

MnM: Real-Time Behavioral Detection of Prompt Injection via Reasoning Stream Monitoring

Nicolás David Galindo

Universidad Nacional de Colombia
Colombia
ngalindoc@unal.edu.co

Juan Jacobo Izquierdo

Universidad Nacional de Colombia
Colombia
juizquierdob@unal.edu.co

Daniel Libardo Diaz

Universidad Nacional de Colombia
Colombia
dandiazgo@unal.edu.co

David Andrés Ramírez

Pontificia Universidad Javeriana
Colombia
da-ramirez@javierana.edu.co

ABSTRACT

Prompt injection remains one of the most practical threats to LLM-based agents, yet most detectors require retraining whenever a new attack pattern or a new language appears. This paper presents **MnM** (Monitor and Mitigate), a dual-layer detection system that replaces gradient-based training with retrieval over a vector database. Layer 1 classifies the incoming prompt by running k -nearest-neighbor retrieval over a database of labeled prompt chunks. Layer 2 classifies the model’s reasoning stream as it is produced, using the same retrieval pipeline against a separate database of compromised and clean reasoning patterns. Both layers share one multilingual embedding model (qwen3-embedding-8b, 4096-d) and one chunking model (mistral-nemo), and their verdicts are fused with a logical OR. We evaluate MnM on 200 English prompts (Gandalf Ignore Instructions + ShareGPT) and 200 Hindi/Hinglish prompts, with the vector databases built from English data only. On English, the best operating point ($k=4$, $\theta=0.7$) reaches 84.5% combined accuracy and 0.86 F1 with 99% recall. On Hindi, where no target-language data exists in the database, the best point ($k=5$, $\theta=0.5$) reaches 77.5% combined accuracy and 0.78 F1. When a small batch of Hindi reference embeddings (10% of the evaluation set) is seeded into the database, the prompt lane accuracy rises from 63.0% to 71.1% and combined F1 rises from 0.78 to 0.79, showing that the retrieval-based architecture can close the cross-lingual gap without retraining.

KEYWORDS

prompt injection, behavioral detection, reasoning stream monitoring, chain-of-thought, LLM agents, multilingual, vector database, AI safety

1 INTRODUCTION

Large language models are increasingly deployed as autonomous agents that browse the web, execute code, and invoke external tools on behalf of users. This agentic setting introduces a qualitatively new attack surface: *prompt injection*, in which adversarial instructions embedded in the environment (a malicious webpage, a crafted document, a poisoned API response) redirect the agent away from its original task and toward goals chosen by the attacker [3, 4]. Unlike jailbreaks that target the model at inference time through the user turn, prompt injection exploits the agent’s

trusted perception of its environment, making it difficult to defend against at the input level alone.

Recent empirical work has established that the threat is not confined to English-language deployments. Burbano et al. [2] demonstrate that indirect prompt injection can reliably hijack embodied agents across English, Chinese, Spanish, and Spanglish, revealing that multilingual capability, one of the most valued properties of modern LLMs, doubles as an attack amplifier. At the architectural level, Guo et al. [5] identify instruction-hierarchy robustness as an open problem: agents trained to follow user instructions do not reliably distinguish between legitimate principal instructions and injected adversarial ones, especially when the injection is phrased in an out-of-distribution style for the hierarchy.

Existing detection approaches address prompt injection primarily through input classification or reasoning probes, but share a common limitation: adapting to a new attack pattern or a new language requires retraining or fine-tuning the detector itself [6, 7, 9]. This rigidity is costly in the multilingual agentic setting, where attack payloads transfer across languages [2] far faster than labeled defense data can be collected. Additionally, any approach that commits to a decision before generation begins cannot catch an injection that is syntactically benign at the prompt level but succeeds by gradually steering the model’s chain-of-thought toward a compromised goal.

We propose **MnM** (*Monitor and Mitigate*), a dual-layer detection system designed to address both failure modes without gradient-based training. The *prompt classifier* (Layer 1) operates on the incoming prompt: it uses a lightweight language model to extract semantically meaningful chunks, embeds each chunk with a high-dimensional multilingual embedding model, and issues a label via k -nearest-neighbor vote against a vector database of known malicious and benign examples. Because detection knowledge lives directly in embedding space, updates are immediate: a small set of newly labeled examples for an unseen attack pattern or a new language can be inserted into the database with no retraining cycle. The classifier bootstraps from existing injection datasets without requiring manual annotation of model behavior.

The *behavioral classifier* (Layer 2) activates at generation time. As the LLM streams its chain-of-thought, MnM chunks the reasoning, embeds each chunk, and compares it against a vector database of known compromised and clean reasoning patterns. A streaming gate aggregates per-chunk scores and issues an early-stop signal

the moment the reasoning drifts toward injected behavior, before the model has produced any harmful output or tool call. Both layers share the same embedding infrastructure and the same training-free philosophy: the behavioral database is built from generated traces, not from human-annotated reasoning steps. Together, they provide two independent detection signals, one at the instruction boundary and one inside the reasoning process, evaluated here on English and Hindi prompts. This extends the multilingual attack surface documented by Burbano et al. [2] to the detector side.

2 STATE OF THE ART

Benchmarking and threat taxonomy. AgentDojo [3] is the standard dynamic benchmark for evaluating prompt injection attacks and defenses in LLM agent pipelines. It provides a suite of realistic tasks across web, file, and calendar environments, and is used in this work to situate our threat model. Ferrag et al. [4] extend the taxonomy beyond classic prompt injection to protocol-level exploits in multi-agent workflows, establishing that the attack surface grows substantially as agents compose and delegate to one another.

Multilingual and agentic attack surface. Burbano et al. [2] show that indirect prompt injection transfers across four languages in embodied agent settings, with success rates that do not degrade significantly in non-English targets. This motivates our multilingual evaluation: a detector that only generalizes to English provides false assurance in the deployment settings that are already under attack. Guo et al. [5] frame the underlying failure as an instruction-hierarchy problem, where agents cannot reliably rank the authority of instructions arriving from different principals, and propose a challenge dataset to measure robustness. MnM addresses a complementary gap: rather than enforcing hierarchy at the policy level, it detects when the model’s own reasoning has already been captured by an injected goal.

Input-level detection. Srinivasan et al. [8] analyze prompt injection detection across Indian multilingual LLMs, finding that cross-lingual transfer in detectors lags behind transfer in attacker payloads, a gap MnM is designed to close by operating on embedding space rather than token-level heuristics. Sato et al. [6] demonstrate that in-context exemplars of injected prompts meaningfully improve detection without fine-tuning, establishing a strong zero-shot baseline. Yu et al. [9] apply fuzzing to generate adversarial prompt variants and evaluate detector robustness, providing a methodology for stress-testing defenses beyond fixed benchmark sets.

Reasoning-aware and RAG-aware detection. Shi et al. [7] propose Meticulous Thought Defender, which uses fine-grained chain-of-thought analysis to identify injection signals in model reasoning. MnM differs in that it operates *online* during the streaming generation rather than post-hoc, enabling early stopping before harmful output is produced. Al Masoud et al. [1] address a related threat in retrieval-augmented generation pipelines: single-turn prompt-leaking attacks that extract system context through crafted queries. SD-RAG’s selective-disclosure framing informs our understanding of what a compromised reasoning trajectory looks like when the model is manipulated into revealing or acting on injected content.

3 APPROACH

The design of **MnM** follows from two practical constraints. The detector has to keep up with new attack patterns and new target languages without going through a retraining cycle, because the multilingual attack surface documented in the literature moves faster than labeled defense data can be collected. Additionally, the detector cannot commit to a verdict before the model has begun to reason, because a non-trivial fraction of injections look benign at the prompt level and only reveal themselves once the model is several steps into its chain-of-thought. These two constraints point toward a retrieval-based architecture rather than a fine-tuned classifier. If the detection knowledge lives in a vector database, then adding a new attack family or a new language reduces to inserting a small batch of labeled examples, with no gradient updates required. MnM applies this idea twice, with two independent classifiers that share embedding infrastructure but operate on different signals. The first runs on the incoming prompt before generation begins. The second runs on the model’s reasoning stream as it is produced. The two verdicts are combined at the end into a single alarm. Figure 1 shows the overall architecture.

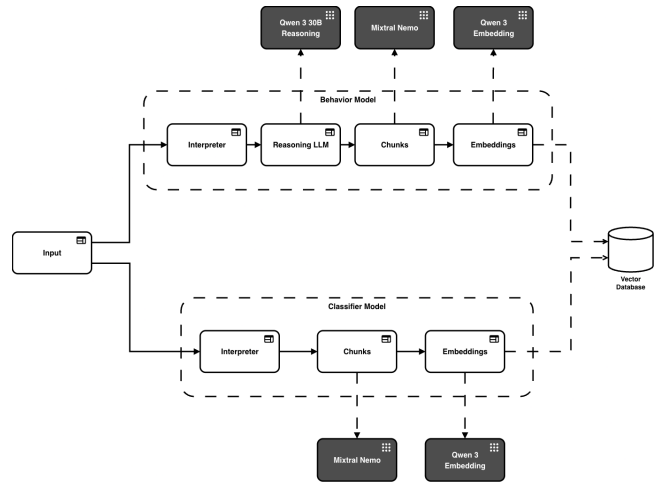


Figure 1: MnM architecture. The input prompt is processed by both the Behavior Model (Layer 2, top) and the Classifier Model (Layer 1, bottom). Both lanes share the same chunking model (Mixtral Nemo) and embedding model (Qwen 3 Embedding), and retrieve from a shared vector database. The Behavior Model additionally generates a reasoning trace through Qwen 3 30B before chunking.

Layer 1: Prompt Classifier. The first layer, the *prompt classifier*, operates on the user prompt before the agent begins to act on it. The prompt is first segmented by a lightweight chunking model (mistral-nemo). The chunker returns a small set of semantically coherent spans together with a tag describing what kind of content each span carries and a salience score that reflects how relevant the span is for an injection decision. Chunking is performed by a separate small model rather than inside the classifier itself, because

the goal at this stage is to isolate the parts of the prompt that actually carry instructional intent before any embedding cost is paid on filler text.

Each chunk is then embedded using a high-dimensional multilingual embedding model (qwen3-embedding-8b). The embedding of every chunk is matched against a vector database of previously labeled chunks using cosine similarity. The database is hosted in PostgreSQL with the pgvector extension, and retrieval is performed in two stages. The first stage uses a binary-quantized index to fetch a candidate set quickly. The second stage reranks the candidates using exact cosine distance. This design keeps retrieval latency bounded as the database grows, which matters because the database is expected to expand over time as new attack examples are added.

Once the k nearest neighbors for a chunk are retrieved, the chunk is labeled by majority among those neighbors. A chunk is treated as malicious when the fraction of its neighbors carrying the malicious label exceeds a configurable threshold. After every chunk in the prompt has been labeled, the prompt itself is labeled by a strict majority vote across the chunks. Both the neighborhood size k and the per-chunk threshold are tunable parameters of the system, and concrete values are deferred to the Evaluation section.

Because the detection knowledge lives in the database rather than in model weights, the detector adapts cheaply. If a new attack family appears, a small batch of labeled examples can be embedded and inserted, and the detector starts catching the new family from that point onward without retraining. The same holds for new target languages, because the embedding model is multilingual and aligns semantically similar text across languages into nearby regions of the embedding space.

Layer 2: Behavioral Classifier. The second layer, the *behavioral classifier*, operates on the model’s reasoning stream rather than on the prompt. When the agent begins to process a prompt, MnM collects the reasoning text emitted by the model (the reasoning field of the streaming response from a reasoning-capable LLM, qwen3-30b-a3b-thinking-2507 in our implementation) and feeds it into the same chunk-embed-retrieve pipeline used in Layer 1. The chunker is the same small model used for the prompt, and the embedding model is the same multilingual encoder. The vector database, however, is different. It stores fingerprints of reasoning patterns rather than of prompts, with each entry labeled as either injection or clean.

This second layer exists because input classification on its own is not sufficient. An injection that is phrased benignly at the prompt level can pass Layer 1 and only reveal itself in the way the model begins to reason about the task. By comparing the live reasoning against a database of known compromised and clean reasoning patterns, the behavioral classifier provides a second independent signal that captures this class of attacks. The classifier produces a per-chunk label and an overall verdict using the same kNN majority logic as Layer 1.

Because the classifier produces a signal per reasoning chunk, the architecture is compatible with streaming use, where the next per-chunk signal becomes available as soon as the next chunk arrives. This property is what makes an early-stop gate possible: a streaming gate that aggregates per-chunk signals as they are produced

can interrupt generation before the model emits a final response or a tool call. The current evaluation operates on full reasoning traces, so the details of the gate aggregation rule are not fixed here. The gate is treated as a downstream consumer of the per-chunk signal, and its operating point is left to future calibration work.

Vector Database and Labeling. Both layers share the same back-end infrastructure and the same embedding model, but their databases are populated from different sources. The Layer 1 database is built by chunking and embedding prompts from the Malicious Prompt Detection Dataset (MPDD), with each chunk inheriting the prompt-level label. The Layer 2 database is built by sampling prompts from MPDD, running them through the same reasoning model used at inference time, chunking the resulting traces, and embedding each chunk. Labels are inherited from the input prompt. A reasoning trace produced from a malicious prompt is added to the injection bank, and a trace produced from a benign prompt is added to the clean bank.

This input-based labeling is convenient because it does not require any per-trace annotation, but it is slightly noisy. A model that resists an injection still produces safe reasoning, and that reasoning is added to the injection bank if it originated from a malicious prompt. The Discussion section returns to this point and identifies outcome-based labeling, where a judge model verifies whether the injection actually succeeded, as the natural next step. The current databases are sufficient to support the evaluation reported in the rest of the paper, but the labeling strategy is a known source of measurement noise.

Hyperparameters and Fusion. MnM exposes two operating-point parameters that apply to both layers identically. The first is the neighborhood size k , which controls how many examples each chunk is compared against during retrieval. The second is the per-chunk threshold, which controls how much agreement among those neighbors is required before a chunk is flagged. Concrete values for both parameters are deferred to the Evaluation section, where they are swept across a held-out set. The per-chunk threshold in particular embeds a precision-recall tradeoff directly into the system. A high threshold produces few flagged chunks but very high precision per flag. A lower threshold catches more borderline cases at the cost of more false positives. Calibrating the threshold against the deployment setting is therefore part of the evaluation rather than a fixed design decision.

The two lane verdicts are fused with a logical OR. The combined detector raises an alarm if either the prompt classifier or the behavioral classifier reports an injection. This choice is conservative on purpose. The two layers detect different failure modes, and an attack that is caught by either layer is an attack that the system can block. In the evaluation harness used in this work, both lanes always run on every prompt, even when the first lane has already raised an alarm, because the harness needs to measure each signal independently before combining them. In a deployed setting it would be reasonable to short-circuit on a Layer 1 hit, since the agent has not begun to reason yet, but the harness reported here treats the two signals as independent measurements.

4 EVALUATION

Datasets. The vector databases that back both layers are built from the Malicious Prompt Detection Dataset (MPDD). The Layer 1 database contains 89 846 chunk embeddings derived from MPDD prompts. The Layer 2 database contains 28 060 chunk embeddings derived from reasoning traces that were generated by running MPDD prompts through the same reasoning model used at inference time (qwen3-30b-a3b-thinking).

Because both databases are built from MPDD, evaluating on MPDD would test on training data for the prompt lane. All evaluation is therefore conducted on held-out datasets that were never used to populate the databases.

English. We construct a balanced test set of 200 prompts: 100 injection prompts drawn from the Gandalf Ignore Instructions dataset (Lakera, MIT license), a collection of human-written instruction-override attacks, and 100 benign prompts drawn from the ShareGPT conversational corpus.

Hindi/Hinglish (zero-shot). We use 200 prompts (100 malicious, 100 benign) from the Hindi and Hinglish Prompts dataset, equally split between Hindi and Hinglish. This is a zero-shot cross-lingual evaluation: the vector databases contain only English data, and no Hindi or Hinglish examples are present at retrieval time.

Hindi/Hinglish (low-label seeding). To test whether the retrieval-based architecture can close the cross-lingual gap with minimal labeled data, we run a second Hindi evaluation after seeding the shared vector database tables with a small batch of Hindi reference embeddings. These reference embeddings correspond to 10% of the Hindi evaluation set (approximately 18 prompts), chunked and embedded through the same pipeline, then inserted into the existing English tables with a +10 000 000 primary-key offset to avoid collisions. This experiment uses 180 prompts (90 malicious, 90 benign).

Models. Three models form the inference pipeline. **mistral-nemo** segments prompts and reasoning traces into semantically coherent chunks. **qwen3-embedding-8b** embeds each chunk into a 4096-dimensional vector. **qwen3-30b-a3b-thinking** generates the reasoning trace that the behavioral layer classifies. All models are accessed through OpenRouter, and all calls use temperature 0.

Infrastructure. Embeddings are stored and retrieved from PostgreSQL 17 with pgvector 0.8.2. Retrieval uses a two-stage pipeline: binary-quantized Hamming HNSW to fetch 64 candidates, followed by exact cosine reranking to return the top- k .

Hyperparameter sweep. The system exposes two parameters: the neighborhood size k and the per-chunk threshold θ (the fraction of a chunk’s k neighbors that must be injection-labeled for the chunk to be flagged). We sweep $k \in \{1, \dots, 10\}$ and $\theta \in \{0.1, 0.2, \dots, 1.0\}$, yielding 100 operating points per evaluation setting. For k values up to the original retrieval depth, the sweep re-uses stored neighbor lists (offline, no database access). For larger k , the sweep re-queries pgvector with the stored embeddings.

Metrics. For each operating point we report accuracy, F1, precision, and recall for the prompt classifier (class), the behavioral classifier (behavior), and the combined OR-fused verdict. Confusion matrices are reported at the best operating point for each evaluation setting.

5 RESULTS

English ($n=200$). Table 1 reports the results at a moderate operating point ($k=3, \theta=0.5$). Both lanes achieve high recall at this setting: the behavior lane reaches 97.0% recall with 0.85 F1, and the prompt lane reaches 95.0% recall with 0.79 F1. OR-fusion pushes combined recall to 100%, meaning every injection in the test set is caught, though precision drops to 0.65 because both lanes contribute false positives at this threshold.

Table 1: English at $k=3, \theta=0.5$ ($n=200$).

Lane	Accuracy	F1	Precision	Recall
Prompt (class)	0.750	0.792	0.679	0.950
Behavior	0.825	0.847	0.752	0.970
Combined (OR)	0.730	0.787	0.649	1.000

The hyperparameter sweep identifies $k=4, \theta=0.7$ as the best operating point (Table 2). At this point the two lanes converge to similar F1 scores (0.82 for both), and the combined verdict reaches 0.86 F1 with 99% recall and 84.5% accuracy. The higher threshold reduces false positives compared to the $\theta=0.5$ setting, which explains the improvement in combined accuracy from 73.0% to 84.5%.

Table 2: English best operating point ($k=4, \theta=0.7, n=200$).

Lane	Accuracy	F1	Precision	Recall
Prompt (class)	0.820	0.820	0.820	0.820
Behavior	0.815	0.812	0.825	0.800
Combined (OR)	0.845	0.865	0.767	0.990

Figure 2 shows the F1 landscape across the sweep grid. The behavior lane is more robust to threshold variation than the prompt lane: its F1 remains above 0.75 across a wide band of k and θ values, while the prompt lane degrades more sharply at extreme thresholds.

Hindi/Hinglish zero-shot ($n=200$). Table 3 reports the zero-shot Hindi results at $k=3, \theta=0.5$. The vector databases contain only English data, so any detection ability in Hindi comes entirely from the multilingual embedding model’s cross-lingual alignment. Under these conditions, the behavioral layer achieves 76.0% accuracy and 0.73 F1, while the prompt layer falls to 61.5% accuracy and 0.51 F1. The gap between the two lanes is larger in Hindi than in English, which is consistent with the expectation that reasoning patterns transfer across languages more reliably than raw prompt text.

Table 3: Hindi/Hinglish zero-shot ($k=3, \theta=0.5, n=200$).

Lane	Accuracy	F1	Precision	Recall
Prompt (class)	0.615	0.510	0.702	0.400
Behavior	0.760	0.733	0.825	0.660
Combined (OR)	0.735	0.739	0.728	0.750

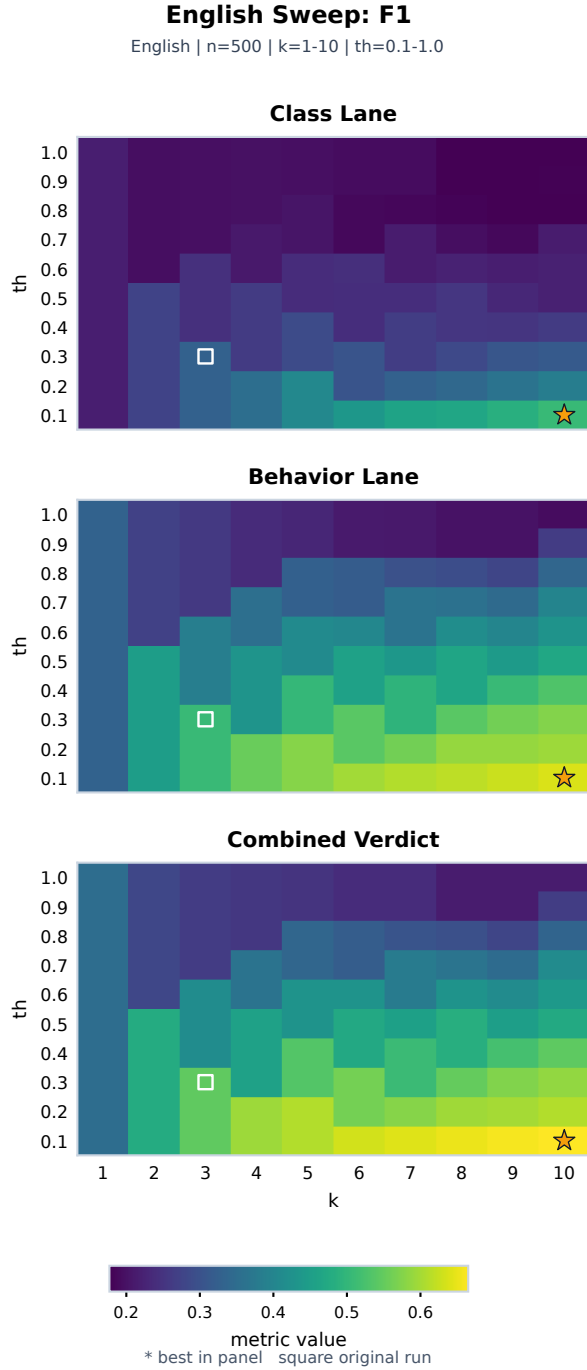


Figure 2: F1 heatmaps across k and θ for the English evaluation.

The sweep identifies $k=5$, $\theta=0.5$ as the best Hindi operating point (Table 4), reaching 77.5% combined accuracy and 0.78 F1. The behavioral layer reaches 79.0% accuracy and 0.76 F1 with 87.2% precision, which indicates that the multilingual embedding model aligns

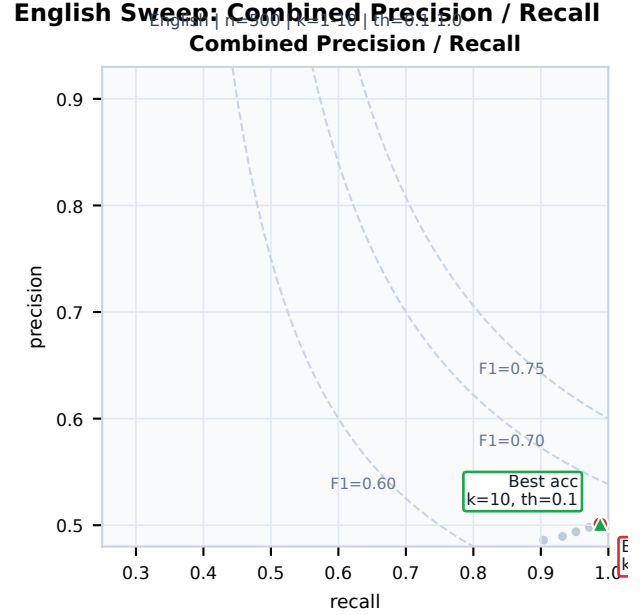


Figure 3: Precision-recall tradeoff for the English evaluation.

reasoning patterns well enough to support cross-lingual detection even without any target-language data. However, the prompt lane remains weak at 63.0% accuracy and 0.53 F1, because embedding-space proximity alone is not enough to reliably match Hindi prompt chunks against English reference chunks.

Table 4: Hindi/Hinglish zero-shot best ($k=5$, $\theta=0.5$, $n=200$).

Lane	Accuracy	F1	Precision	Recall
Prompt (class)	0.630	0.526	0.732	0.410
Behavior	0.790	0.764	0.872	0.680
Combined (OR)	0.775	0.776	0.772	0.780

Hindi/Hinglish with low-label seeding ($n=180$). Table 5 reports the best operating point after seeding the database with 10% Hindi reference embeddings. The prompt lane improves from 63.0% to 71.1% accuracy and from 0.53 to 0.70 F1, confirming that the retrieval-based architecture responds directly to the addition of target-language examples. The behavioral lane shows a smaller gain (from 79.0% to 80.0% accuracy), which is expected because reasoning traces are generated in English by the model regardless of the input language, so the behavior lane already had relevant neighbors before seeding.

The combined F1 moves from 0.78 (zero-shot) to 0.79 (seeded), a modest improvement. However, the more telling change is in the prompt lane, where the addition of approximately 18 Hindi reference prompts raises F1 by 0.18 points (from 0.53 to 0.70). This result demonstrates the core property of the retrieval-based design: new

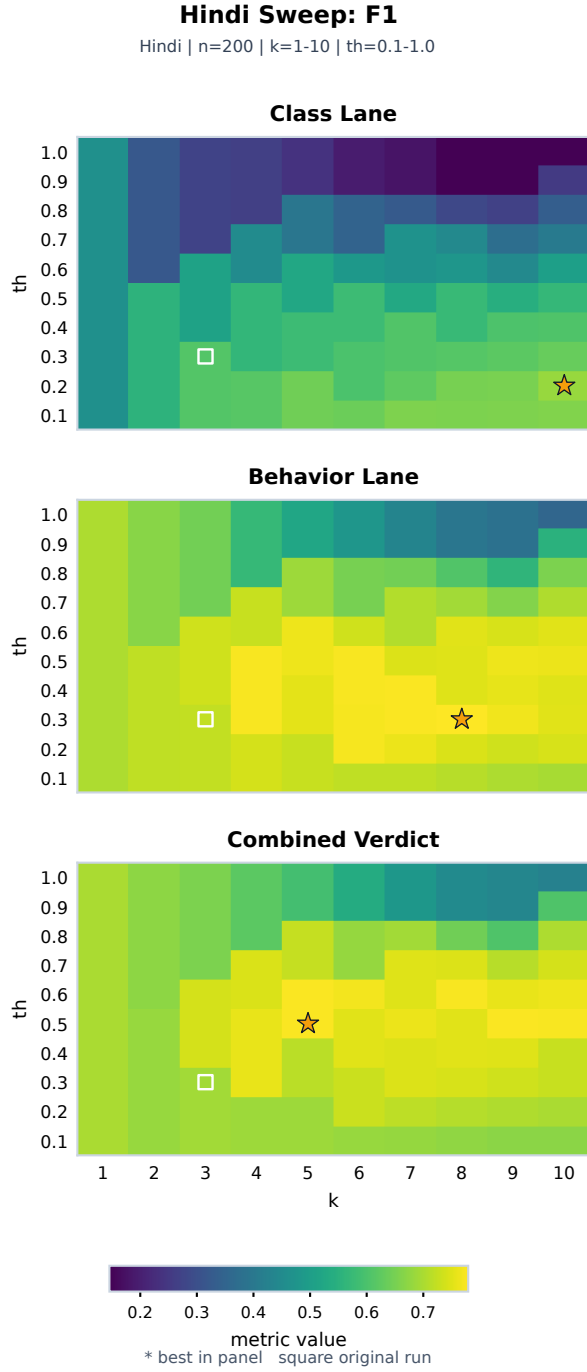


Figure 4: F1 heatmaps across k and θ for the Hindi/Hinglish zero-shot evaluation.

detection capability for an unseen language can be added by inserting a small batch of labeled examples, without touching the model weights or rerunning any training pipeline.

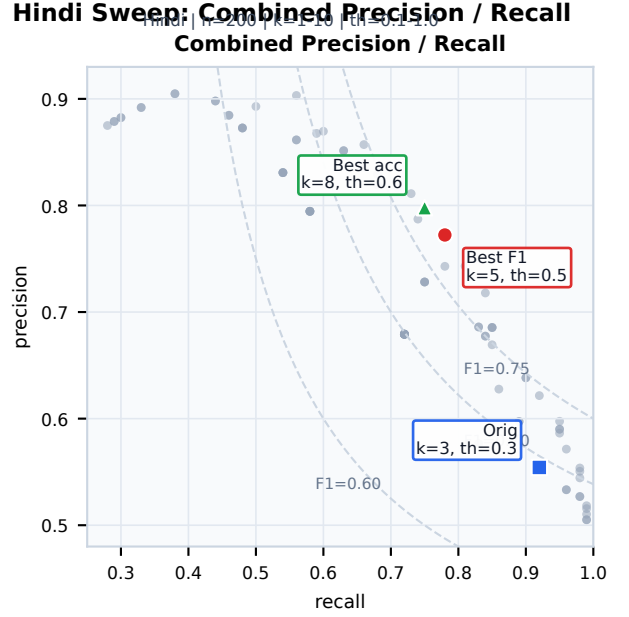


Figure 5: Precision-recall tradeoff for the Hindi/Hinglish zero-shot evaluation.

Table 5: Hindi/Hinglish with low-label seeding, best point ($k=4, \theta=0.3, n=180$).

Lane	Accuracy	F1	Precision	Recall
Prompt (class)	0.711	0.705	0.721	0.689
Behavior	0.800	0.793	0.821	0.767
Combined (OR)	0.756	0.786	0.698	0.900

Cross-language comparison. Table 6 summarizes the best operating points across all three evaluation settings.

Table 6: Best operating points across evaluation settings.

Setting	k	θ	Accuracy	F1	Recall
English ($n=200$)	4	0.7	0.845	0.865	0.990
Hindi zero-shot ($n=200$)	5	0.5	0.775	0.776	0.780
Hindi seeded ($n=180$)	4	0.3	0.756	0.786	0.900

Three patterns hold across the evaluation. First, the behavioral layer outperforms the prompt layer in every setting, which supports the claim that reasoning-level signals capture attacks that input classification misses. Second, OR-fusion boosts recall at the expense of precision, with the tradeoff governed by θ . Third, the performance gap between English and Hindi is moderate (approximately 9 points of combined F1 at the best operating points), suggesting that the multilingual embedding space transfers detection ability across languages even without any target-language data. The seeding experiment narrows part of this gap on the prompt

lane, though the behavioral lane, which was already strong in the zero-shot setting, benefits less.

6 DISCUSSION

The behavioral layer of MnM targets a failure mode that input classifiers cannot catch: an injection that looks benign at the prompt level but succeeds by gradually steering the model’s reasoning. By operating on the reasoning stream directly, MnM can detect this drift and issue an early-stop signal before the model completes its generation. This property is architecturally impossible for any post-hoc or input-only system.

The gate design embeds a precision-recall tradeoff as a first-class hyperparameter. A strict threshold catches high-confidence injections with few false positives, while a lenient threshold catches lower-confidence injections at the cost of higher false-positive rates. The evaluation results show that the operating point matters more than the underlying embedding quality: moving from $\theta=0.5$ to $\theta=0.7$ on English raises combined accuracy from 73.0% to 84.5% with the same embedding model and the same database. Calibrating this tradeoff on a held-out set is therefore a necessary step for any deployment.

The multilingual evaluation surfaces a distributional challenge that is specific to the behavioral approach: if the reasoning patterns in the vector database are drawn from English traces only, cross-lingual generalization depends entirely on the embedding model’s multilingual alignment. Our Hindi evaluation provides an initial signal on this, with the behavioral lane reaching 79.0% accuracy in the zero-shot setting. However, the prompt lane struggles at 63.0% accuracy under the same conditions. The low-label seeding experiment addresses this directly. By adding a small batch of Hindi reference embeddings (approximately 10% of the evaluation set) into the shared database, the prompt lane improves by over 8 percentage points in accuracy and its F1 rises from 0.53 to 0.70. This supports the practical claim of the retrieval-based architecture: new language support can be bootstrapped from a small set of labeled examples rather than requiring a full retraining cycle.

At the same time, the seeding experiment introduces a nuance to the “training-free” claim. While MnM does not use gradient-based training, it does require curated labeled data in the vector database. The distinction is practical rather than theoretical: adding 18 labeled Hindi prompts to the database is far cheaper than fine-tuning an embedding model or retraining a classifier, but it is not zero effort. For languages or attack families where even a small labeled dataset is unavailable, the system falls back to whatever cross-lingual transfer the embedding model provides on its own.

A separate limitation is the labeling strategy. Traces are labeled by input (malicious prompt $\Rightarrow$ injection bank, benign prompt $\Rightarrow$ clean bank). This is slightly noisy because a model that resists an injection produces safe reasoning despite originating from a malicious prompt. Outcome-based labels, using a judge to verify whether the injection actually succeeded, would produce a cleaner bank and are a natural next step.

Broader coverage remains necessary. The evaluation covers English and Hindi, but the multilingual attack surface documented by Burbano et al. [2] spans at least four language families. Extending the evaluation to the full set of languages identified in that

work, including Chinese and Spanish, would strengthen the generalization claims.

7 CONCLUSION

We presented MnM, a dual-layer system for real-time detection of prompt injection in LLM agents. The main contribution is a behavioral classifier that monitors the model’s reasoning stream as it is generated, compares chunks against a vector database of compromised and clean reasoning patterns, and can trigger an early-stop gate before a harmful response or tool call is emitted. The system requires no gradient-based training and has been evaluated on English and Hindi prompts, reaching 84.5% combined accuracy on English and 77.5% on Hindi in a zero-shot cross-lingual setting.

The low-label seeding experiment shows that adding a small batch of target-language reference embeddings to the database can close part of the cross-lingual performance gap, especially for the prompt lane, without any retraining. This supports the core design hypothesis: that encoding detection knowledge in a vector database rather than in model weights makes the system easier to extend to new languages and attack families.

MnM addresses the multilingual attack surface documented by Burbano et al. [2] from the detector side, and the instruction-hierarchy gap identified by Guo et al. [5] from the behavioral monitoring angle. Open problems include calibrating the streaming gate on held-out data, scaling the reasoning bank, improving the labeling strategy with outcome-based judges, and extending the evaluation to more languages.

REFERENCES

- [1] Aiman Al Masoud, Marco Arazzi, and Antonino Nocera. 2026. SD-RAG: A framework for secure selective disclosure in retrieval-augmented generation against single-turn prompt-leaking attacks. *Expert Systems with Applications* 331 (2026), 133154. <https://doi.org/10.1016/j.eswa.2026.133154>
- [2] Luis Burbano, Diego Ortiz, Qi Sun, Siwei Yang, Haoqin Tu, Cihang Xie, Yinzhi Cao, and Alvaro A Cardenas. 2026. CHAI: Command Hijacking against embodied AI. arXiv:2510.00181 [cs.CR] <https://arxiv.org/abs/2510.00181> To appear at IEEE SaTML 2026.
- [3] Edoardo DeBenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. 2024. AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. arXiv:2406.13352 [cs.CR] <https://arxiv.org/abs/2406.13352>
- [4] Mohamed Amine Ferrag, Norbert Tihanyi, Djallel Hamouda, Leandros Maglaras, Abderrahmane Lakas, and Merouane Debbah. 2026. From prompt injections to protocol exploits: Threats in LLM-powered AI agents workflows. *ICT Express* 12, 2 (2026), 353–383. <https://doi.org/10.1016/j.icte.2025.12.001>
- [5] Chuan Guo, Juan Felipe Ceron Uribe, Sicheng Zhu, Christopher A. Choquette-Choo, Steph Lin, Nikhil Kandpal, Milad Nasr, Rai, Sam Toyer, Miles Wang, Yaodong Yu, Alex Beutel, and Kai Xiao. 2026. IH-Challenge: A Training Dataset to Improve Instruction Hierarchy on Frontier LLMs. arXiv:2603.10521 [cs.AI] <https://arxiv.org/abs/2603.10521>
- [6] Go Sato, Shusaku Egami, Yasuyuki Tahara, Akihiko Ohsuga, and Yuichi Sei. 2026. Addressing Prompt Injection in Large Language Models via In-Context Learning. *Computers, Materials & Continua* 87, 2 (2026). <https://doi.org/10.32604/cmc.2026.078188>
- [7] Lijuan Shi, Yajing Kang, Jie Hu, Xinchu Li, and Mingchuan Yang. 2025. Meticulous Thought Defender: Fine-Grained Chain-of-Thought (CoT) for Detecting Prompt Injection Attacks of Large Language Models. *IEEE Access* 13 (2025), 113194–113207. <https://doi.org/10.1109/ACCESS.2025.3583759>
- [8] Jagadeesan Srinivasan, Silla Regi, Anbarasa Anbarasan, Suresh A, Vetriselvi T, and Sivakumar Venu. 2026. Detection and analysis of prompt injection in Indian multilingual large language models. *Scientific Reports* 16 (04 2026). <https://doi.org/10.1038/s41598-026-43883-0>
- [9] Jiahao Yu, Yangguang Shao, Hanwen Miao, and Junzheng Shi. 2025. PROMPT-FUZZ: Harnessing Fuzzing Techniques for Robust Testing of Prompt Injection in LLMs. arXiv:2409.14729 [cs.CR] <https://arxiv.org/abs/2409.14729>