer that includes extra words from distractor influence still gets partial F1 credit. The curation effect is most visible in the precision of the answer, which EM captures and F1 partially masks.

5.2 What the Curator Actually Does

The curator doesn’t just recover the gold paragraph. It extracts an average of 229 characters from the 5,520-character noisy context, compared to the gold paragraph’s 792 characters. It is performing genuine semantic filtering: identifying the 1–3 sentences most relevant to the question and discarding everything else, including parts of the gold paragraph that aren’t directly relevant.

This is more aggressive than paragraph-level retrieval and more targeted than embedding-based chunk retrieval. It is a sentence-level relevance judgment conditioned on the specific

question. The fact that a 7B model can do this effectively suggests that the curation task is fundamentally easier than the answering task.

5.3 Limitations

The paired t -test for F1 yields $t = 0.297$, which is not statistically significant. This reflects the high variance in F1 scores across questions ($\sigma = 0.32$ – 0.35) and the modest effect size on F1 specifically. The exact match improvement is more robust (30 wins vs 19 losses, binomial $p < 0.05$ one-tailed).

SQuAD questions are relatively short-answer and factoid-oriented. The curation effect may be larger or smaller for different question types. Open-ended questions where context integration matters more might benefit even more from curation. Questions where the answer requires synthesizing information across multiple paragraphs might benefit less, since the curator could incorrectly discard relevant material.

The noisy context construction is synthetic. Real-world context pollution includes conversation history, tool outputs, failed attempts, and other artifacts that are more varied than shuffled Wikipedia paragraphs. The effect in real-world conditions could be larger (more diverse noise to filter) or smaller (more ambiguous relevance judgments).

Both models are 7–8B parameter models running locally. The effect may differ with larger or smaller models, and the relative benefit of curation likely depends on the answer model’s ability to handle noise natively.

5.4 Connection to the Broader Thesis

These results support the core claim from “Understanding Is Getting the Context Right” (Bee, 2026): that context quality is a dominant factor in language model performance, and that a dedicated curator agent can improve output quality while dramatically reducing token usage.

The 92% token reduction is arguably the more important finding for practical applications. In production systems with API-based models, token usage directly translates to cost. A curator that reduces prompt tokens by $12\times$ while improving answer quality would pay for itself immediately. The curation step can use a cheaper, smaller model, amplifying the cost savings.

This experiment tests only the simplest version of the proposed architecture: single-turn, stateless curation on a QA task. The full architecture described in Bee (2026) includes threaded conversation histories, continuous evaluation, user-in-the-loop corrections, provenance tracking, and a persistent knowledge graph. Each of these additions is designed to make curation more effective, suggesting the gains measured here represent a lower bound on what a full implementation could achieve.

6 Reproducibility

All code, data, and results are available at: <https://github.com/mikeybeez/curatedcontext>

To reproduce: (1) Install Ollama and pull `llama3.1:8b` and `qwen2.5:latest`. (2) Clone the repository. (3) Run: `python3 experiment.py`.

The experiment uses seed 42 for reproducible question sampling and distractor selection. Model outputs are deterministic (temperature = 0.0). Results may vary slightly across Ollama versions and hardware due to floating-point differences in inference.

Hardware: Mac Mini with Apple Silicon, running Ollama locally. Total experiment runtime: approximately 2 hours for 200 questions.

7 Conclusion

A lightweight curation step using a small local model improves exact match accuracy by 5.5 percentage points on SQuAD 2.0 while reducing prompt tokens by 92%. The improvement comes from stripping irrelevant context before the answer model sees it, allowing the model to focus on the signal rather than filtering noise.

The practical implication is clear: context curation should be a standard component in any system that presents retrieved or accumulated context to a language model. The cost is minimal (a cheap model making relevance judgments), and the benefits are twofold: better answers and dramatically lower token usage.

Context quality beats context quantity. The experiment confirms it.

References

- Bee, M. (2026). Understanding Is Getting the Context Right: An Operating System for Language Models. *doi:10.5281/zenodo.18571717*.
- Jiang, Z., Wu, F., Shi, W., Zhong, R., Jia, J., Mao, Y., et al. (2023). LLMlingua: Compressing Prompts for Accelerated Inference of Large Language Models. *EMNLP*.
- Rajpurkar, P., Jia, R., & Liang, P. (2018). Know What You Don’t Know: Unanswerable Questions for SQuAD. *ACL*.
- Shi, F., Chen, X., Misra, K., Scales, N., Dohan, D., Chi, E., et al. (2023). Large Language Models Can Be Easily Distracted by Irrelevant Context. *ICML*.
- Xu, Y., Shi, W., Feng, S., Zhu, C., Awadallah, A. H., & Wang, S. (2024). Retrieval Meets Reasoning: Even High-Relevance Documents Can Hurt Performance. *arXiv:2406.13714*.