Research Article

In RAG We Trust? Measuring Robustness of Retrieval-Augmented Generation Under Document Poisoning

Iliano Fasolino[1]

1. Department of Computer Science, University of Milan, Italy

Retrieval-augmented generation (RAG) grounds a language model in retrieved documents, which reduces hallucination but creates a new attack surface: if retrieved text is tampered with, the model may repeat the falsehood. We study how much a small quantized model, Llama 3.1 8B, degrades when a fraction of its retrieved context is poisoned. Three corruption strategies are tested, entity swap, number swap, and negation, each applied to zero, one, two, or three of the three retrieved passages, over a factorial sweep of 588 runs on a fact-checking task built from FEVER. Accuracy falls from 77.9% on clean context to 43.5% when all three passages are corrupted. Entity swap is the most damaging in terms of answers flipped from correct to wrong, and number-based corruption shows a sharp threshold once poisoned passages form the majority. The model rarely invents new falsehoods; instead its dominant reaction is to abstain, and hallucination actually drops under attack. These results quantify a practical weakness of RAG and identify abstention, not fabrication, as the main behaviour to plan for.

Correspondence: papers@team.qeios.com — Qeios will forward to the authors

I. Introduction

Large language models answer questions fluently but carry two well-known liabilities: their knowledge is frozen at training time, and they sometimes state false facts with full confidence. Retrieval-augmented generation, proposed by Lewis et al. [1], addresses both by fetching relevant documents at query time and placing them in the prompt as evidence. The model is then asked to answer from that evidence rather

than from memory alone. This design has become the default recipe for question answering over private or fast-moving knowledge bases.

The same design introduces a dependency that is easy to overlook. A RAG answer is only as trustworthy as the documents it retrieves. If an attacker can edit, inject, or otherwise corrupt even part of the retrieved set, the grounding step that was meant to improve reliability becomes a channel for misinformation. Shi et al. [2] showed that irrelevant context alone can pull a model off track; corrupted but on-topic context is a stronger lever, because it looks exactly like the evidence the model was told to trust.

Prior work frames this as an open problem. Zhou et al. [3] list robustness under adversarial or noisy retrieval among the main unsolved trustworthiness issues for grounded models. What is missing for a practitioner running a small model on modest hardware is a concrete measurement: how far does accuracy fall, which kind of corruption bites hardest, and does the model fail loudly by fabricating or quietly by refusing.

We answer those three questions empirically. Our contribution is threefold. First, we build a reproducible poisoning testbed around a 4-bit Llama 3.1 8B model, a MiniLM retriever, and a FAISS index of nearly twenty thousand FEVER evidence sentences. Second, we run a full factorial experiment, three corruption strategies crossed with four poisoning levels over a fixed query set, and report accuracy, a fooled rate, abstention, and a hallucination proxy for every cell. Third, we characterise the failure surface: entity corruption flips the most answers, numeric corruption jumps once it reaches a majority of the context, and the model's default response to contradiction is to abstain rather than to invent. Every number reported here comes from the same 588-run table, so the strategies can be compared on equal footing.

II. Method

A. Retrieval and Generation Pipeline

The pipeline follows the standard three stages of RAG. A user question is embedded, the nearest passages are retrieved from a vector index, and those passages are concatenated into a prompt that the language model answers.

Passages and questions are embedded with the all-MiniLM-L6-v2 sentence transformer [4], which maps text to 384-dimensional vectors and is small enough to run comfortably on a CPU. Vectors are normalised

to unit length and stored in a FAISS flat inner-product index [5], so that the inner product between a query and a passage equals their cosine similarity. For a query q and passage d the retrieval score is

$$sim(q, d) = \frac{\mathbf{e}_q \cdot \mathbf{e}_d}{\|\mathbf{e}_q\| \|\mathbf{e}_d\|}, \tag{1}$$

where $\mathbf{e}_q$ and $\mathbf{e}_d$ are the embeddings. The index holds an exact copy of every vector and is scanned in full at query time, which is appropriate at this corpus size. The top three passages, $k = 3$, are passed to the generator.

The generator is Llama 3.1 8B Instruct, quantized to 4 bits in the GGUF Q4_K_M format and run through llama.cpp on CPU. Full-precision weights would need about 16 GB of memory; 4-bit quantization brings the footprint near 5 GB, which fits the 8 GB machine used for all runs. Decoding uses temperature 0.1 so that answers are close to deterministic, which matters when the same query is evaluated many times. The prompt instructs the model to answer only from the supplied context and to reply “Not enough information” when the context does not settle the question.

B. Poisoning Strategies

Document poisoning here means rewriting a retrieved passage so that it states something false while remaining fluent and on-topic. The rewrite happens after retrieval and before the prompt is built, so the retriever itself is never attacked; only the evidence handed to the model is altered. Three strategies are implemented, each targeting a different class of fact.

Entity swap replaces named entities with plausible alternatives of the same type. “The Eiffel Tower is located in Paris, France” becomes “The Eiffel Tower is located in Madrid, Spain.” Grammar and register are untouched, so nothing flags the sentence as suspicious except the fact itself.

Number swap perturbs a numeric value to a nearby wrong one, for example turning “over 13,000 miles” into “over 3,000 miles.” A language model has no internal arithmetic check for such a claim and tends to echo whatever figure the context provides.

Negation inserts a negation that flips the truth value of a statement, so “Barack Obama was the 44th President” becomes “Barack Obama was never the 44th President.” The word “never” is a strong lexical signal, and we expected the model to treat it with more suspicion than a silent entity substitution.

C. Poisoning Level

The poisoning level is the number of the three retrieved passages that are corrupted before generation. We sweep all four values, from none to all three:

$$p \in \{0/3, 1/3, 2/3, 3/3\}. \tag{2}$$

At $0/3$ the passages are left untouched, giving a within-condition baseline. At $3/3$ every passage the model sees has been rewritten. The passages to corrupt are chosen at random among the three, which lets a single clean passage sit anywhere in the ordering.

D. Evaluation Metrics

For every run the model is queried twice on the same retrieved set, once clean and once poisoned, and four quantities are recorded.

Accuracy checks whether the reference keyword for a question appears in the answer, case-insensitively. *Fooled rate* is the fraction of runs where the clean answer was correct but the poisoned answer was not; it isolates the damage caused by the attack from questions the model could not answer anyway. *Abstention rate* counts answers that decline to commit, detected through a small set of phrases such as “not enough information” and “cannot determine.” *Hallucination* is approximated by lexical overlap: an answer is flagged when fewer than 30% of its content words appear in any retrieved passage,

$$hallu = \mathbf{1}\left[\frac{|W_a \cap W_d|}{|W_a|} < 0.3\right], \tag{3}$$

with W_a the content words of the answer and W_d those of the retrieved passages. This proxy is coarse, and we treat it as a trend indicator rather than a precise detector.

E. Experimental Design

The evaluation set contains 49 fact-checking queries, of which 37 are distinct, spanning locations, dates and counts, yes/no claims, and people. Ten information-rich passages covering the queried facts were added to the index because raw FEVER evidence sentences are often too terse to support a full answer. Crossing the query set with three strategies and four poisoning levels yields $49 \times 3 \times 4 = 588$ runs, each producing a clean and a poisoned answer for $1,176$ generations in total. Because a full sweep takes roughly a day on CPU, results are written to disk after every run and a checkpoint records completed cells, so an interrupted sweep resumes without repeating work.

III. Experiments and Results

A. Setting

All runs use the corpus of 19,597 FEVER dev evidence sentences plus the ten added passages, the MiniLM retriever, and the quantized Llama 3.1 8B generator described above, with $k = 3$ and temperature 0.1 held fixed. The dataset comes from the FEVER benchmark of Thorne et al. [6], which pairs claims with Wikipedia evidence and is a standard resource for fact verification.

B. Accuracy Degradation

Poisoned passages	Clean acc. (%)	Poisoned acc. (%)	Fooled rate (%)	Abstain rate (%)
0 / 3	78.2	69.4	9.5	21.1
1 / 3	77.6	63.9	15.0	29.3
2 / 3	78.2	53.1	25.9	38.8
3 / 3	77.6	43.5	34.0	51.7

Table I. Accuracy and failure behaviour by poisoning level, averaged over all three strategies. Each level aggregates 147 runs.

Table I tracks the model as more of its context is corrupted. The clean arm stays near 78% throughout, which confirms that retrieval quality is stable and that any movement in the poisoned arm is caused by the attack rather than by drift in question difficulty. Poisoned accuracy falls in step with the attack, from 69.4% at one corrupted passage down to 43.5% when all three are corrupted, a total loss of 34 points. The fooled rate more than triples over the same range, and abstention climbs from 21% to 52%.

Figure 1 plots the same trend per strategy. The three curves separate clearly. Entity swap drives the steepest fall, and by the full-poisoning level it sits below the other two. The right panel, which shows the

gap against the clean arm, makes the ordering explicit and shows that no strategy is harmless once two of the three passages are corrupted.

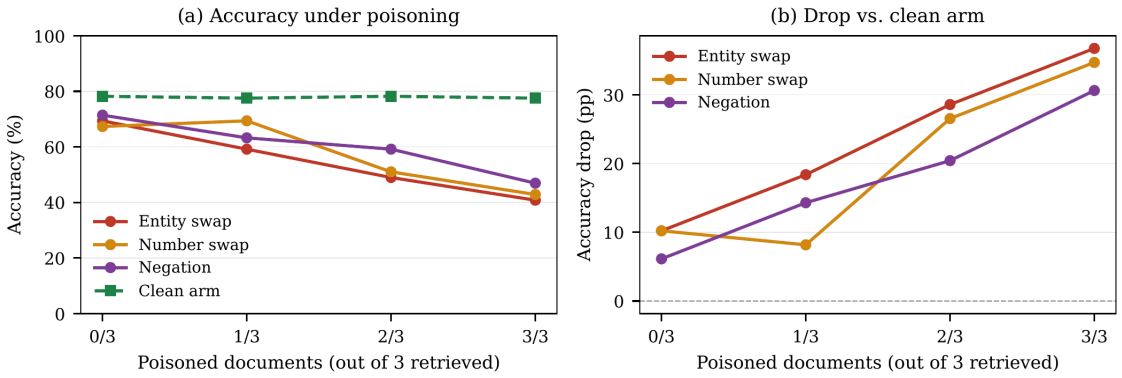

Figure 1. Accuracy as a function of how many of the three retrieved passages are poisoned. Panel (a) shows poisoned accuracy per strategy against the shared clean baseline; the clean arm stays near 78% while every poisoned curve declines. Panel (b) shows the drop relative to the clean arm, where entity swap opens the widest gap at high poisoning.

One detail in Table I needs to be read carefully. The $0/3$ row still shows a fooled rate near 10% and an accuracy about 9 points below the clean arm. That residual comes from two sources. The poisoned arm retrieves independently of the clean arm, so the two do not always see the identical passage ordering, and the reference keyword matcher is strict. We therefore read the clean-arm column, not the $0/3$ poisoned cell, as the true no-attack baseline.

C. Which Strategy Hurts Most

Strategy	Clean	Poisoned	Fooled	Abstain
	acc. (%)	acc. (%)	rate (%)	rate (%)
Entity swap	78.1	54.6	23.5	43.4
Number swap	77.6	57.7	20.4	30.1
Negation	78.1	60.2	19.4	32.1

Table II. Behaviour by strategy, averaged over all four poisoning levels. Each strategy aggregates 196 runs.

Table II ranks the strategies. Entity swap has the highest fooled rate at 23.5% and the lowest poisoned accuracy, so it flips the largest share of answers that the model would otherwise get right. It also provokes the most abstention, 43.4%, which fits its mechanism: swapping “Paris, France” for “Madrid, Spain” can leave the passage internally inconsistent, and the model often reacts to that inconsistency by refusing to answer. Number swap and negation are close behind on fooled rate but keep higher poisoned accuracy.

The heatmap in Figure 2 gives the full strategy-by-level grid. Two patterns stand out. Fooled rates rise from left to right in every row, and the number-swap row is nearly flat across the first two columns before it jumps.

doi.org/10.32388/7HXKQE

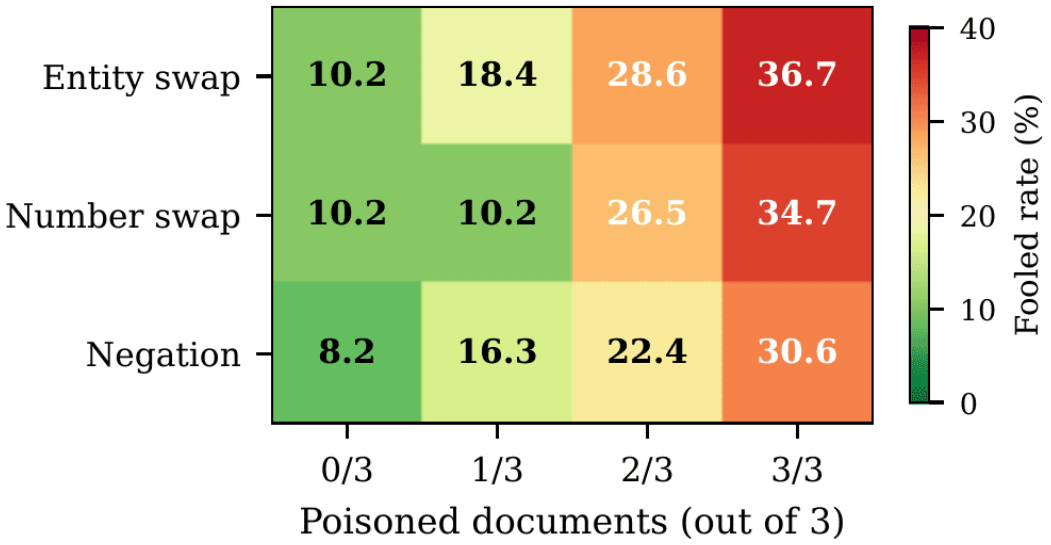

Figure 2. Fooled rate for every strategy and poisoning level. Colour runs from green (safe) to red (many answers flipped). The number-swap row barely changes between zero and one poisoned passage, then rises steeply.

D. A Threshold in Numeric Corruption

Figure 3 isolates that pattern. For entity swap and negation the fooled rate grows in roughly even steps as poisoning increases. Number swap behaves differently. It stays at 10.2% for both zero and one corrupted passage, then leaps to 26.5% at two, a jump of more than 16 points, and continues to 34.7% at three. The plateau then cliff is the signature of a majority effect: a single wrong figure among two correct ones is outvoted, but once two of three passages carry the wrong figure the model follows the majority. Because a model cannot sanity-check a bare number the way it can catch “Madrid, Spain,” numeric corruption is quiet until it reaches critical mass and then turns sharply dangerous.

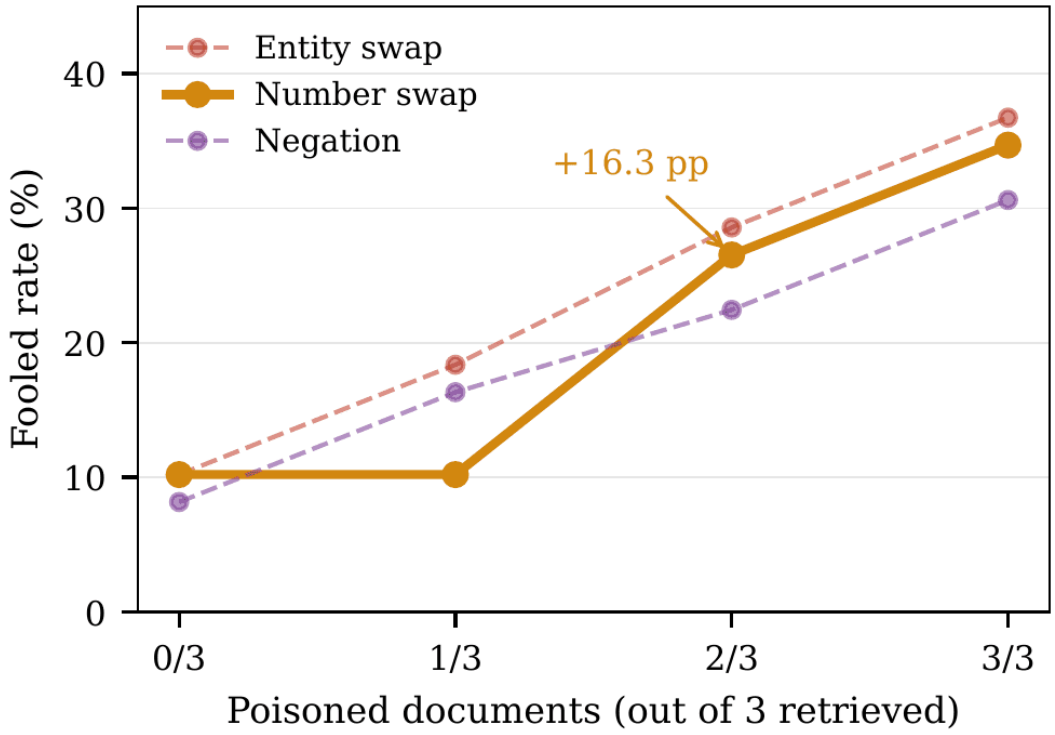

Figure 3. Fooled rate by poisoning level. Entity swap and negation grow steadily, whereas number swap (bold) stays flat through one poisoned passage and then jumps by 16 points once poisoned passages become the majority.

doi.org/10.32388/7HXKQE

E. How the Model Fails

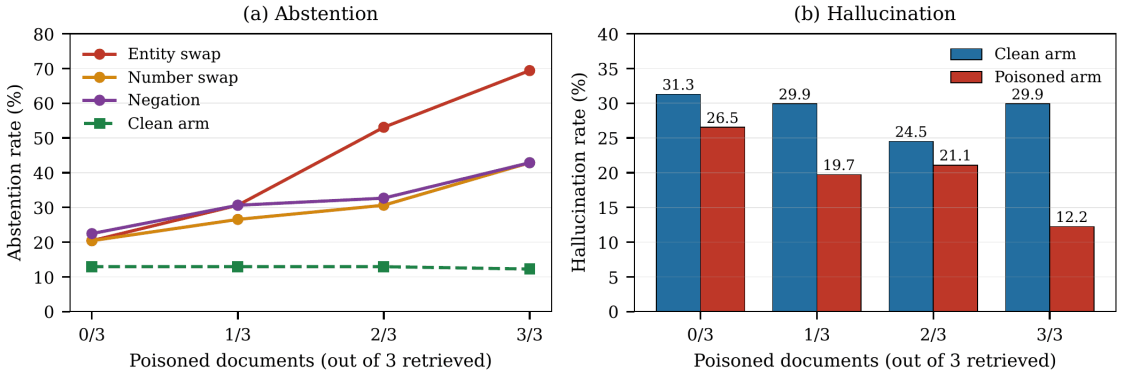

Figure 4. Failure behaviour across poisoning levels. Panel (a): abstention rises with poisoning for every strategy, most steeply for entity swap. Panel (b): hallucination, measured by low overlap with the retrieved passages, is higher on the clean arm than the poisoned arm for all three strategies.

The most useful result for a system designer is not how often the model is wrong but how it is wrong. Two failures are possible under attack. The model can accept the falsehood and state it, or it can notice the conflict and decline. It could also invent a third answer grounded in neither, which the hallucination proxy is meant to catch.

The evidence points firmly to abstention. Averaged over all runs the model abstains on 35.2% of poisoned queries, well above the 21.1% fooled rate. Figure 4(a) shows abstention climbing with poisoning for every strategy and reaching 69% for entity swap at full corruption. Hallucination moves the other way. Figure 4(b) shows that low-overlap answers are more common on the clean arm, near 29%, than on the poisoned arm, near 20%. Corrupted context does not push the model to fabricate; it pushes the model to give up. Contradiction among passages seems to trigger caution rather than invention.

Figure 5 decomposes only those queries the model answered correctly on clean context, so every bar starts from a fact the model knew. As poisoning grows, the green “still correct” band shrinks and is replaced mostly by blue abstention, with the red “fooled” band growing more slowly. Negation is the most resilient: even at full corruption a large share of originally correct answers survives, which matches our expectation that an explicit “never” is easier to distrust than a silent swap.

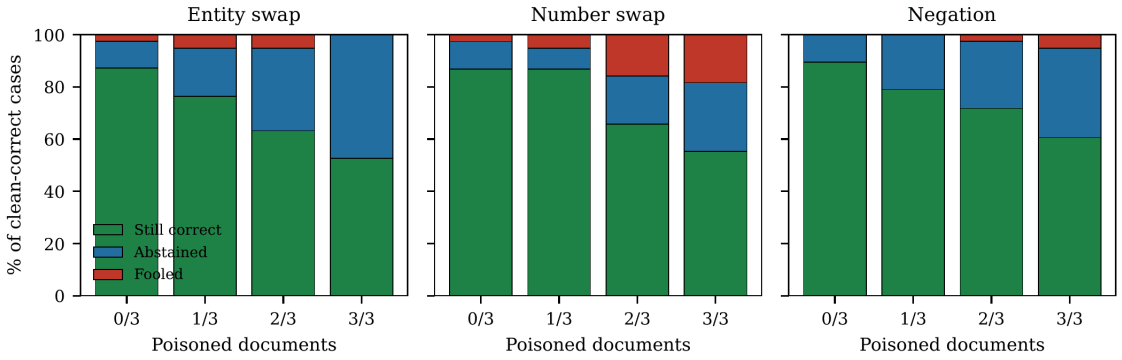

Figure 5. Outcome of queries that were correct on clean context, split by strategy and poisoning level. Green is still correct, blue is abstained, red is fooled. The lost green area is absorbed mainly by abstention, and negation retains the most correct answers.

III-F. Which Queries Break

Fooled rates are uneven across questions. Figure 6 lists the ten most fragile ones. Questions about a well-known person's biography and about the landmark facts in our added passages top the list, because those passages contain the richest set of swappable entities. Yes/no and location questions are more exposed than open numeric ones, whose lower clean accuracy leaves less room for a correct-to-wrong flip. The pattern is a reminder that vulnerability depends on the fact being asked about, not only on the attack.

doi.org/10.32388/7HXKQE

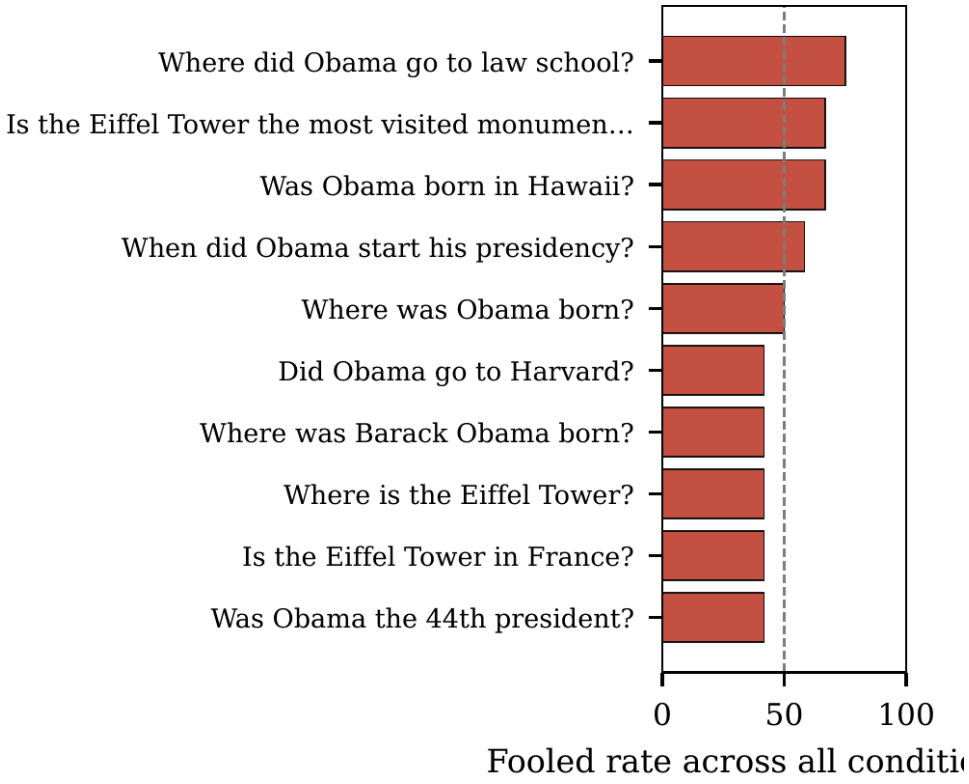

Figure 6. The ten queries with the highest fooled rate across all conditions. Biographical and landmark questions dominate because the passages that answer them carry many swappable entities.

G. Relation to Prior Work

Our findings extend the RAG picture drawn by Lewis et al. [1] from the cooperative case to an adversarial one: retrieval that lifts accuracy on clean corpora becomes a liability on corrupted ones. Where Shi et al. [2] showed that off-topic context distracts a model, we show that on-topic but false context does more, because it wears the appearance of the evidence the model was told to trust. The abstention-over-fabrication behaviour we measure gives an empirical answer to one robustness question raised in the trustworthiness survey of Zhou et al. [3], at least for a small quantized model. The retrieval-integrated pre-training of Guu et al. [7] shares the grounding assumption our attack targets.

IV. Threat Model and Discussion

The attacker assumed here can alter the text of retrieved passages after retrieval and before generation, but cannot retrain the model or change the retriever. This maps onto realistic settings such as a compromised document store, a poisoned web source that a live retriever pulls in, or a tampered cache between retrieval and prompting. The attacker does not need to fool the similarity search; editing content that is already retrieved is enough.

Under that model the results carry three practical messages. Numeric facts deserve special protection, because a model cannot verify a figure and numeric corruption turns sharply harmful once it dominates the context. Cross-checking retrieved numbers against an independent source, or flagging disagreement among passages, is worth the cost in domains where quantities matter. Redundancy helps but only to a point: the majority effect means two trustworthy passages defend against one bad one, yet the guarantee vanishes as soon as corrupted passages take the majority, so raising k without raising source diversity offers little real protection. Finally, abstention should be treated as a signal, not as noise. A model that starts refusing far more often than usual may be reacting to conflicting evidence, and a monitored abstention rate could serve as a cheap early warning of a poisoned store.

V. Conclusion

We measured how a 4-bit Llama 3.1 8B model behaves when its retrieved context is poisoned, across three corruption strategies and four poisoning levels in a 588-run sweep. Accuracy falls from 77.9% to 43.5% as corruption goes from none to total. Entity swap flips the most answers, and numeric corruption follows a threshold, harmless as a minority and dangerous once it forms the majority of the context. The model's main response to attack is to abstain rather than to fabricate, and its hallucination rate falls under poisoning instead of rising.

The study has clear limits. It covers one small quantized model on a compact query set with rule-based corruption, so the absolute numbers should not be read as universal. The hallucination proxy is a lexical heuristic, not a semantic judge, and the added passages make retrieval easier than raw FEVER evidence would. Each limit points to a concrete next step. We plan to repeat the sweep on a larger instruction-tuned model and on a second corpus to test whether the threshold and the abstention behaviour hold, to replace rule-based edits with model-generated adversarial passages that keep entity swaps internally

consistent, and to measure whether a simple defence, such as flagging cross-passage disagreement before generation, recovers accuracy without suppressing correct answers.

References

1. ^a, ^b*Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, Küttler H, Lewis M, Yih WT, Rocktäschel T, Riedel S, Kiela D (2020). "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." Adv Neural Inf Process Syst. **33**:9459–9474.*
2. ^a, ^b*Shi F, Chen X, Misra K, Scales N, Dohan D, Chi EH, Schärli N, Zhou DY (2023). "Large Language Models Can Be Easily Distracted by Irrelevant Context." Proc Int Conf Machine Learning (ICML). 31210–31227.*
3. ^a, ^b*Zhou K, Zhu Y, Chen Z, Chen S, Zhao S, Wen JR (2023). "Don't Make Your LLM an Evaluation Benchmark Cheater." arXiv preprint arXiv:2311.01964.*
4. ^*Reimers N, Gurevych I (2019). "Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks." Proc Conf Empirical Methods in Natural Language Processing (EMNLP). 3982–3992.*
5. ^*Johnson J, Douze M, Jégou H (2019). "Billion-Scale Similarity Search with GPUs." IEEE Trans Big Data. **7**(3): 535–547.*
6. ^*Thorne J, Vlachos A, Christodoulopoulos C, Mittal A (2018). "FEVER: A Large-Scale Dataset for Fact Extraction and Verification." Proc Conf North American Chapter of the Association for Computational Linguistics (NAACL-HLT). 809–819.*
7. ^*Guu K, Lee K, Tung Z, Pasupat P, Chang M (2020). "REALM: Retrieval-Augmented Language Model Pre-Training." Proc Int Conf Machine Learning (ICML). 3929–3938.*

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.