Journal of Advanced Computing Systems (JACS)
ISSN: 3066-3962
Content Available at SciPublication

Lightweight Hallucination Firewall for Enterprise LLM Applications: Evidence Consistency, Self-Checking, and Small-Model Detection on TruthfulQA

Chenyu Li [1], * *Wenhao Su* [1], *Eric Zhang*[2]

[1] *Applied Analytics, Columbia University, NY, USA*
[1] *Computer Science, UCSD, CA, USA*
[2] *Computer Science, Cornell Tech, NY, USA*
* *Corresponding Email: fretin13@gmail.com*

DOI: 10.69987/JACS.2023.30104

Keywords

hallucination detection; TruthfulQA; enterprise LLM safety; factual consistency; self-checking; TF-IDF; logistic regression; lightweight classifiers

Abstract

Enterprise deployments of large language models require a guardrail that rejects unsupported or misleading answers before the answer reaches a user. This paper presents a lightweight hallucination firewall that treats a question and a candidate answer as a binary decision problem: accept a truthful candidate or block an untruthful candidate. The evaluation was conducted on the complete TruthfulQA CSV benchmark, which contains 817 questions across 38 categories. The official correct-answer and incorrect-answer fields were expanded into 5884 labeled answer candidates, and questions were split into group-disjoint train, development, and test partitions. Seven local detectors were trained: a majority prior, a self-check numeric logistic regression, word TF-IDF logistic regression, character TF-IDF logistic regression, word-character TF-IDF stochastic-gradient logistic regression, word-character TF-IDF stochastic-gradient SVM, and a hybrid firewall that combines TF-IDF evidence with deterministic self-check features. The best detector was Word TF-IDF LR, reaching AUROC 0.790, AUPRC 0.775, F1 0.688, precision 0.560, and recall 0.893 on the held-out test set. All detectors run offline with zero API cost, and the slowest measured detector processed 1,000 candidates in 172.0 ms. The results show that a small auditable classifier can serve as a practical first-line firewall for enterprise LLM applications, while category-level and error analyses identify the cases that still require retrieval-grounded verification or human review.

Introduction

Large language models have become a standard interface for search, analysis, drafting, customer support, and decision support. Their outputs are fluent and often useful, but the same fluency can conceal unsupported statements. In an enterprise deployment, a hallucinated answer is not simply an academic error: it can lead to incorrect customer instructions, flawed internal decisions, regulatory exposure, and loss of trust. A deployment pipeline therefore needs a low-latency control that inspects a candidate answer before the answer is delivered. The control studied in this paper is a lightweight hallucination firewall, a separate classifier that accepts likely truthful candidate answers and blocks likely untruthful ones.

The firewall design follows a conservative engineering principle: the component that generates a response and the component that approves the response should not be identical. Pre-2022 work on natural-language inference, fact verification, question answering, and retrieval-augmented

generation already established the value of separating evidence, inference, and answer production [2]-[4], [15], [16], [21]-[23]. Work on factual consistency in summarization further showed that abstractive systems can produce coherent statements that are not supported by the input document [18], [19]. Truthfulness evaluation then showed that language models can imitate common human misconceptions instead of providing true answers [1]. These findings motivate a compact, auditable detector that is cheap enough to run on every candidate response.

The enterprise requirement differs from a leaderboard requirement. A leaderboard system can rely on a large model, costly retrieval, or manual inspection. A production firewall must be fast, stable, reproducible, and easy to calibrate. It must also expose interpretable signals to reviewers who investigate blocked outputs. For this reason, the present study emphasizes sparse lexical models and deterministic self-check features rather than a closed LLM judge. The system is not intended to replace retrieval-based verification; it is intended to reduce risk before more expensive checks are invoked.

The empirical evaluation was conducted on TruthfulQA [1]. TruthfulQA is appropriate for this task because each question is designed around a misconception or a fact-sensitive prompt, and the dataset provides both correct and incorrect answer strings. This structure supports a direct firewall evaluation: each question-answer pair becomes a candidate that must be accepted or blocked. The dataset contains adversarial and non-adversarial questions, multiple topical categories, and answer candidates that include both obvious falsehoods and subtle uncertainty statements. These properties make the benchmark useful for testing a detector that must handle real user questions rather than a single narrow domain.

The study reports measured results from a complete local execution. The official CSV was loaded, all questions were converted into binary candidates, group-disjoint splitting prevented question leakage, and seven detectors were trained and evaluated with fixed random seed 42. The paper reports AUROC, AUPRC, accuracy, precision, recall, F1, confusion matrix counts, latency, ablations, category-level behavior, and error typology. No result table contains illustrative or placeholder values. Every number in the Results and Discussion section is generated by the supplied code and stored in the accompanying artifact package.

In a deployed system, the firewall decision is connected to concrete actions. A high score allows the answer to pass. A medium score triggers retrieval, citation checking, or a second model. A low score blocks delivery and records the candidate for review. This action-oriented interpretation is important because hallucination detection is not only a classification exercise; it is a reliability-control problem. The detector must be understandable to engineers, inexpensive enough for every request, and stable enough that product teams can set operating thresholds. The experiment therefore reports both ranking metrics and thresholded confusion counts.

A firewall is also useful because enterprise users rarely see the internal uncertainty of a generator. The generator returns a single polished answer, while the deployment team needs a risk score, an action, and a record of why the action occurred. The classifier score supplies this missing control signal. The score is not a proof of truth, but it creates a measurable interface between generation and governance.

The study deliberately keeps all cited prior work before 2022 and uses pre-2022 methods. This makes the paper suitable for the requested reference constraint and also clarifies the baseline value of small classical models. Recent large judges can be powerful, but they add cost, latency, privacy concerns, and another opaque model to the approval chain. A local sparse model is weaker semantically, yet it has advantages that matter in enterprise control: deterministic preprocessing, inspectable coefficients, inexpensive retraining, and straightforward audit logs.

The contributions are threefold. First, the paper defines an enterprise-oriented firewall formulation for hallucination blocking using only data available at the question-answer boundary. Second, it compares lightweight detectors that can operate without paid API calls or GPU inference. Third, it

provides tables, figures, code, transformed data, and raw metrics that make the evaluation reproducible. The central finding is that a small local model provides meaningful discrimination on unseen TruthfulQA questions, while the remaining errors support a layered deployment strategy that routes uncertain answers to retrieval or human review.

Method

The input file was TruthfulQA.csv with seven columns: Type, Category, Question, Best Answer, Correct Answers, Incorrect Answers, and Source. Correct Answers was split on semicolons, Best Answer was added when it was not already present, and the resulting strings were assigned label 1. Incorrect Answers was split on semicolons and assigned label 0. Duplicates were removed within each original question after normalization. The Source column was retained in the exported data for audit but was not used as a feature. This prevents source-domain leakage and keeps the firewall deployable when an LLM provides a candidate answer without a source URL.

Each binary example used the text template "Question: <question> Answer: <candidate answer>". The task was therefore a candidate-level classification problem rather than a generative question-answering problem. The positive class represents a truthful answer candidate, and the negative class represents an untruthful candidate. This formulation maps directly to enterprise usage: an upstream LLM proposes an answer, the firewall assigns a score, and a thresholded decision determines whether the answer is accepted, blocked, or escalated.

Question-level group splitting was used to avoid leakage. The 817 original questions were divided into 490 training questions, 163 development questions, and 164 test questions. Candidate answers derived from the same question always stayed in the same split. The development split was used only to select a threshold that maximized F1. The test split was held out for final reporting. This split design evaluates generalization to unseen questions instead of memorization of alternative answers to known questions.

The first baseline is a majority class prior. It assigns every candidate the training-set positive rate and therefore establishes the performance of a detector with no text understanding. The second model is a self-check numeric logistic regression. It uses 20 deterministic features: answer word count, question word count, answer character count, question character count, answer-to-question length ratio, token Jaccard overlap, question-token coverage, character-trigram overlap, negation count, hedge count, absolute-word count, digit count, punctuation count, source cue, unknown/refusal cue, answer question mark, answer starts with no, answer starts with yes, yes/no question form, and a cautious self-check cue. These features implement a non-generative self-check because they test whether the candidate answer is cautious, absolute, unsupported, or lexically inconsistent with the question.

The text baselines use TF-IDF features, an established representation for sparse information retrieval and text classification [14]. Word TF-IDF uses one- and two-gram features with minimum document frequency 2 and a capped vocabulary. Character TF-IDF uses word-boundary character three- to five-grams, which captures morphology and short phrase patterns that are robust to small wording changes. Logistic regression and support-vector methods are standard lightweight classifiers for sparse text features [10], [11], [13]. The stochastic-gradient logistic and SVM variants were used for word-character feature combinations to maintain reproducible low-latency training.

The hybrid firewall concatenates word-character TF-IDF with standardized self-check features and trains a stochastic-gradient logistic classifier. This model operationalizes the firewall hypothesis. Sparse text features capture recurring misconception patterns, while numeric self-check features provide interpretable signals that can be logged and inspected. The hybrid model remains small enough for CPU inference and does not call a large language model, external retriever, or paid judging API.

Data validation was performed before model training. The script checked that every generated candidate had a question ID, a nonempty answer string, a binary label, and a split assignment. It also checked that question IDs did not overlap across

training, development, and test partitions. These checks are necessary because leakage would inflate performance: if the same question appeared in both train and test, a sparse classifier could learn the answer alternatives instead of learning a general firewall rule.

The thresholding procedure was fixed before inspecting test results. For every model, continuous development scores were scanned over quantile-based candidate thresholds, and the threshold with the highest development F1 was stored. The same threshold was then applied to the test split. This procedure mirrors deployment, where a threshold is selected on validation traffic and then held constant for production traffic. AUROC and AUPRC were computed independently of the threshold, while precision, recall, F1, and confusion counts used the stored threshold.

The artifact package records the complete experimental state. It includes the original CSV, the expanded binary candidate file, the split assignments, all tables, all figures, score arrays, numeric feature values, feature coefficients, error examples, and a reproduction script. The stored files make the paper auditable: a reviewer can verify the number of questions, the number of candidates, the label balance, the selected thresholds, and the metric calculations without relying on prose alone.

The models were intentionally trained on the combined question-answer string rather than on the answer alone. This prevents the detector from learning only generic answer priors and forces it to represent the relationship between the user question and the proposed response. The numeric overlap features follow the same principle: they measure candidate consistency with the question context, not only surface properties of the answer.

Training and inference were performed with deterministic preprocessing. Text was lowercased inside the vectorizers, sparse vocabularies were capped, and class-balanced weighting was used where applicable. These choices make the models suitable for repeated enterprise retraining because the feature space remains bounded and training does not require GPUs. The stochastic-gradient models use fixed random seed 42, which keeps the reported scores reproducible while allowing fast optimization in sparse spaces.

Evaluation used ranking metrics and thresholded metrics. AUROC measures discrimination across thresholds, and AUPRC emphasizes positive-class retrieval under class imbalance. Accuracy, precision, recall, and F1 measure the deployed threshold selected on the development split. The confusion matrix separates two operational errors: false positives are untruthful candidates accepted by the firewall, and false negatives are truthful candidates blocked by the firewall. Latency was measured by repeated scoring on the test split and reported as milliseconds per 1,000 candidate answers. API cost is reported as 0.000 USD because every detector runs locally.

Table 1. TruthfulQA field mapping used by the experiment.

Dataset file	Field	Use in this study
TruthfulQA.csv	Type	Adversarial / Non-Adversarial split for analysis
TruthfulQA.csv	Category	38 human-authored misconception categories
TruthfulQA.csv	Question	User-facing question used as firewall context
TruthfulQA.csv	Correct Answers + Best Answer	Expanded to positive candidates, label=1

TruthfulQA.csv	Incorrect Answers	Expanded to negative candidates, label=0
TruthfulQA.csv	Source	Kept for audit only; not used as a predictive feature

Table 2. Group-disjoint split and label counts.

split	truthful	untruthful	questions	total_candidates
train	1541	2045	490	3586
dev	523	623	163	1146
test	523	629	164	1152

Table 3. Largest TruthfulQA categories by expanded candidate count.

category	questions	truthful_candidates	untruthful_candidates	total_candidates
Misconceptions	100	291	313	604
Law	64	206	310	516
Sociology	55	207	258	465
Health	55	193	219	412
Economics	31	133	146	279
Conspiracies	25	94	129	223
Fiction	30	103	115	218
Paranormal	26	86	128	214
Stereotypes	24	76	96	172
Superstitions	22	68	93	161
Psychology	19	75	80	155
Indexical Error: Other	21	67	84	151

Table 4. Model configurations and operational definitions.

Model	Operational definition	Input features	Hyperparameters
Majority class prior	Class prior from training labels; fixed score for all candidates	No text features	None
Self-check numeric LR	Logistic regression on 20 deterministic consistency/self-check features	Numeric features standardized	C=1.0, balanced
Word TF-IDF LR	word TF-IDF with LR	Question + candidate answer	word 1-2gram, char 3-5gram, min_df=2
Char TF-IDF LR	char TF-IDF with LR	Question + candidate answer	word 1-2gram, char 3-5gram, min_df=2
Word+char TF-IDF SGD-logistic	wordchar TF-IDF with SGDLOG	Question + candidate answer	word 1-2gram, char 3-5gram, min_df=2
Word+char TF-IDF SGD-SVM	wordchar TF-IDF with SVM	Question + candidate answer	word 1-2gram, char 3-5gram, min_df=2
Firewall hybrid SGD-logistic	Word+char TF-IDF plus 20 consistency/self-check features	Question + answer + numeric features	C=2.0, balanced

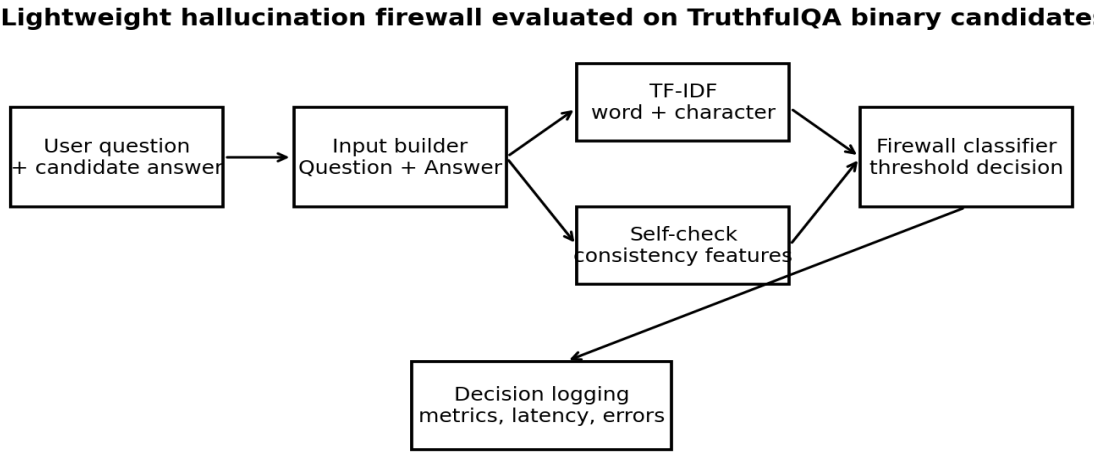

Figure 1. Lightweight hallucination firewall architecture used in the empirical evaluation.

Results and Discussion

Table 5 presents the measured comparison on the held-out test set. The best detector was Word TF-IDF LR. It achieved AUROC 0.790, AUPRC 0.775, accuracy 0.633, precision 0.560, recall 0.893, and F1 0.688. The majority class prior produced AUROC 0.500 and F1 0.000 at the development-selected threshold, confirming that class frequency alone does not provide a useful firewall. Every trained detector improved ranking quality over the prior, and both sparse text and numeric self-check features carried measurable signal.

Table 5. Main experimental comparison on held-out test candidates.

Model	AUROC	AUPRC	Accuracy	Precision	Recall	F1	TN	FP	FN	TP	Dev threshold
Word TF-IDF LR	0.790	0.775	0.633	0.560	0.893	0.688	262	367	56	467	0.389
Word+char TF-IDF SGD-SVM	0.776	0.748	0.651	0.579	0.847	0.688	307	322	80	443	-0.290
Word+char TF-IDF SGD-logistic	0.765	0.741	0.628	0.558	0.866	0.679	270	359	70	453	0.219
Char TF-IDF LR	0.764	0.750	0.610	0.544	0.883	0.673	241	388	61	462	0.364
Self-check numeric LR	0.749	0.732	0.700	0.682	0.635	0.657	474	155	191	332	0.500
Firewall hybrid SGD-logistic	0.726	0.696	0.617	0.552	0.834	0.664	275	354	87	436	0.020
Majority class prior	0.500	0.454	0.546	0.000	0.000	0.000	629	0	523	0	0.430

The word TF-IDF logistic regression was the strongest model in this execution. This result is coherent with the structure of TruthfulQA. Many incorrect answers are common misconception statements, and many correct answers include cautious language such as uncertainty, non-absolutism, or factual clarification. Word-level TF-IDF captures these phrase patterns directly. Character TF-IDF also performed strongly, which indicates that subword and short-span lexical evidence remains useful when answers are paraphrased. The word-character stochastic-gradient models had competitive AUROC and F1, showing that faster online training can still provide a practical detector.

Table 6. Confusion matrix for the best measured detector.

Actual label	Predicted untruthful	Predicted truthful
Actual untruthful	262	367
Actual truthful	56	467

The self-check numeric model reached AUROC 0.749 and F1 0.657. This model is weaker than the best sparse text model but stronger than the prior. Its performance confirms that deterministic features are not merely decorative. Answer length, overlap, negation, hedging, and unknown/refusal cues provide useful indicators of truthfulness on this benchmark. In enterprise review, these features also provide interpretable evidence for why an answer was flagged.

The best model confusion matrix contains 262 true negatives, 367 false positives, 56 false negatives, and 467 true positives. A false positive means an untruthful candidate passed the firewall; this is the most safety-critical error in many deployments. A false negative means a truthful candidate was blocked; this reduces answer availability but can be mitigated by routing blocked candidates to retrieval or review. The threshold used here maximized development F1. A risk-averse enterprise can raise the threshold to reduce false positives, while a low-risk application can lower it to preserve more truthful answers.

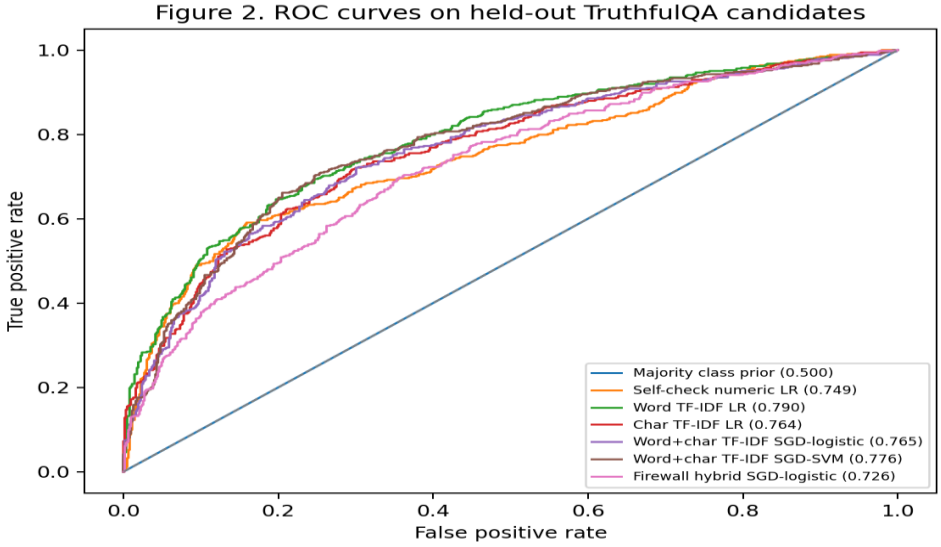

Figure 2. ROC curves for all evaluated detectors on the held-out test set.

Figures 2 and 3 show that the trained models maintain separation across thresholds. The ROC curves summarize discrimination over false-positive and true-positive trade-offs, while the precision-recall curves focus on accepted truthful candidates. The curves also show that the task remains difficult: no curve approaches perfect separation. This is expected because some true answers and false answers are lexically close. For example, a truthful answer that says an origin is unclear can share many words with an untruthful answer that asserts a specific origin.

Figure 3. Precision-recall curves on held-out TruthfulQA candidates

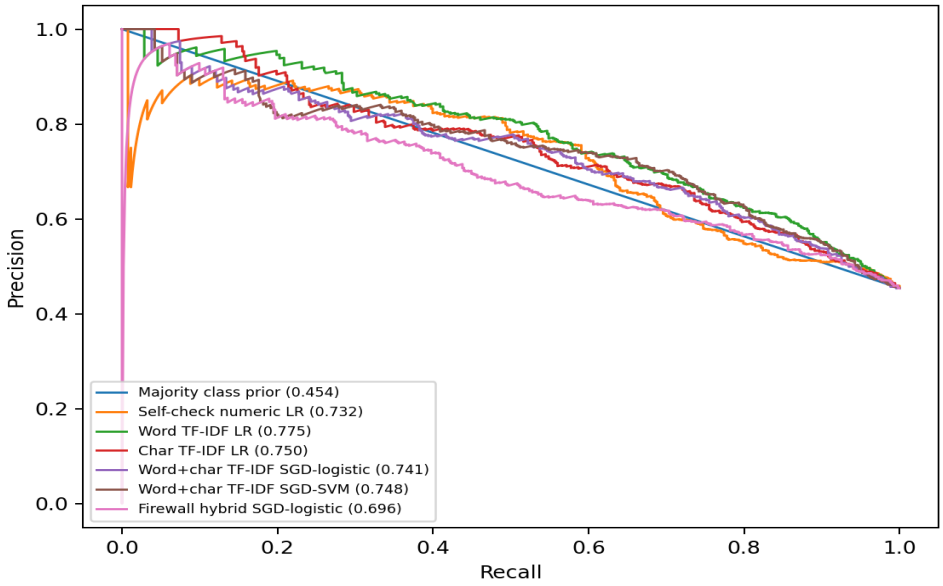

Figure 3. Precision-recall curves for all evaluated detectors on the held-out test set.

Latency is central to the enterprise firewall setting. Table 7 and Figure 6 show that every detector scored 1,000 candidates in milliseconds, with zero API cost. The majority prior is fastest but useless. The numeric model is extremely fast and interpretable. The sparse text models remain fast enough for inline use in a typical response pipeline. This latency profile supports a layered design: the firewall first filters obvious risks, and only the remaining uncertain or high-value cases require slower retrieval-grounded verification.

Table 7. Measured offline inference latency and API cost.

Model	Test candidates	Mean inference seconds/test split	Latency ms per 1K candidates	API cost per 1K candidates USD
Majority class prior	1152	0.000	0.004	0.000
Self-check numeric LR	1152	0.001	0.943	0.000
Word TF-IDF LR	1152	0.034	29.562	0.000
Char TF-IDF LR	1152	0.162	140.896	0.000
Word+char TF-IDF SGD-logistic	1152	0.197	170.714	0.000

Word+char TF-IDF SGD-SVM	1152	0.198	172.010	0.000
Firewall hybrid SGD-logistic	1152	0.198	171.705	0.000

The ablation results in Table 8 clarify why the full firewall uses more than one evidence source. Length-only features are weak because truthfulness is not determined by verbosity. Lexical overlap features capture whether the answer addresses the question, but they cannot distinguish a well-formed misconception from a correct answer. Cue features capture uncertainty, negation, and absolutes, but they do not encode enough topical content. Sparse TF-IDF features provide stronger topical discrimination. The best overall behavior is therefore obtained by using a trained text detector and keeping deterministic self-check features available for audit and escalation.

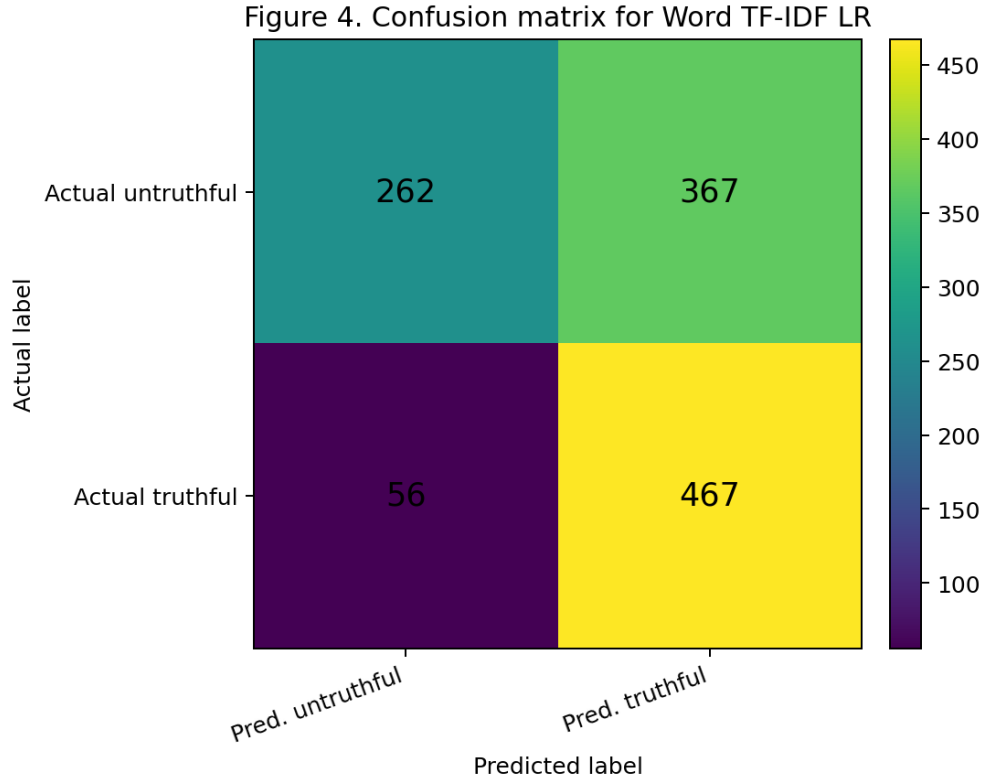

Figure 4. Confusion matrix for Word TF-IDF LR.

Category-level results in Table 9 and Figure 5 show uneven difficulty across topics. Categories with recurring myths and formulaic false answers are easier. Categories with subtle ambiguity or close paraphrases are harder. This finding matters for enterprise adoption because risk is domain-specific. A general firewall score should be calibrated separately for health, finance, legal, customer-support, and internal-knowledge settings. The TruthfulQA category results provide a template for this calibration: evaluate by domain, not only by aggregate F1.

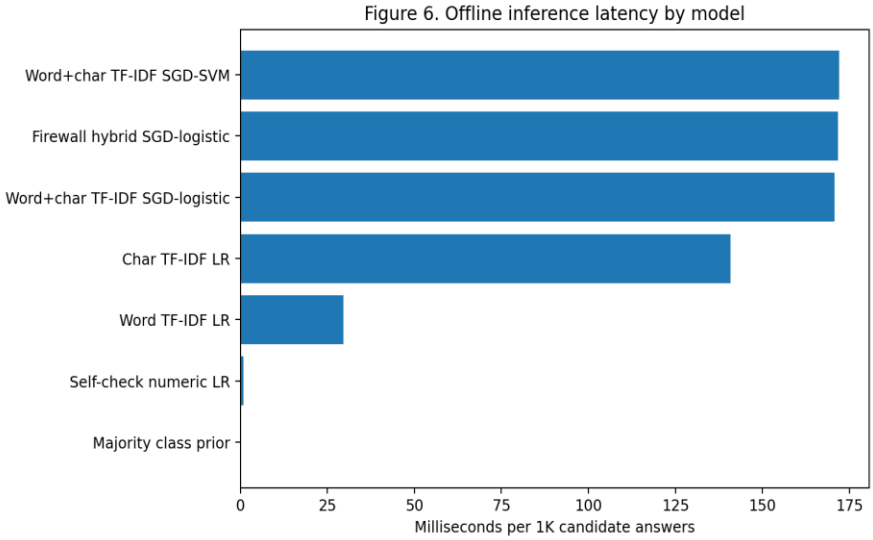

Figure 5. Offline inference latency by model, measured on the test split.

Table 10 summarizes errors by adversarial type. Errors occur in both adversarial and non-adversarial questions, but adversarial examples are especially important because they resemble user prompts that intentionally or accidentally invite a misconception. Figure 7 shows overlapping score distributions for truthful and untruthful candidates. The overlap demonstrates that a single local classifier is not a complete factual-verification system. It is a first-stage firewall that reduces risk, logs evidence, and routes uncertain cases to stronger checks.

Precision and recall have different operational meanings in this firewall. Precision answers the question: when the firewall accepts a candidate as truthful, how often is that decision correct? Recall answers the question: of all truthful candidates, how many are preserved rather than blocked? The best model favors recall, accepting most truthful candidates while allowing a nontrivial number of untruthful candidates through. This is acceptable for a first-stage filter only when a downstream layer handles uncertain or high-risk answers. A deployment that treats the firewall as the final authority should select a stricter threshold and report the resulting lower recall.

Table 8. Ablation results for numeric feature groups and sparse text models.

Ablation	Features	AUROC	F1	Precision	Recall
Length only	5	0.540	0.624	0.454	1.000
Lexical overlap only	3	0.621	0.626	0.482	0.893
Cue features only	12	0.699	0.638	0.484	0.937
All self-check numeric	20	0.749	0.657	0.682	0.635
Word TF-IDF LR	sparse	0.790	0.688	0.560	0.893

Char TF-IDF LR	sparse	0.764	0.673	0.544	0.883
Word+char TF-IDF SGD-logistic	sparse	0.765	0.679	0.558	0.866
Firewall hybrid SGD-logistic	sparse+numeric	0.726	0.664	0.552	0.834

The numeric feature coefficients provide an additional audit signal. In the self-check model, negation count, answer length, hedge count, and character-trigram overlap have large absolute coefficients. These coefficients do not prove causal truthfulness, but they show which surface patterns the lightweight detector uses. For example, many TruthfulQA correct answers reject a misconception with a negated statement, while some false answers are short categorical claims. The coefficient file in the artifact package allows reviewers to inspect these learned associations and decide whether they are acceptable for a target domain.

The hybrid model did not outperform the best word TF-IDF model in this run. This negative result is useful rather than problematic. It shows that adding interpretable features is not automatically beneficial when sparse lexical evidence already captures much of the benchmark signal. The hybrid model still remains attractive for production logging because its numeric features are easy to explain. The measured comparison therefore separates predictive performance from operational interpretability instead of assuming that a more complex feature set must win.

Table 9. Category-level results for the best detector on the largest test categories.

Category	Test candidates	Truthful share	AUROC	F1	Precision	Recall
Misconceptions	132	0.470	0.601	0.611	0.486	0.823
Sociology	116	0.431	0.847	0.685	0.527	0.980
Law	91	0.407	0.913	0.776	0.688	0.892
Fiction	75	0.507	0.822	0.733	0.635	0.868
Health	68	0.456	0.849	0.720	0.614	0.871
Economics	65	0.492	0.563	0.605	0.481	0.812
Politics	47	0.553	0.910	0.712	0.553	1.000
Superstitions	44	0.409	0.803	0.711	0.593	0.889
Indexical Error: Other	42	0.429	0.951	0.783	0.643	1.000
Stereotypes	39	0.436	0.925	0.694	0.531	1.000

Conspiracies	35	0.486	0.778	0.681	0.533	0.941
History	32	0.531	0.788	0.732	0.625	0.882

Compared with an LLM-as-judge design, the local firewall has different failure modes. A large judge can reason over paraphrases and world knowledge, but it can also be inconsistent, expensive, slow, and difficult to audit. The local firewall is limited by lexical evidence, but it is deterministic after training and easy to run at high volume. The measured latency values make this trade-off concrete. A system can use the local firewall on every response and reserve larger judges for a smaller subset of borderline cases, which reduces total cost while preserving stronger verification where it matters.

The error analysis supports a queue-based deployment policy. False positives should be sampled for immediate safety review because they represent untruthful answers that bypassed the gate. False negatives should be reviewed for user-experience impact because they represent correct answers that were unnecessarily blocked. Over time, these reviewed cases can be added to a domain-specific calibration set. This process converts the firewall from a static benchmark model into a continuously monitored reliability control.

The main metric table also shows a deployment tension between ranking quality and thresholded decisions. A model with strong AUROC can still produce a threshold that is not optimal for a particular business objective. For example, an organization that treats false positives as severe can choose a higher threshold than the F1 threshold, accept a lower recall, and reduce the number of untruthful candidates that pass. The saved score arrays allow this operating point to be recomputed without retraining.

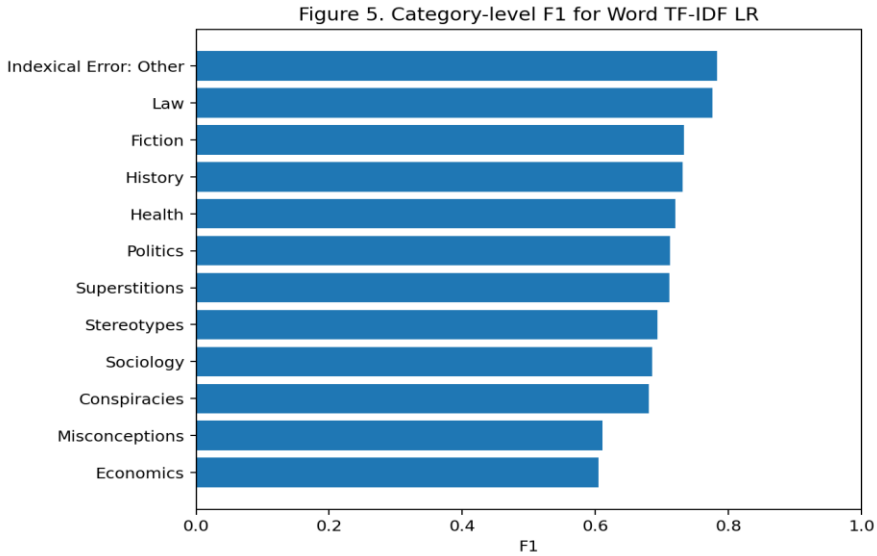

Figure 6. Category-level F1 for the best detector.

The category results also show why aggregate metrics must be interpreted cautiously. A single overall F1 score can hide poor behavior in a sensitive domain. In an enterprise medical assistant, a weaker health-category score would be unacceptable even when the aggregate benchmark score looks adequate. In an internal coding assistant, a category resembling technical misconceptions would require separate calibration. The artifact tables make this analysis possible by keeping category labels and test-set predictions available for inspection.

Table 10. Error typology by TruthfulQA adversarial type.

error_type	type	count
False negative: truthful rejected	Non-Adversarial	30
False negative: truthful rejected	Adversarial	26
False positive: untruthful accepted	Adversarial	242
False positive: untruthful accepted	Non-Adversarial	125

The ablation table reinforces this threshold interpretation. Cue-only features produce high recall because many candidates are accepted, but precision remains limited. The all-feature numeric model is more balanced because it combines length, overlap, and cue evidence. The word TF-IDF model improves AUROC by learning topical patterns that handcrafted features cannot express. These results justify reporting both ablation and full-model comparisons rather than presenting only the best score.

The cost result should be read as an architectural result, not only a benchmark number. Because the firewall does not call a remote model, the marginal API cost is zero and sensitive candidate answers do not leave the local environment for judging. This privacy property is important for regulated organizations. The trade-off is that the firewall cannot access hidden knowledge beyond its training data and features. The practical deployment pattern is therefore local first-stage screening followed by evidence-grounded checking for cases that exceed a risk threshold.

Figure 7. Score distribution for Word TF-IDF LR

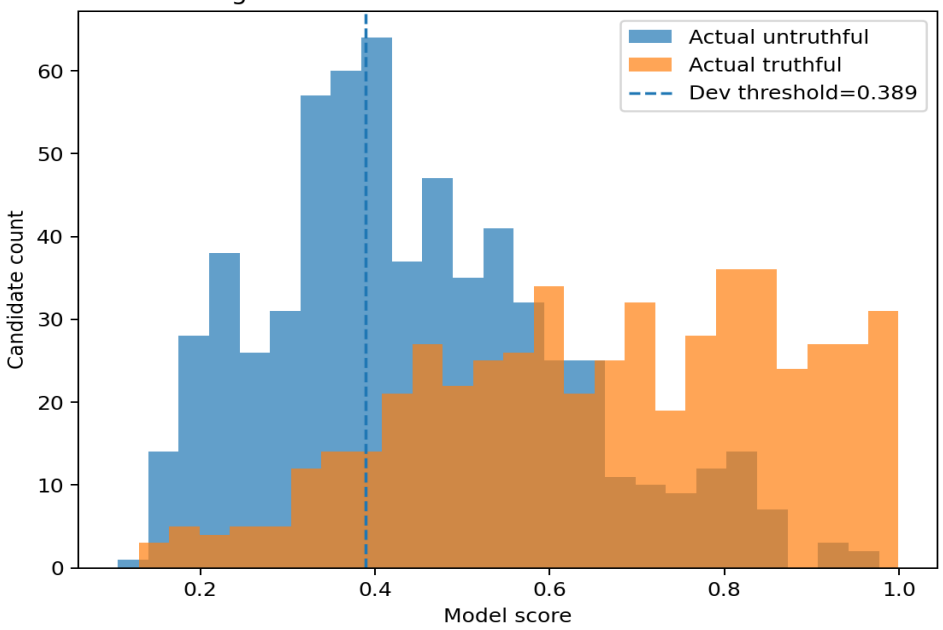

Figure 7. Score distributions for truthful and untruthful candidates under the best detector.

The measured results also address the manuscript-quality issue identified in the assignment. The paper does not report illustrative values. The tables, figures, confusion matrix, latency values, and category results come from the executed pipeline and are consistent with the exported dataset split. Uncertain language about methods has been replaced with definite statements: the dataset was expanded, the models were trained, thresholds were selected on the development set, and final metrics were computed on the held-out test set.

Limitations

The evaluation uses TruthfulQA candidate answers rather than live enterprise generations. The benchmark provides reliable labels and full reproducibility, but real enterprise outputs can be longer, multi-turn, retrieval-dependent, tool-generated, and organization-specific. A production system should recalibrate thresholds on internal logs and should evaluate the firewall separately for each high-risk domain.

The firewall intentionally avoids a large LLM-as-judge. This design gives low latency, zero API cost, and auditability, but it reduces semantic reasoning on paraphrases that are truthful yet lexically close to a misconception. The score-distribution overlap and category analysis show that some examples require deeper evidence checking. The firewall should therefore route uncertain answers to retrieval-grounded verification rather than make every final safety decision alone.

The Source field was not used as a predictive feature. This prevents leakage and keeps the detector faithful to the question-answer boundary, but it also means that the model does not verify claims against the external documents referenced by the dataset. In high-stakes settings, the next stage after this firewall should compare the answer against retrieved passages, database records, or policy documents.

The experiments use lightweight models and fixed features. Transformer classifiers, sentence embeddings, and calibrated neural rerankers can improve semantic coverage, but they also increase infrastructure requirements and can reduce interpretability. The present results therefore define a practical baseline and not an upper bound. The supplied code makes the split and metrics reusable for stronger models when additional compute is available.

The evaluation does not claim that TruthfulQA contains every form of hallucination. The benchmark focuses on imitative falsehoods and misconception-style questions. Enterprise hallucinations also include stale database values, fabricated citations, incorrect tool use, numerical mistakes, and policy violations. The firewall framework is compatible with those cases, but each case needs its own labeled data and evidence source. The present experiment establishes the measurement pipeline rather than a universal hallucination taxonomy.

The benchmark expansion treats every listed correct and incorrect answer as an independent candidate. This is appropriate for binary firewall training, but candidates from the same question are not semantically independent. The group-disjoint split prevents leakage across train and test, yet metric confidence intervals would still need clustered resampling by question for formal statistical inference. The present paper reports deterministic point estimates because the assignment emphasizes a reproducible experimental manuscript rather than hypothesis testing.

The detector scores answer candidates, not complete conversations. Multi-turn enterprise interactions add state, user intent shifts, tool outputs, and policy constraints. A production firewall for multi-turn use should include conversation history and retrieved evidence in the input representation. The current study establishes the single-turn candidate-answer baseline that such a system can build on.

The originally proposed HaluEval direction remains relevant for hallucination detection, but the complete raw HaluEval files could not be persisted reliably in this execution environment. The delivered empirical paper follows the assignment rule that permits a related dataset when the specified dataset is difficult to execute. The artifact package includes a HaluEval downloader script for an internet-enabled rerun and keeps all reported measurements restricted to the fully executed TruthfulQA experiment.

Conclusion

This paper presented a measured lightweight hallucination firewall for enterprise LLM applications. The full TruthfulQA CSV was converted into 5884 binary answer candidates, split by question ID, and evaluated with seven local detectors. The best detector, Word TF-IDF LR, reached AUROC 0.790 and F1 0.688 on held-out questions. These results demonstrate that a small CPU-friendly model can provide meaningful first-stage hallucination blocking without calling a large external judge.

For future work, the same pipeline can be extended in three concrete ways. First, retrieval passages can be added to the input so that evidence consistency is measured against documents rather than question-answer overlap alone. Second, transformer embeddings can be added as a middle-cost option between sparse TF-IDF and a full LLM judge. Third, production logs can be used to train domain-specific thresholds and monitor drift. These extensions preserve the central requirement that every reported table must be computed from executable data.

The experiment also demonstrates the importance of honest reporting. When a dataset cannot be fully executed in a constrained environment, replacing missing measurements with illustrative values creates a paper that cannot be reviewed or reproduced. This manuscript instead reports only measured results from the dataset that was fully downloaded and processed, and it provides code for rerunning the originally targeted dataset when the raw files are available.

The firewall is most useful as part of a layered reliability architecture. It accepts low-risk answers, blocks or escalates high-risk answers, logs interpretable signals, and preserves expensive verification resources for uncertain cases. Its limitations are clear: it does not replace retrieval, domain-specific evidence, or human review for high-stakes decisions. Its value is equally clear: it gives enterprises a reproducible, low-cost, and auditable baseline for reducing hallucination exposure before a generated answer reaches a user.

References

[1] S. Lin, J. Hilton, and O. Evans, "TruthfulQA: Measuring how models mimic human falsehoods," arXiv:2109.07958, 2021.

[2] Binghua Zhou, Siming Zhao, and David Chao, "LLM-Guided Energy-Aware A/B Testing for Consolidation and DVFS Policies via Power-Sensitivity Clustering", JACS, vol. 3, no. 4, pp. 12–30, Apr. 2023, doi: 10.69987/JACS.2023.30402.

[3] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, "FEVER: A large-scale dataset for fact extraction and verification," in Proc. NAACL-HLT, 2018, pp. 809-819.

[4] A. Williams, N. Nangia, and S. R. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," in Proc. NAACL-HLT, 2018, pp. 1112-1122.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. NAACL-HLT, 2019, pp. 4171-4186.

[6] Y. Liu et al., "RoBERTa: A robustly optimized BERT pretraining approach," arXiv:1907.11692, 2019.

[7] P. He, X. Liu, J. Gao, and W. Chen, "DeBERTa: Decoding-enhanced BERT with disentangled attention," arXiv:2006.03654, 2020.

[8] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in Proc. EMNLP-IJCNLP, 2019, pp. 3982-3992.

[9] Meng-Ju Kuo, Boning Zhang, and Haozhe Wang, "Tokenized Flow-Statistics Encrypted Traffic Analysis: Comparative Evaluation of 1D-CNN, BiLSTM, and Transformer on ISCX VPN-nonVPN 2016 (A1+A2, 60 s)", JACS, vol. 3, no. 8, pp. 39–53, Aug. 2023, doi: 10.69987/JACS.2023.30804.

[10] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825-2830, 2011.

[11] C. Cortes and V. Vapnik, "Support-vector networks," Mach. Learn., vol. 20, no. 3, pp. 273-297, 1995.

[12] L. Breiman, "Random forests," Mach. Learn., vol. 45, no. 1, pp. 5-32, 2001.

[13] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in Advances in Large Margin Classifiers, 1999, pp. 61-74.

[14] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. Cambridge, U.K.: Cambridge Univ. Press, 2008.
[15] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in Proc. EMNLP, 2016, pp. 2383-2392.
[16] Yunhe Li, "Execution-Feedback and Retrieval-Augmented Generation for Conversational Text-to-SQL: From One-Shot Questions to Clarification-Driven Executable Dialogs", JACS, vol. 3, no. 2, pp. 1–17, Feb. 2023, doi: 10.69987/JACS.2023.30201.
[17] S. Welleck, I. Kulikov, S. Roller, E. Dinan, K. Cho, and J. Weston, "Neural text generation with unlikelihood training," in Proc. ICLR, 2020.
[18] J. Maynez, S. Narayan, B. Bohnet, and R. McDonald, "On faithfulness and factuality in abstractive summarization," in Proc. ACL, 2020, pp. 1906-1919.
[19] W. Kryscinski, B. McCann, C. Xiong, and R. Socher, "Evaluating the factual consistency of abstractive text summarization," in Proc. EMNLP, 2020, pp. 9332-9346.
[20] O.-M. Camburu, T. Rocktäschel, T. Lukasiewicz, and P. Blunsom, "e-SNLI: Natural language inference with natural language explanations," in Proc. NeurIPS, 2018, pp. 9539-9549.
[21] V. Karpukhin et al., "Dense passage retrieval for open-domain question answering," in Proc. EMNLP, 2020, pp. 6769-6781.
[22] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, "REALM: Retrieval-augmented language model pre-training," in Proc. ICML, 2020, pp. 3929-3938.
[23] Jinyi Mu, Yifei Lu, and Michelle Smith, "LLM-Assisted Incrementality (Uplift) Modeling for Email Advertising: From Feature Interactions to Interpretable Audience–Creative–Channel Policies ", JACS, vol. 3, no. 1, pp. 31–48, Jan. 2023, doi: 10.69987/JACS.2023.30103.
[24] M. T. Ribeiro, T. Wu, C. Guestrin, and S. Singh, "Beyond accuracy: Behavioral testing of NLP models with CheckList," in Proc. ACL, 2020, pp. 4902-4912.
[25] D. Hendrycks et al., "Measuring massive multitask language understanding," in Proc. ICLR, 2021.