© AUG 2024 | IRE Journals | Volume 8 Issue 2 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV8I2-1722808

Detecting Prompt Injection in Retrieval-Augmented Generation Pipelines Using Lightweight Classifiers

SRIKUMAR NAYAK[1], TANYA JANE[2]
[1]*Department of Artificial Intelligence, LTIMindtree Research, NYC, USA*
[2]*Data Science Team, IBM Corporation, Austin, TX, USA*

Abstract*— Retrieval-Augmented Generation (RAG) grounds large language model (LLM) outputs in externally retrieved documents, but this same mechanism creates a novel attack surface: adversarial instructions embedded inside retrieved content can hijack the downstream model's behavior, a threat known as indirect prompt injection. Unlike classical adversarial machine learning, which perturbs input features to evade a classifier, indirect prompt injection exploits the inability of an LLM to structurally distinguish trusted developer instructions from untrusted retrieved data. This paper proposes and evaluates a lightweight, model-agnostic defense: a supervised classifier that screens retrieved passages for injected instructions before they reach the generation model's context window. We construct a synthetic, multi-domain RAG corpus spanning banking, healthcare, life sciences, general technology, and customer-support retrieval scenarios, and train three lightweight classifiers — logistic regression, a calibrated linear support vector machine, and a random forest — on TF-IDF features. On a held-out test set the best classifier achieves near-perfect detection (F1 = 1.000, AUC = 1.000). Critically, we then evaluate generalization against an adaptive-obfuscation test set using homoglyph substitution, payload splitting, and unseen synonym paraphrasing not present in training; recall degrades to 89.5–92.0% while precision remains at 100%, revealing an asymmetric robustness gap between naive and adaptive attackers. We further quantify the false-positive-rate/usability tradeoff via a decision-threshold sweep, showing how operators can tune the detector for high-recall security-critical deployments (e.g., banking transaction assistants) versus low-friction deployments (e.g., customer support). We discuss deployment implications for banking, healthcare, and life-sciences RAG systems, where both missed injections and false blocks carry asymmetric operational costs, and outline future work on adversarially robust training and cross-lingual obfuscation.

Index Terms*— Prompt injection, retrieval-augmented generation, large language models, adversarial machine learning, intrusion detection, TF-IDF, lightweight classifiers, LLM security, AI safety, banking security, healthcare AI, life sciences AI.

I. INTRODUCTION

Large language models (LLMs) augmented with retrieval — Retrieval-Augmented Generation, or RAG — have become the dominant architecture for grounding generative AI in enterprise knowledge bases, regulatory documents, and live operational data [1]. By fetching relevant passages at inference time and inserting them into the model's context window, RAG reduces hallucination and allows deployment without full model retraining. However, this design introduces a structural vulnerability: the LLM has no reliable mechanism to distinguish an instruction issued by the system designer from an instruction that merely appears inside retrieved data [2], [3]. An attacker who can influence any document a retriever might surface — a web page, a support ticket, a PDF, an email — can embed natural-language instructions that the model may follow as if they came from its operator. This class of attack, termed indirect prompt injection, has been demonstrated against real LLM-integrated applications, including cases where retrieved content coerced an assistant into exfiltrating data or issuing unauthorized actions [3].

Indirect prompt injection differs fundamentally from classical adversarial machine learning. Where adversarial perturbation attacks exploit a classifier's decision boundary through imperceptible input changes [4], prompt injection exploits a semantic and architectural gap: language models process instructions and data through the same channel. This makes many conventional ML-security defenses — adversarial training against gradient-based perturbations, input sanitization for numerical features — poorly matched to the problem, since the attack payload is natural language, not a crafted numerical perturbation.

This paper investigates whether a lightweight, pre-generation screening classifier can serve as a practical first line of defense in RAG pipelines. Such

a classifier sits between the retriever and the generator: every retrieved passage is scored for injection likelihood before being concatenated into the model's prompt. Because RAG pipelines are latency-sensitive and often run at high query volume, we deliberately restrict our study to lightweight, classical machine learning classifiers over sparse lexical features, rather than a second LLM-based judge, which would add substantial latency and cost.

A. Contributions

(1) We formalize the retrieval-time injection screening problem and propose a lightweight classifier architecture suitable for inline RAG deployment. (2) We construct a multi-domain synthetic corpus spanning banking, healthcare, life sciences, technology, and customer-support retrieval contexts, including hard-negative examples that reuse injection-adjacent vocabulary in benign contexts. (3) We empirically evaluate three lightweight classifiers and show near-perfect in-distribution detection but a measurable recall gap under adaptive obfuscation. (4) We quantify the false-positive/usability tradeoff via threshold analysis and discuss sector-specific deployment implications for banking, healthcare, and life sciences.

II. RELATED WORK

A. Retrieval-Augmented Generation

RAG was introduced as a method for combining a parametric language model with a non-parametric retrieval index, improving factual grounding on knowledge-intensive tasks [1]. Follow-on transformer architectures such as attention-based sequence models [5] and pretrained encoders [6] underpin most modern retrievers and generators, and open-source toolkits [7] have made RAG pipelines broadly accessible to practitioners, which has in turn widened the population of deployed systems exposed to retrieval-borne attacks.

B. Prompt Injection and LLM-Specific Threats

Early red-teaming work catalogued techniques for overriding a language model's system instructions through crafted user input [2]. Subsequent work extended this to the indirect setting, showing that instructions embedded in third-party content retrieved by an LLM-integrated application — search results, documents, emails — can hijack application behavior without any direct interaction between attacker and victim [3]. This indirect setting is the specific threat model addressed by our classifier, since the injected content arrives through the retrieval channel rather than the user-input channel.

C. Adversarial Machine Learning

Adversarial example research established that small, often imperceptible input perturbations can flip a classifier's decision [4], motivating a large body of work on robust training and certified defenses. While our threat model is linguistic rather than perturbation-based, the underlying tension — a defense that performs well against known attack patterns but degrades against adaptive, unseen ones — mirrors long-standing findings in adversarial robustness research, which we use to motivate our adaptive-obfuscation evaluation.

D. Anomaly and Intrusion Detection with Lightweight Models

Outside the LLM context, lightweight anomaly detectors such as isolation forest [8] have been widely used for unsupervised outlier detection in security telemetry [9], and classical supervised classifiers remain competitive for text-based abuse detection, including phishing and spam filtering from URL and content features [10]. Scikit-learn [11] and comparable toolkits have made such lightweight classifiers a practical default for latency-constrained security screening, which motivates our choice of TF-IDF plus classical classifiers over a second heavyweight LLM judge. Foundational network intrusion detection work established that statistical and machine-learning approaches to traffic classification face a persistent gap between laboratory and deployed performance [12], while signature-based systems such as Snort [13] and later statistical language-model approaches to unknown-attack detection [14] represent the earlier, non-learned baseline that ML-based detectors were designed to improve upon. Surveys of data-mining methods for intrusion detection [15] and empirical studies of deep learning's effectiveness for cybersecurity tasks [16] similarly report that lightweight, interpretable models often match deep architectures on tabular or lexical security-telemetry tasks at a fraction of the inference cost — a finding consistent with our own choice of classical classifiers for RAG-passage screening.

E. Malware and Behavioral Detection

Machine-learning-based malware analysis has a long history of using behavioral and lexical features rather than static signatures, including early work extracting n-gram and API-call features for automatic behavior classification [17] and later convolutional approaches operating directly on raw binary features [18]. These lines of work established that lightweight lexical or n-gram representations, similar to the TF-IDF features used here, can be highly effective for text- and sequence-based security classification tasks even without deep architectures.

F. Adversarial Robustness, Poisoning, and Backdoors

Beyond the original adversarial-example findings [4], subsequent work showed that adversarial perturbations generalize across model architectures and input domains [19], that adversarially crafted inputs can evade detection systems entirely [20], and that certified or empirically robust defenses remain difficult to achieve without significant utility cost [21]. A related and more directly relevant thread examines training-time attacks: data poisoning and backdoor ("trojan") attacks that implant hidden triggers into a model during training so that the model behaves normally on clean inputs but misbehaves on attacker-chosen triggers [22], [23]. This training-time threat model is conceptually distinct from, but structurally analogous to, the retrieval-time injection threat studied in this paper: in both cases, an adversary manipulates content the model will process as though it were trustworthy.

G. Model Extraction, Inversion, and Membership Inference

A separate body of adversarial ML work treats the deployed model itself as the attack surface, showing that an adversary with only query access can extract a functionally equivalent copy of a proprietary model [24], reconstruct sensitive training-data attributes through model inversion [25], or infer whether a specific record was part of a model's training set via membership inference [26]. These attacks target model confidentiality and training-data privacy rather than inference-time behavior hijacking, but they motivate the broader observation that ML systems, including RAG pipelines, expose multiple distinct attack surfaces that require independent defenses.

H. Federated and Privacy-Preserving Learning

Federated learning was proposed to allow collaborative model training across organizations without centralizing raw data [27], with subsequent surveys cataloguing open problems in communication efficiency, robustness to malicious participants, and privacy guarantees [28]. Although this paper does not train its detector in a federated setting, federated threat-intelligence sharing is a natural extension for organizations that wish to pool labeled injection examples without exposing proprietary retrieval corpora, and we discuss this further in Section VIII.

I. Explainable AI for Security Operations

Model-agnostic explanation methods such as LIME [29] and SHAP [30] were developed to make individual classifier decisions interpretable to human reviewers, a capability increasingly expected of security-operations tooling so that analysts can audit why a given alert fired rather than treating the classifier as an opaque gate. We adopt this motivation in Section VIII when discussing token-level attribution as future work for the proposed detector.

J. Deepfake and Synthetic-Media Detection

Generative adversarial networks [31] enabled a rapid improvement in synthetic image and video quality, which in turn motivated dedicated forensic datasets and detectors for manipulated facial video [32]. Although outside this paper's text-only scope, deepfake detection shares the same underlying pattern as RAG-injection detection: a lightweight, purpose-built classifier attempting to catch content engineered specifically to evade or deceive a downstream consumer, whether human or model.

K. Text Representation and Classical Classifiers

The TF-IDF weighting scheme used in this paper's feature extraction traces to classical information-retrieval term-weighting research [33], and support-vector and ensemble classifiers over sparse lexical features were shown early on to be highly effective for text categorization [34], predating the deep-learning era and remaining a strong, low-latency baseline for text classification tasks such as the one studied here. Random forests [35], the foundation of one of our three evaluated classifiers, and support-vector machines [36] more generally remain standard, well-understood baselines against which more complex architectures are typically compared.

L. Gap Addressed by This Work

© AUG 2024 | IRE Journals | Volume 8 Issue 2 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV8I2-1722808

Prior work either documents the existence of indirect prompt injection as an attack [2], [3] or addresses adversarial robustness, poisoning, extraction, or privacy risks in the general ML sense [4], [19]-[26], but empirical evaluation of lightweight, inline classifiers purpose-built for RAG-passage screening — including their behavior under adaptive obfuscation and their false-positive cost to downstream usability — remains comparatively underexplored. This paper targets that gap.

III. THREAT MODEL AND PROBLEM FORMULATION

A. System Model

Let D denote a document store and q a user query. A dense retriever encodes queries and documents through an embedding function phi and returns the top-k passages by cosine similarity:

$$\mathcal{R}_k(q) = \arg\max_{d \in D}^{(k)} \mathrm{sim}(\phi(q), \phi(d)) \tag{1}$$

$$\mathrm{sim}(u, v) = \frac{u^\top v}{\|u\|_2 \, \|v\|_2} \tag{2}$$

The retrieved set, a system instruction s, and the query are serialized into a single context window C, which the generative model M consumes to produce the response y, where the operator $\oplus$ denotes ordered concatenation:

$$C = s \oplus d_1 \oplus d_2 \oplus \cdots \oplus d_k \oplus q, \qquad y = \mathcal{M}(C) \tag{3}$$

Crucially, M processes s (trusted) and each d_i (untrusted, attacker-influenceable) through the same token channel, with no architectural type distinction between instruction and data — the structural root cause of indirect prompt injection.

B. Adversary Model

We assume an adversary who cannot access model weights or the system prompt but can publish or modify content likely to be indexed and later retrieved. The adversary applies an injection transform psi that embeds an instruction payload iota, drawn from an attack family I, into an otherwise benign passage d. The adversary's objective is to maximize the probability that the model's output falls in an adversarial target set Y_adv, subject to a stealth constraint that the perturbed passage remains within a bounded semantic distance Delta of the original so as to survive retrieval and evade cursory human inspection:

$$\max_{\iota \in I} \Pr[\mathcal{M}(s \oplus \psi(d, \iota) \oplus q) \in \mathcal{Y}_{\mathrm{adv}}] \quad \text{s.t. } \Delta(\psi(d, \iota), d) \le \varepsilon \tag{4}$$

This formulation makes explicit that prompt injection is an inference-time integrity attack on the context window, distinct from training-time poisoning [22], [23] or perturbation attacks on a classifier's own inputs [4].

C. Detection Objective

The defender introduces a screening function f_theta that maps a candidate passage's feature representation x(d) to an injection probability and applies a decision threshold tau to obtain a hard label; the indicator 1[·] returns one when its predicate holds:

$$f_\theta : \mathcal{X} \to [0, 1], \qquad \hat{y}(d) = \mathbf{1}[f_\theta(x(d)) \ge \tau] \tag{5}$$

Learning seeks the parameters theta minimizing the expected risk under the data distribution P, where l is a surrogate loss (log-loss for the probabilistic models, hinge loss for the SVM):

$$\mathcal{R}(f_\theta) = \mathbb{E}_{(x, y) \sim \mathcal{P}}[\ell(f_\theta(x), y)] \tag{6}$$

D. Feature Representation

Each passage is mapped to a sparse lexical vector over a vocabulary V of word unigrams and bigrams. We use sublinear term-frequency scaling to damp the effect of repeated tokens, smoothed inverse document frequency to avoid division by zero for unseen terms, and L2 normalization so that passage length does not bias the decision boundary:

$$\widetilde{\mathrm{tf}}(t, d) = (1 + \log \mathrm{tf}(t, d)) \cdot \mathbf{1}[\mathrm{tf}(t, d) > 0] \tag{7}$$

$$\mathrm{idf}(t) = \log \frac{1 + N}{1 + \mathrm{df}(t)} + 1 \tag{8}$$

$$x_t(d) = \frac{\widetilde{\mathrm{tf}}(t, d) \cdot \mathrm{idf}(t)}{\sqrt{\sum_{t' \in V} (\widetilde{\mathrm{tf}}(t', d) \cdot \mathrm{idf}(t'))^2}} \tag{9}$$

Here tf(t,d) is the raw count of term t in passage d, df(t) is the number of passages containing t, and N is the corpus size. The resulting unit-norm vector x(d) = [x_t(d)]_{t in V} is the common input to all three classifiers, is computable in time linear in passage length, and requires no GPU inference — a deliberate design choice for inline, latency-bounded RAG screening.

E. Logistic Regression Detector

The logistic model posits a linear log-odds boundary in the TF-IDF space, with the sigmoid link function sigma mapping scores to calibrated probabilities:

$$P_\theta(y = 1 \mid x) = \sigma(w^\top x + b), \qquad \sigma(z) = \frac{1}{1 + e^{-z}} \tag{10}$$

With labels encoded as y_i in {-1, +1}, parameters are obtained by minimizing the L2-regularized negative log-likelihood, a smooth convex objective admitting a unique global optimum:

$$\min_{w,b} J(w,b)=\frac{\lambda}{2}\|w\|_2^2 + \sum_{i=1}^{n} \log\left(1+\exp\left(-y_i(w^\top x_i + b)\right)\right) \tag{11}$$

where lambda governs the strength of regularization (equivalently C = 1/lambda). The gradient with respect to the weight vector, used by the quasi-Newton solver, is

$$\nabla_w J = \lambda w - \sum_{i=1}^{n} y_i x_i \sigma(-y_i(w^\top x_i + b)) \tag{12}$$

which shows each training passage contributes in proportion to how confidently it is currently misclassified.

F. Support Vector Machine Detector

The linear SVM seeks the maximum-margin separating hyperplane, trading margin width against slack variables xi_i that permit soft-margin violations for the non-separable case:

$$\min_{w,b,\xi} \tfrac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{n}\xi_i \quad \text{s.t. } y_i(w^\top x_i + b) \ge 1-\xi_i,\ \xi_i \ge 0 \tag{13}$$

Solving the corresponding Lagrangian dual expresses the decision boundary purely through inner products of support vectors, with dual coefficients alpha_i bounded by the penalty C:

$$\max_{\alpha} \sum_{i=1}^{n}\alpha_i - \tfrac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^\top x_j \quad \text{s.t. } 0\le\alpha_i\le C,\ \sum_i \alpha_i y_i = 0 \tag{14}$$

Because the SVM decision function returns unbounded margin scores rather than probabilities, we apply Platt scaling, fitting a one-parameter-pair logistic transform on held-out folds so that the SVM emits comparable, threshold-tunable probabilities:

$$P(y=1 \mid g(x)) = \frac{1}{1+\exp(A\,g(x)+B)}, \qquad g(x) = w^\top x + b \tag{15}$$

G. Random Forest Detector

The random forest grows an ensemble of B decision trees on bootstrap resamples with randomized feature subsets at each split. Splits are chosen to maximize the impurity decrease under the Gini criterion, where p_c is the proportion of class c in a node S partitioned into children S_L and S_R:

$$G(S) = 1 - \sum_{c\in\{0,1\}} p_c^2, \qquad \Delta G = G(S) - \frac{|S_L|}{|S|}G(S_L) - \frac{|S_R|}{|S|}G(S_R) \tag{16}$$

The ensemble's injection probability is the average of the individual tree estimates, which reduces variance relative to any single tree:

$$f_{\mathrm{RF}}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x;\Theta_b), \qquad \Theta_b \sim \mathrm{Bootstrap}(\mathcal{D}_{\mathrm{train}}) \tag{17}$$

H. Evaluation Metrics

Detection quality is summarized by precision, recall (equivalently the true-positive rate), and the false-positive rate, defined over the confusion-matrix counts TP, FP, TN, FN:

$$\mathrm{Pr} = \frac{TP}{TP+FP}, \quad \mathrm{Re} = \mathrm{TPR} = \frac{TP}{TP+FN}, \quad \mathrm{FPR} = \frac{FP}{FP+TN} \tag{18}$$

The F-beta score harmonizes precision and recall with a tunable emphasis beta, reducing to the balanced F1 at beta = 1; a security-critical deployment favoring recall would set beta > 1:

$$F_\beta = \frac{(1+\beta^2)\cdot \mathrm{Pr}\cdot \mathrm{Re}}{\beta^2\,\mathrm{Pr} + \mathrm{Re}}, \qquad F_1 = F_\beta \text{ at } \beta = 1 \tag{19}$$

The threshold-independent area under the ROC curve admits the probabilistic interpretation as the chance a random injected passage is scored above a random clean one:

$$\mathrm{AUC} = \int_0^1 \mathrm{TPR}(\mathrm{FPR}^{-1}(u))du = \Pr[f_\theta(x^+) > f_\theta(x^-)] \tag{20}$$

I. Cost-Sensitive Threshold Optimization

Because the operational costs of the two error types are asymmetric and sector-dependent, we treat threshold selection as explicit risk minimization. Let pi_0 and pi_1 be the clean and injected class priors, alpha the cost of a false positive (a blocked legitimate passage) and beta the cost of a false negative (a missed injection). The expected misclassification cost as a function of the threshold is

$$C(\tau) = \alpha\,\pi_0\,\mathrm{FPR}(\tau) + \beta\,\pi_1\,\mathrm{FNR}(\tau), \qquad \pi_c = \Pr[y=c] \tag{21}$$

For a well-calibrated probabilistic detector, the cost-minimizing threshold has the closed form given by the Bayes decision rule, making explicit how a higher penalty on missed injections beta drives the optimal threshold downward toward higher recall:

$$\tau^* = \arg\min_{\tau\in[0,1]} C(\tau) \implies \tau^* = \frac{\alpha\,\pi_0}{\alpha\,\pi_0 + \beta\,\pi_1} \tag{22}$$

Section VII instantiates alpha and beta qualitatively for banking, healthcare, and life-sciences deployments; Section VI-C reports the empirical threshold sweep.

J. Detection and Threshold-Tuning Procedures

Algorithm 1 specifies the end-to-end inline screening procedure applied to each retrieval result before context assembly; Algorithm 2 specifies the offline cost-sensitive threshold calibration that produces tau* from a labeled validation set.

Algorithm 1 Inline RAG Injection Screening

```
Require: query q, store D,
retriever R_k, threshold tau,
            fitted vectorizer V,
detector f_theta, model M
```

© AUG 2024 | IRE Journals | Volume 8 Issue 2 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV8I2-1722808

```
Ensure:  grounded response y or
review escalation
1:  Dk <- R_k(q)                  //
top-k retrieval
2:  A <- {}                       //
admitted passages
3:  for each d in Dk do
  4:      x <- V.transform(d)
  // Eq. (7)-(9)
  5:      s <- f_theta(x)
  // injection prob.
  6:      if s >= tau then
    7:          flag d; route to
    human review
  8:      else
    9:          A <- A union {d}
  10:     end if
11: end for
12: C <- s_sys ⊕ A ⊕ q
// Eq. (3)
13: return y <- M(C)
```

Algorithm 2 Cost-Sensitive Threshold Calibration

```
Require: validation set {(x_i,
y_i)}, costs alpha, beta,
            priors pi_0, pi_1,
grid T subset [0,1]
Ensure:  cost-optimal threshold
tau*
1:  best <- +infinity;  tau* <-
0.5
2:  for each tau in T do
  3:      compute FPR(tau),
  FNR(tau)
  4:      C(tau) <-
  alpha*pi_0*FPR + beta*pi_1*FNR
  5:      if C(tau) < best then
    6:          best <- C(tau);
    tau* <- tau     // Eq.(21)
  7:      end if
8:  end for
9:  return tau*
// cf. Eq.(22)
```

IV. DATASET

Because no public, sector-diverse corpus of labeled RAG-retrieval-time injection attempts was available for this study, we constructed a synthetic corpus designed to approximate realistic retrieval passages across five domains relevant to enterprise RAG deployments (Table 1).

Domain	Retrieval Context Example
Banking / Finance	Loan policy excerpts, AML thresholds, transaction workflows
Healthcare	EHR procedure notes, discharge/triage guidance
Life Sciences	Assay protocols, trial and regulatory summaries
General Technology	API docs, infra and deployment notes
Customer Support	Policy FAQs, refund/escalation procedures

TABLE 1. Domains represented in the synthetic RAG retrieval corpus.

Clean passages combine two to three domain-appropriate factual sentence templates with randomized numeric parameters. To avoid a trivially separable dataset, 35% of clean passages additionally include a hard-negative sentence that reuses injection-adjacent vocabulary ("ignore," "override," "instruction," "delete," "admin") in an ordinary business context — for example, an IT policy statement about account deletion after a retention period. Injected passages combine one to two clean sentences with an injected instruction segment drawn from ten template patterns representative of reported indirect-injection techniques [2], [3]: role override, exfiltration requests, tool-call coercion, and encoded payloads. 55% of injected passages are additionally obfuscated using case variation, character spacing, or synonym substitution to mimic a mildly adaptive attacker. The corpus totals 3,000 labeled passages (1,500 clean / 1,500 injected), split 75/25 into training and held-out test sets. A separate 800-passage adaptive-obfuscation set, using homoglyph substitution, payload splitting across sentences, and synonym substitutions not present during training, is held out entirely for generalization testing (Table 2).

Split	Clean	Injected	Total
Training (75%)	1,125	1,125	2,250
Held-out test (25%)	375	375	750
Adaptive-obfusc. set	400	400	800

© AUG 2024 | IRE Journals | Volume 8 Issue 2 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV8I2-1722808

TABLE 2. Corpus composition and splits.

V. EXPERIMENTAL SETUP

All experiments use scikit-learn [11]. TF-IDF vectorization (Eqs. 7-9) uses word unigrams and bigrams, a minimum document frequency of 2, sublinear term-frequency scaling, and a 5,000-feature vocabulary cap. We evaluate three lightweight classifiers chosen for their low inference latency relative to a second LLM-based judge: L2-regularized logistic regression (Eqs. 10-12, C=1.0), a linear support vector machine (Eqs. 13-14, C=1.0) with Platt-scaling calibration (Eq. 15) for probability outputs, and a random forest (Eqs. 16-17; 200 trees, max depth 12). Models are trained on the 2,250-passage training split via Algorithm 1's feature pipeline and evaluated on (i) the 750-passage held-out standard test set and (ii) the 800-passage adaptive-obfuscation set, which uses obfuscation strategies absent from training to probe generalization against an adapting attacker.

VI. RESULTS AND ANALYSIS

A. Standard Test Set Performance

Table 3 reports performance on the held-out standard test set. All three classifiers achieve perfect separation (F1 = 1.000, AUC = 1.000), indicating that lexical cues from the ten injection templates and their moderate obfuscation are highly separable from clean, domain-specific retrieval passages even in the presence of hard-negative examples.

Model	Acc.	Prec.	Recall	F1	AUC	FPR
Logistic Regression	1.000	1.000	1.000	1.000	1.000	0.000
Linear SVM (cal.)	1.000	1.000	1.000	1.000	1.000	0.000
Random Forest	1.000	1.000	1.000	1.000	1.000	0.000

TABLE 3. Classifier performance on the standard held-out test set (n=750).

FIGURE 1. ROC curves for the three lightweight classifiers on the standard test set.

B. Generalization Under Adaptive Obfuscation

Table 4 reports performance on the adaptive-obfuscation set, which uses homoglyph substitution, mid-sentence payload splitting, and unseen synonym paraphrases. Recall drops to 89.5-92.0% across classifiers while precision remains at 100%, meaning the detectors never flag a clean passage as malicious but increasingly miss injected passages as obfuscation diverges from the training distribution. The calibrated linear SVM generalizes best (recall = 0.920), followed by logistic regression (0.903) and random forest (0.895).

Model	Acc.	Prec.	Recall	F1	AUC	FPR
Logistic Regression	0.951	1.000	0.903	0.949	1.000	0.000
Linear SVM (cal.)	0.960	1.000	0.920	0.958	1.000	0.000
Random Forest	0.948	1.000	0.895	0.945	1.000	0.000

TABLE 4. Classifier performance on the adaptive-obfuscation set (n=800, unseen obfuscation strategies).

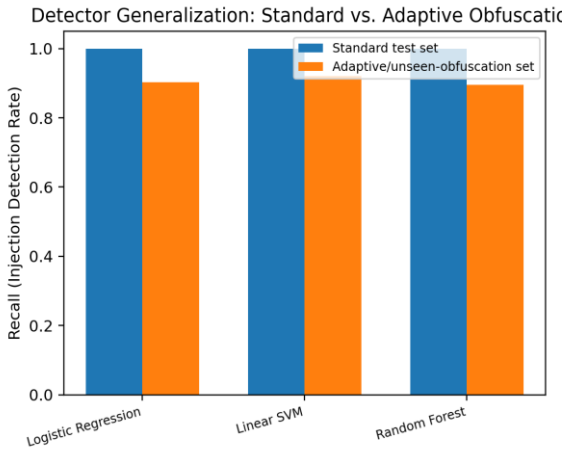

FIGURE 2. Recall on standard vs. adaptive-obfuscation test sets, illustrating the generalization gap.

C. False-Positive / Usability Tradeoff

Figure 3 sweeps the decision threshold of the best-performing model (logistic regression, selected by standard-set AUC) and plots false-positive rate against false-negative rate. At low thresholds, near-zero false negatives are achievable at a nonzero false-positive cost, meaning some legitimate retrieved passages are blocked or flagged for review. Operators can select an operating point according to the relative cost of a missed injection versus a blocked legitimate document, a decision that we argue should be sector-specific (Section VIII).

© AUG 2024 | IRE Journals | Volume 8 Issue 2 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV8I2-1722808

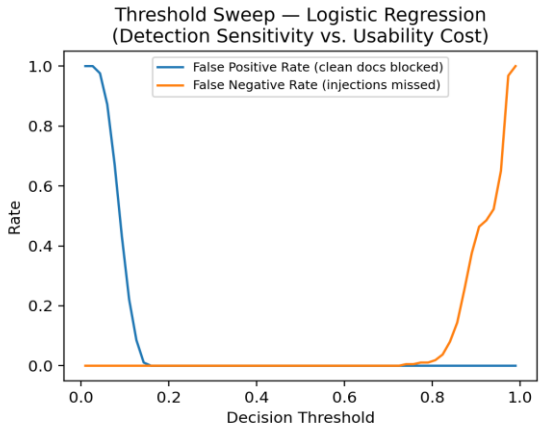

FIGURE 3. Threshold sweep showing the false-positive/false-negative tradeoff for the best-performing classifier.

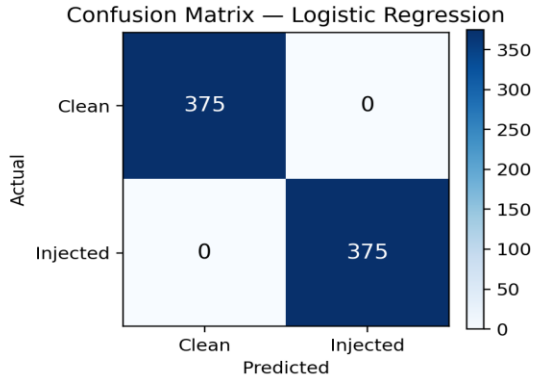

FIGURE 4. Confusion matrix for the best-performing classifier at the default 0.5 threshold on the standard test set.

D. Discussion of Findings

The near-perfect standard-set results should be interpreted cautiously: they demonstrate that lightweight lexical classifiers can reliably catch injection attempts resembling known patterns, which is a meaningful first line of defense, but the adaptive-set results show this reliability erodes as attackers diverge from known phrasing. The consistently perfect precision across both settings is operationally significant — it suggests the tested lightweight classifiers rarely alarm on benign, injection-adjacent business language, an important property for production usability.

VII. SECTOR-SPECIFIC DEPLOYMENT IMPLICATIONS

A. Banking and Financial Services

RAG-powered banking assistants that retrieve policy documents, transaction histories, or customer correspondence are attractive injection targets because a successful injection could coerce unauthorized fund transfers, disclosure of account data, or bypassed compliance checks. In this setting, the cost of a missed injection (beta in Section III-C) is severe and regulatory exposure (e.g., AML/KYC obligations) argues for operating the classifier at a low decision threshold, favoring recall over precision, with flagged passages routed to human review rather than silently blocked, preserving usability.

B. Healthcare

Clinical RAG systems retrieving from electronic health records or clinical guidelines face a different asymmetry: an injected instruction that alters triage guidance or suppresses a contraindication warning could directly affect patient safety, while an overly aggressive detector that blocks legitimate clinical text could delay care. This argues for a moderate threshold combined with a mandatory human-in-the-loop review step for any flagged clinical content, consistent with existing clinical decision-support oversight practices.

C. Life Sciences

In pharmaceutical and life-sciences settings, RAG systems retrieving from regulatory submissions, trial protocols, or lab data are exposed to injection risks that could corrupt reported results, alter dosage guidance, or leak proprietary compound data to external systems if a tool-call-coercion style payload succeeds. Because life-sciences retrieval corpora are often more static and internally curated than open-web sources, a moderately high threshold may be viable, but provenance tracking of ingested documents remains an important complementary control alongside the classifier.

D. Cross-Sector Observation

Across all three sectors, the classifier is best understood as a triage layer rather than a complete control: it reduces the volume of injected content reaching the LLM's context window and provides an auditable signal for compliance logging, but should be paired with least-privilege tool permissions for any agentic action the LLM can take, so that even an undetected injection cannot directly cause harm.

VIII. FUTURE WORK

(1) Adversarially robust training: incorporating obfuscated and paraphrased injection examples during training, potentially via adversarial data augmentation, to close the recall gap observed in Section VI-B. (2) Cross-lingual and multi-script obfuscation: extending evaluation to non-English retrieval corpora and mixed-script homoglyph attacks, which the present TF-IDF representation may be particularly vulnerable to. (3) Ensemble and cascade architectures: combining the lightweight classifier as a fast first-pass filter with a slower, more capable LLM-based judge only for borderline-confidence passages, to recover recall without paying full LLM-judge latency on every retrieved passage. (4) Real-world corpus validation: replacing the synthetic corpus with labeled data from production RAG deployments or red-team exercises, subject to appropriate data governance, to validate that synthetic-corpus findings transfer to operational settings. (5) Sector-specific threshold policy studies: formal cost-sensitive threshold optimization using the usability-cost function of Section III-C, parameterized with sector-specific alpha/beta estimates from banking, healthcare, and life-sciences stakeholders. (6) Explainability: extending the classifier with token-level attribution so that flagged passages can be reviewed efficiently by SOC analysts or compliance reviewers, consistent with explainable-AI practices for security operations.

IX. CONCLUSION

This paper evaluated lightweight, TF-IDF-based classifiers as an inline defense against indirect prompt injection in RAG pipelines. Across a synthetic, multi-domain corpus, the classifiers achieved near-perfect detection of injection patterns resembling their training distribution but showed a measurable recall gap of roughly 8-11 percentage points under adaptive obfuscation not seen during training, while precision remained perfect throughout. We argued that the false-positive/false-negative tradeoff exposed by threshold analysis should be tuned per deployment sector, with banking and healthcare RAG systems favoring recall-oriented thresholds paired with human review, and outlined a research agenda toward adversarially robust, cascade-based, and explainable detection architectures for production RAG security.

REFERENCES

[1] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Proc. NeurIPS, 2020, pp. 9459-9474.
[2] F. Perez and I. Ribeiro, "Ignore previous prompt: Attack techniques for language models," arXiv:2211.09527, 2022.
[3] K. Greshake et al., "Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection," arXiv:2302.12173, 2023.
[4] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in Proc. ICLR, 2015.
[5] A. Vaswani et al., "Attention is all you need," in Proc. NeurIPS, 2017, pp. 5998-6008.
[6] 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.
[7] T. Wolf et al., "Transformers: State-of-the-art natural language processing," in Proc. EMNLP: Syst. Demonstrations, 2020, pp. 38-45.
[8] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in Proc. IEEE ICDM, 2008, pp. 413-422.
[9] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Comput. Surv., vol. 41, no. 3, pp. 1-58, 2009.
[10] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," Expert Syst. Appl., vol. 117, pp. 345-357, 2019.
[11] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825-2830, 2011.
[12] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in Proc. IEEE Symp. Security and Privacy, 2010, pp. 305-316.
[13] M. Roesch, "Snort: Lightweight intrusion detection for networks," in Proc. USENIX LISA, 1999, pp. 229-238.
[14] K. Rieck and P. Laskov, "Language models for detection of unknown attacks in network traffic," J. Comput. Virology, vol. 2, no. 4, pp. 243-256, 2007.
[15] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," IEEE Commun.

Surveys Tuts., vol. 18, no. 2, pp. 1153-1176, 2016.

[16] G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, and M. Marchetti, "On the effectiveness of machine and deep learning for cyber security," in Proc. Int. Conf. Cyber Conflict (CyCon), 2018, pp. 371-390.

[17] K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning," J. Comput. Security, vol. 19, no. 4, pp. 639-668, 2011.

[18] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in Proc. Int. Conf. Malicious and Unwanted Software (MALWARE), 2015, pp. 11-20.

[19] C. Szegedy et al., "Intriguing properties of neural networks," in Proc. ICLR, 2014.

[20] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in Proc. IEEE Eur. Symp. Security and Privacy, 2016, pp. 372-387.

[21] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in Proc. IEEE Symp. Security and Privacy, 2017, pp. 39-57.

[22] T. Gu, B. Dolan-Gavitt, and S. Garg, "BadNets: Identifying vulnerabilities in the machine learning model supply chain," arXiv:1708.06733, 2017.

[23] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," arXiv:1712.05526, 2017.

[24] F. Tramer, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in Proc. USENIX Security Symp., 2016, pp. 601-618.

[25] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in Proc. ACM SIGSAC CCS, 2015, pp. 1322-1333.

[26] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in Proc. IEEE Symp. Security and Privacy, 2017, pp. 3-18.

[27] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in Proc. AISTATS, 2017, pp. 1273-1282.

[28] P. Kairouz et al., "Advances and open problems in federated learning," Foundations and Trends in Machine Learning, vol. 14, no. 1-2, pp. 1-210, 2021.

[29] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?': Explaining the predictions of any classifier," in Proc. ACM SIGKDD, 2016, pp. 1135-1144.

[30] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in Proc. NeurIPS, 2017, pp. 4765-4774.

[31] I. Goodfellow et al., "Generative adversarial networks," in Proc. NeurIPS, 2014, pp. 2672-2680.

[32] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Niessner, "FaceForensics++: Learning to detect manipulated facial images," in Proc. IEEE/CVF ICCV, 2019, pp. 1-11.

[33] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," Inf. Process. Manage., vol. 24, no. 5, pp. 513-523, 1988.

[34] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in Proc. ECML, 1998, pp. 137-142.

[35] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5-32, 2001.

[36] C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, vol. 20, no. 3, pp. 273-297, 1995.