

ZERO HARM AI DETECTORS - PUBLIC API GUIDE

Version scope: current codebase in this repository.

This guide documents the customer-facing Python APIs exported by 'zero_harm_ai_detectors'.

1) Install

```
# Heuristic mode only (fast path)
pip install zero_harm_ai_detectors

# AI mode support (transformers/torch/spacy)
pip install 'zero_harm_ai_detectors[ai]'
```

2) Main Entry Point

```
from zero_harm_ai_detectors import detect

result = detect(
    text,
    targets=DetectTarget.ALL,
    redaction_strategy="token",
    ai_config=None, # pass AIConfig(...) to enable AI mode
)
```

Parameters

- 'text: str' - Input text to scan.
- 'targets: DetectTarget | int' - Bitmask of targets ('PII', 'SECRET', 'HARMFUL').
- 'redaction_strategy: str | RedactionStrategy' - 'token', 'mask_all', 'mask_last4', 'hash'.
- 'ai_config: AIConfig | None' - If provided, AI mode is used automatically.

Return Type

- 'DetectionResult'
- 'original_text: str'
- 'redacted_text: str'
- 'detections: list[Detection]'
- 'mode: str' -> "heuristic" or "ai"
- 'harmful: bool'
- 'harmful_scores: dict[str, float]'
- 'severity: str' -> "none" | "low" | "medium" | "high"

3) Enums and Core Types

‘DetectTarget’ (bitmask)

- ‘DetectTarget.PII’
- ‘DetectTarget.SECRET’
- ‘DetectTarget.HARMFUL’
- ‘DetectTarget.ALL’

Example:

```
targets = DetectTarget.PII | DetectTarget.SECRET
result = detect(text, targets=targets)
```

‘RedactionStrategy’

- ‘RedactionStrategy.TOKEN’ (“token”)
- ‘RedactionStrategy.MASK_ALL’ (“mask_all”)
- ‘RedactionStrategy.MASK_LAST4’ (“mask_last4”)
- ‘RedactionStrategy.HASH’ (“hash”)

‘Detection’

- ‘type: str’
- ‘text: str’
- ‘start: int’
- ‘end: int’
- ‘confidence: float’
- ‘metadata: dict’

‘HarmfulResult’

- ‘harmful: bool’
- ‘severity: str’
- ‘scores: dict[str, float]’

4) Heuristic APIs (always available)

Aggregate

- ‘detect_all_heuristic(text, detect_pii=True, detect_secrets=True, detect_harmful=True, redaction_strategy="token") -> DetectionResult’

Individual Detectors

- ‘detect_emails(text) -> list[Detection]’
- ‘detect_phones(text) -> list[Detection]’

- 'detect_ssns(text) -> list[Detection]'
- 'detect_credit_cards(text) -> list[Detection]'
- 'detect_bank_accounts(text) -> list[Detection]'
- 'detect_dob(text) -> list[Detection]'
- 'detect_drivers_licenses(text) -> list[Detection]'
- 'detect_mrn(text) -> list[Detection]'
- 'detect_addresses(text) -> list[Detection]'
- 'detect_person_names_heuristic(text) -> list[Detection]'
- 'detect_secrets_heuristic(text) -> list[Detection]'
- 'detect_harmful_heuristic(text) -> HarmfulResult'

5) AI APIs (optional dependencies)

Availability

- 'AI_AVAILABLE: bool'
- 'check_ai_available() -> bool'

Configuration

'AIConfig' fields:

- 'backend: str = "transformers"' ("transformers" or "spacy")
- 'ner_model: str = "dslim/bert-base-NER"'
- 'spacy_model: str = "en_core_web_sm"'
- 'harmful_model: str = "unitary/multilingual-toxic-xlm-roberta"'
- 'ner_threshold: float = 0.70'
- 'harmful_threshold: float = 0.5'
- 'device: str = "cpu"' ("cpu" or "cuda")
- 'max_length: int = 512'
- 'error_handling: str = "raise"' ("raise", "open", "closed")

Pipeline Access

- 'get_pipeline(ai_config: AIConfig | None = None) -> AIPipeline'
- 'detect_all_ai(...) -> DetectionResult'
- 'AIPipeline', 'NERDetector', 'HarmfulContentDetector'

AI Fallback Behavior

- If AI NER fails:
 - transformers backend attempts spaCy fallback if available,
 - then falls back to heuristic person detection.
- Harmful detection in AI path uses transformers when available.
 - If unavailable/fails, falls back to heuristic harmful detection.

6) Validation and Utilities

- 'validate_input(text, config=...)'
- 'validate_input_soft(text, config=...)'
- 'InputConfig', 'InputValidationError', 'InputTooLongError'
- 'apply_redaction(text, detection_type, strategy)'
- 'redact_spans(text, detections, strategy="token")'
- 'find_secrets(text)'
- 'luhn_check(card_number)'
- 'shannon_entropy(text)'

7) Typical Usage

Heuristic mode (default)

```
from zero_harm_ai_detectors import detect, DetectTarget

result = detect(
    "Contact me at alice@example.com and 503-555-0100",
    targets=DetectTarget.PII,
)
print(result.redacted_text)
```

AI mode (auto-enabled by 'ai_config')

```
from zero_harm_ai_detectors import detect, AIConfig

result = detect(
    "Dr. Alice Brown works at OpenAI in San Francisco.",
    ai_config=AIConfig(backend="transformers"),
)
print(result.mode) # "ai"
```

8) Stability Notes for Customers

- Prefer 'detect(...)' as the primary integration point.
- Use enum bitmasks ('DetectTarget') instead of hard-coded integers.
- Treat detector internals ('_detect_*') as private and non-contractual.
- Use 'DetectionResult.to_dict()' for API serialization boundaries.