

Design Notes: Seven Cross-Domain Pre-processing and Detection Techniques for Prompt-Injection Defense (prompt-shield v0.5.0)

Thamilvendhan Munirathinam

Independent — <https://github.com/mthamil107/prompt-shield>

2026-06-18

Abstract. This technical report documents the novel detection and pre-processing techniques introduced in version 0.5.0 of the prompt-shield library — an open-source prompt-injection firewall released under Apache 2.0. Seven techniques are described with sufficient algorithmic detail to be independently reimplemented: many-shot structural detection via coefficient-of-variation analysis of turn markers (d029); a fail-soft YAML-rules engine with highest-severity-wins precedence (d030); a two-stage language-enforcement detector combining Unicode script-ratio analysis with optional langdetect (d031); operator-defined denied-topic detection with multi-group keyword clusters and most-hits-wins resolution (d032); multi-turn topic-drift detection via Jaccard similarity between the current turn and a self-anchored fingerprint of the conversation's first turns (d033); a four-stage idempotent normalization pipeline that emits change-tracking metadata; and a fan-out multi-encoding preprocessor (base64, hex, URL, HTML entities, ROT13) that feeds decoded candidates back through the full detector stack. We also document the removal of the 512-token input-length cap on the DeBERTa-v3 semantic classifier via overlap-chunked max-pool aggregation. This is a companion to arXiv:2604.18248 (the main prompt-shield paper covering d027 / d028 / adversarial fatigue). It is published as a dated public disclosure under CC BY 4.0; the techniques described here are released as prior art and the author makes no claim to patent rights over them.

Keywords: prompt injection, LLM security, AI safety, prior art, many-shot jailbreak, topic drift, homoglyph normalization, multi-encoding decoder, language enforcement

1. Purpose and scope

This document is a technical design note for the novel detection and pre-processing techniques introduced in prompt-shield v0.5.0 that are not covered in the companion arXiv paper (arXiv:2604.18248). It is published as a dated public disclosure so that the techniques described here are unambiguously prior art as of the timestamp on this document and its DOI on Zenodo.

The companion paper covers d027 (stylometric discontinuity), d028 (Smith-Waterman sequence alignment with semantic substitution matrix), and the adversarial fatigue tracker. This document covers the seven additional cross-domain techniques shipped in v0.5.0, in enough algorithmic detail to be independently reimplemented from this text alone.

All techniques are released under Apache 2.0 (code) and CC BY 4.0 (this description). The intent of this disclosure is not to restrict use — it is to ensure that no future patent application can claim novelty over these techniques.

2. d029 — Many-shot Structural Detection

Threat model. Anthropic (2024) disclosed a class of attack where a prompt contains a large number of structurally identical fake conversation turns that condition the model to comply with the final turn. Verbatim regex on any single turn fails. Structural features detect it:

- Turn marker density: count canonical markers (Q:, A:, Human:, Assistant:, [INST], etc.) divided by total non-whitespace chars. Above ~1 marker per 80 chars in long prompts is anomalous.
- Inter-turn period regularity: measure character distance between adjacent markers and compute coefficient of variation (stdev/mean). Genuine conversations have $CV \geq 0.5$; mechanically generated many-shot attacks have $CV \leq 0.15$.

- Total turn count threshold: ≥ 30 turn markers in a single prompt is rare in legitimate use.

Decision rule. Detection fires when turn count ≥ 30 AND turn density above threshold AND $CV \leq 0.15$. Confidence scales with how far regularity exceeds threshold.

Novelty claim. Existing literature on many-shot jailbreaking focuses on attack construction. This technique reframes detection as periodicity analysis on structural markers, borrowing from time-series anomaly detection — short conversations have irregular intervals, mechanically padded ones do not.

Implementation reference. `src/prompt_shield/detectors/d029_many_shot_structural.py`.

3. d030 — Custom YAML Rules Engine

Threat model. Operators routinely need to block organization-specific terms (internal codenames, sensitive data class names) that no general detector can know. Hand-editing detector Python is high-friction; a declarative rule format lowers it.

Technique. A directory of YAML rule files is loaded at engine startup. Each rule has the schema:

```
rules:
- id: internal-codename
  pattern: '\binternal-codename-X\b'
  severity: high          # critical | high | medium | low
  action: block           # block | flag | log
  description: "..."/>

```

Loading is fail-soft per rule: malformed YAML, invalid regex, or entries missing id/pattern are logged and skipped without aborting the rest of the ruleset. On detection, when multiple rules match, highest-severity wins (critical > high > medium > low). The chosen rule's severity, action, and metadata propagate to the engine-level decision.

Novelty claim. The fail-soft rule loader + highest-severity-wins precedence + per-rule action verb is a distinctive combination — most regex-rule engines are either fail-hard or first-match-wins.

Implementation reference. `src/prompt_shield/detectors/d030_custom_rules.py`.

4. d031 — Two-Stage Language Enforcement

Threat model. Many deployments are explicitly English-only or restricted to a small allow-list. Multilingual jailbreaks (translations of common attack templates into less-trained languages) bypass English-only RLHF safety signal. Blocking non-allow-list languages is a cheap, high-precision input gate.

Technique. Detection is two-stage to minimize dependency surface.

Stage 1 — fast-path script analysis (no dependencies). Pre-compiled regexes for Unicode script blocks: Cyrillic, Greek, Arabic, Hebrew, Devanagari, Thai, and the CJK union (Han, Hiragana, Katakana, Hangul). For each script, compute the ratio of matching characters to total non-whitespace, non-digit characters. If any non-allowed script exceeds 15% of input characters, the inferred language is rejected immediately.

Stage 2 — langdetect fallback (optional dependency). If the input is Latin-script and the fast path doesn't trigger, optionally invoke langdetect with seed=0 for determinism. Accept the top guess only if probability ≥ 0.75 ; otherwise treat as allowed (high uncertainty defaults to permissive to avoid false positives on short or technical input).

Novelty claim. The fast-path script-ratio gate is well-known in isolation. The novelty is the two-stage gated combination: script-fraction analysis as a no-dependency fast path, with langdetect invoked only when the fast path is inconclusive AND optional dependencies are present. Makes the detector deployable in environments that cannot pull in langdetect while still benefiting from it when available.

Implementation reference. `src/prompt_shield/detectors/d031_language_enforcement.py`.

5. d032 — Operator-Defined Denied Topics

Threat model. Deployments often need to block off-topic requests entirely (medical, legal, political opinions in a code-assistant). These are not 'attacks' in the prompt-injection sense — they are policy violations. The same engine should evaluate them with a single declarative interface.

Technique. Operator-defined topic groups, each a named keyword cluster. A detection fires when the number of keyword hits from any single group meets a configurable minimum threshold (default 2).

```
d032_topic_enforcement:
  denied_topics:
    - name: medical_advice
      keywords: ["diagnose", "prescription", "dosage", "symptoms"]
      severity: high
    - name: legal_advice
      keywords: ["lawsuit", "attorney", "court", "litigation"]
      severity: medium
  min_keyword_hits: 2
```

Keywords are matched as word-bounded literal phrases. Among multiple matching topics, the topic with the most hits wins; ties resolved by configuration order.

Novelty claim. Combination of (a) multiple competing keyword groups with per-group severity, (b) min-hits-per-group threshold (so isolated word usage doesn't fire — "I'm not asking for medical advice" doesn't trigger on "advice"), and (c) most-hits-wins precedence. This sits between naive single-keyword block-lists and full ML topic classifiers.

Implementation reference. `src/prompt_shield/detectors/d032_topic_enforcement.py`.

6. d033 — Multi-Turn Topic Drift via Jaccard Anchor Similarity

Threat model. A class of slow jailbreaks operates across many turns. Each individual turn is benign — a coding question, then a casual aside, then a hypothetical, then an escalated request. No single turn is detected; the cumulative drift from the conversation's established topic is what makes the final harmful request succeed.

Technique. Anchored n-gram drift.

- Build the anchor: concatenate the first N turns of the conversation (default N=2). This is the 'what the conversation is about' reference. Computed once per session.
- Tokenize and stopword-filter both the anchor and the current turn. Use a small English stoplist (~70 common words) to remove function words that would inflate similarity.
- N-gram fingerprint: build the set of token n-grams (default bigrams, configurable). Each fingerprint is a set of tuples.
- Jaccard similarity between the anchor fingerprint A and current-turn fingerprint C: $|A \cap C| / |A \cup C|$.
- Decision: if similarity is below `min_anchor_similarity` (default 0.05) AND the conversation has at least `min_turns` (default 4), fire detection. The minimum-turns gate prevents firing on short interactions that haven't established a topic.

Confidence scaling. $\min(0.95, 0.4 + (\text{threshold} - \text{similarity}) \times 4)$. A complete topic flip (similarity = 0) yields high confidence; a borderline drift yields lower confidence.

Novelty claim. This is the most distinctive technique in v0.5.0. Existing multi-turn jailbreak detection focuses on either (a) semantic-embedding distance from a seed prompt (heavy ML dependency) or (b) detecting specific escalation patterns. The Jaccard-against-anchor approach is novel in three ways: (1) no model dependency — pure n-gram set arithmetic, sub-millisecond evaluation; (2) the anchor is the conversation's own first turns, not an externally supplied 'safe' prompt — the detector adapts to each conversation's established topic without operator configuration of what the topic is; (3) two-gate decision ($\text{similarity} < T$ AND $\text{turn_count} \geq M$) prevents false-positives on legitimate topic switches in short conversations.

Implementation reference. `src/prompt_shield/detectors/d033_topic_drift.py`. Test coverage: `tests/detectors/test_d033_topic_drift.py` (10 tests).

7. Normalization Pipeline (pre-detector)

Threat model. Many obfuscation attacks rely on the input visually rendering identically to a known attack while being lexically different — Cyrillic homoglyphs, zero-width characters splitting trigger words, full-width Unicode reading identically to ASCII. Detectors that match against the raw byte stream miss these.

Pipeline structure. Four idempotent stages, each independently toggleable: (1) NFKC normalization composes combining marks and maps compatibility characters; (2) zero-width stripping removes U+200B, U+200C, U+200D, U+FEFF, U+2060; (3) Cyrillic-to-Latin homoglyph mapping uses a curated table of 30+ visually-identical Cyrillic→Latin mappings (one-way only — never reverse, to avoid mangling legitimate Cyrillic text); (4) whitespace collapse trims multi-space, tabs, newlines to single spaces.

Each stage is idempotent: running it twice produces the same output as running it once. This guarantees consistent input across normalization-aware and normalization-naïve detectors composed in the same engine.

Result object. The pipeline returns `NormalizationResult(text, original, changes: list[str])` so downstream detectors can see which normalizations fired — useful both for explanation and as a detection signal in itself (an input that was zero-width-stripped is itself suspect).

Novelty claim. The idempotent-stage pipeline with change-tracking output. Existing libraries typically apply normalizations destructively without surfacing what changed. The change-tracking output enables a meta-detector ('input contained zero-width characters') on top of the standard detectors.

Implementation reference. `src/prompt_shield/normalization/pipeline.py`.

8. Multi-Encoding Preprocessor (pre-detector)

Threat model. Encoded payloads are the canonical way to smuggle attack content past detectors: base64, hex, URL-encoded, HTML-entity-encoded, ROT13. Single-encoding detectors miss combined attacks (`base64(rot13('ignore...'))`).

Technique. Fan-out candidate set. The preprocessor returns a `DecodedSet` — a list of `DecodedCandidate(text, encoding, source_span)` for each successfully decoded substring. Detectors then run on each candidate independently.

- Base64: regex `(?<![A-Za-z0-9+/=])([A-Za-z0-9+/]{16,}={0,2})(?![A-Za-z0-9+/])` — captures contiguous base64-charset runs of ≥ 16 chars with lookahead to preserve padding = characters.
- Hex: `(?<![0-9a-fA-F])([0-9a-fA-F]{2}){6,}(?![0-9a-fA-F])` — even-length hex runs of ≥ 12 characters.
- URL: `urlib.parse.unquote` applied to `%XX`-containing substrings.
- HTML entities: `html.unescape` applied to `&XXX;` / `&#NNN;` patterns.
- ROT13: heuristic — any contiguous letters-only word ≥ 7 chars is a candidate; decoded variant is added.

Decoded candidates feed back through the full detector pipeline. A detection on a decoded candidate is reported with the original (encoded) span as the match position.

Novelty claim. The fan-out candidate-set model. Most detectors are written as `decode_if_possible(text) → detect(text)`, handling one encoding per substring. The fan-out model treats decoded variants as additional inputs to the full detector stack, so a base64-encoded ROT13-encoded attack is caught when (a) base64 decode produces ROT13 text, (b) ROT13 decode produces the original attack, (c) any detector matches the final decoded form.

Implementation reference. `src/prompt_shield/decoders/preprocessor.py`.

9. Removal of d022 Input-Length Cap (semantic ML classifier)

Prior limitation. The DeBERTa-v3 semantic classifier has a 512-token model context. Inputs longer than ~6000 characters were previously truncated to the first 512 tokens, leaving the tail of long inputs uncovered.

Technique. Sliding chunked max-pool. Long inputs are split into overlapping chunks (chunk_size=512 tokens, chunk_stride=384 tokens, 128-token overlap), capped at max_chunks=8. Each chunk is independently scored by DeBERTa. The final confidence is the max over all chunk confidences (max-pool aggregation), not the mean.

Why max-pool. Mean-pool dilutes the signal — a 5000-token prompt with one malicious 100-token segment averages out to 'looks fine'. Max-pool preserves the signal: any segment scoring high triggers the engine. For a defensive detector, max-pool is the correct aggregation.

Novelty claim. Chunking itself is standard. The combination of (a) overlap to avoid splitting tokens across chunks, (b) max-pool aggregation, and (c) a hard chunk cap (max_chunks=8) to bound worst-case compute is the contribution. The cap prevents adversarial input-padding attacks that would force the detector to run for unbounded time.

Implementation reference. src/prompt_shield/detectors/d022_semantic_classifier.py.

10. Summary of novelty claims

Technique	Novel?	Primary novelty claim
d029 many-shot structural	Yes	Periodicity / CV analysis on turn markers
d030 custom YAML rules	Partial	Fail-soft loader + highest-severity-wins
d031 language enforcement	Yes	Two-stage script-ratio fast path + optional langdetect
d032 topic enforcement	Yes	Multi-group keyword clusters + min-hits + most-hits-wins
d033 topic drift	Yes (strongest)	Jaccard-against-self-anchor + two-gate decision
Normalization pipeline	Partial	Idempotent stages with change-tracking output
Multi-encoding preprocessor	Yes	Fan-out candidate set fed back through full detector stack
d022 chunking	Partial	Overlap + max-pool + hard chunk cap

11. Disclosure intent

This document, the companion paper arXiv:2604.18248, the public git history at <https://github.com/mthamil107/prompt-shield>, and the released artifact prompt-shield-ai==0.5.0 on PyPI constitute public disclosure of the techniques described above. They are released under permissive licenses (CC BY 4.0 for the text, Apache 2.0 for the code, with the explicit patent grant therein) so that anyone may freely use, modify, extend, or commercially deploy them.

The author makes no claim of patent rights on these techniques and disclaims any intent to seek such rights. This disclosure is intended to establish prior art so that no third party may seek patent rights over them either.

References

- Munirathinam, T. (2026). Beyond Pattern Matching: Seven Cross-Domain Techniques for Prompt Injection Detection. arXiv:2604.18248. <https://arxiv.org/abs/2604.18248>
- Anthropic (2024). Many-shot jailbreaking. <https://www.anthropic.com/research/many-shot-jailbreaking>
- prompt-shield v0.5.0 source code. <https://github.com/mthamil107/prompt-shield/tree/v0.5.0> (Apache 2.0)
- CHANGELOG.md, prompt-shield v0.5.0 entry. <https://github.com/mthamil107/prompt-shield/blob/v0.5.0/CHANGELOG.md>