

ko-pii

한국어 PII 검출·가명화 라이브러리 — 구현 & 사용법

규칙 + 사전 + 체크섬 코어(ML 없이 · 외부 의존성 0) · 33종 PII
+ 옵션 NER 하이브리드 — 외부 검증 F1 0.97 (v1.14)

```
pip install ko-pii
```

빠른 시작

```
# ① 검출 — DetectionResult 리스트 반환
from ko_pii.detect import detect_all

for d in detect_all("홍길동 사무관 010-1234-5678, 900101-1234567"):
    print(d.label, d.text, d.start, d.end, d.confidence)
# PERSON 홍길동 0 3 1.0
# PHONE 010-1234-5678 ...
# RRN 900101-1234567 ...
```

```
# ② 가명화 (가역)
from ko_pii import Anonymizer, ProcessingMode

r = Anonymizer(mode=ProcessingMode.STRICT, strategy="tokenize").process(text)
r.text # "<PERSON_1> 사무관 <PHONE_1>, <RRN_1>"
r.vault.reveal("<PERSON_1>") # "홍길동" ← vault 가진 쪽만 복원
```

검출 — 3가지 방식 (ML 없이)

방식	구현	대상
체크섬 <code>checksum/</code>	주민·사업자·법인 = mod-11 가중합 / 카드 = Luhn	결정적 PII → ≈100%
사전 <code>dictionaries/</code>	성씨·직책·기관·대학·학과·행정구역·법정동	인명·주소·기관
컨텍스트 점수 <code>context/</code>	가산 점수기 (아래)	이름 등 비결정적

→ `detect_all()` = 28개 검출기 순차 실행 → 33종 PII (이름 점수는 다음 장 ↓)

이름 검출 — 한국어 문맥 점수 ★

주민번호처럼 결정적이지 않은 이름은 → 주변 한국어 신호를 점수로 합산, 합 ≥ 0.50 이면 PERSON

신호	무엇인가 (한국어 단서)	점수
필드라벨	공문서·양식의 이름칸 라벨 — 성명: · 신청인 바로 뒤	+0.50
직책	직함 — 공무원(사무관 · 과장) / 민간(대표 · 부장) 인접	+0.35
조사	한국어 조사 이/가/은/는/을/를 가 붙음 (장혁이)	+0.35
결정적 PII	전화·주민번호·이메일이 25자 내 인접	+0.40
성씨	한국 성씨로 시작 (187개)	+0.15
감점	일반어 -0.40 · 1글자 -0.30 · 영문/숫자 -0.20	

있을 때 vs 없을 때 (실측): 성명: 홍길동 ✓ / 홍길동 ✗ · 장혁이 떠났다 ✓ / 장혁 떠났다 ✗

→ 조사·직함·양식 라벨 = 한국어 특화 신호 — 정규식·해외 도구엔 없는 차별점

아키텍처 — 파이프라인

입력 텍스트

- ① 정규화 `core/unicode_norm` (NFKC·제로폭 제거) + `io_/text_normalizer` (공백·줄바꿈)
 ↳ `offset` 역매핑 → 마스킹 위치 보존
- ② 검출 `detect_all` → 28개 검출기 (`patterns/`)
- ③ 충돌해소 정렬(시작·위험도·길이·확신도) 후 `sweep` → 겹침 제거

▼
`DetectionResult[]` (`label·start·end·text·confidence·risk·legal_basis`)

- ④ 정책판정 모드별 BLOCK / REVIEW / PASS
- ▼
- ⑤ 가명화 전략 적용 → 결과 텍스트 + Vault(복원용)

검출과 가명화가 분리 → `DetectionResult` 만 받아 다른 파이프라인에 끼우기 쉬움

사용법 — 옵션 & CLI

```
# 검출 필터
detect_all(text, include=["RRN", "PHONE"])      # 특정 라벨만
detect_all(text, exclude=["PERSON"])

# 모드(차단강도) × 전략(치환방식)
Anonymizer(mode=ProcessingMode.PARANOID,        # 5종: PARANOID/STRICT/BALANCED/PERMISSIVE/AUDIT
            strategy="fpe")                    # 6종: tokenize/redact/asterisk/hashed/partial/fpe

# 문서 파일 입력 (파서 자동)
from ko_pii.io_ import read_text
text = read_text("민원.hwp")                  # HWP·PDF·DOCX·XLSX
```

```
# CLI
ko-pii doc.pdf -m STRICT -s tokenize -o masked.txt
ko-pii ./docs --batch --recursive --workers 4    # 디렉토리 일괄
ko-pii --labels                                  # 33종 라벨 키 목록 (NEW)
```

33종 카테고리 — include/exclude 라벨 키

그룹	라벨 키
결정적 검증 (8)	RRN 주민 · FRN 외국인 · BUSINESS_REG 사업자 · CORP_REG 법인 · DRIVER_LICENSE 면허 · PASSPORT 여권 · CARD 카드 · PNU 토지
키워드 anchor (8)	MEDICAL_INSURANCE 건강보험 · PRESCRIPTION_ID 처방 · EDI_DRUG 약품코드 · FAX · ACCOUNT 계좌 · EMPLOYEE_ID 사번 · PETITION_ID 민원 · COURT_CASE 사건
형식 검증 (7)	PHONE · EMAIL · IP · URL · POSTAL_CODE 우편 · VEHICLE 차량 · DOC_ID 문서번호
사전·컨텍스트 (6)	PERSON 성명 · ADDRESS 주소 · NATIONALITY 국적 · EDUCATION 학력 · MAJOR 전공 · POSITION 직책
인적 속성 (4)	DT_BIRTH 생년월일 · AGE 나이 · HEIGHT 신장 · WEIGHT 체중

```
from ko_pii.labels import ALL_LABELS, LABEL_INFO      # 코드에서 정보 조회
detect_all(text, include=["PERSON"])                  # 이름만 탐지
```

→ CLI `ko-pii --labels` 와 코드 레지스트리가 단일 정보 (redact 한글명과 테스트로 동기화)

구현 구조 (모듈 맵)

모듈	역할
patterns/ (26)	카테고리별 검출기 — regex + 체크섬/사전/컨텍스트 조합
checksum/	주민·카드(Luhn)·사업자·법인 검산
context/	이름 가산 점수기 + 17개 거부률
dictionaries/	성씨·직책·기관·대학·학과·행정구역·법정동(1만)
modes/	6가지 가명화 전략
vault/	가역 토큰 매핑 + AES-GCM 암호화 + 감사로그
io_/	HWP·PDF·DOCX·XLSX·CSV 파서 + dispatcher
legal/ · analytics/	법적근거 매핑 · 결합위험도·k-익명성

확장 (전부 선택 설치, 코어 불변)

```
# 가역 가명화 + 영속 Vault + 감사로그
r.vault.save("vault.json"); ReversibleVault.load("vault.json")

# 토큰 NER 하이브리드 (v1.14) - 룰=결정적 ID, ML=퍼지 *교체* (role_split)
from ko_pii.integrations.hf_token_ner import HFTokenNERAdapter
Anonymizer(secondary_detector=HFTokenNERAdapter("out/ner/final"),
            merge_mode="role_split") # 외부 검증 F1 0.97 구성 그대로

# 문서 분류기 하이브리드 (pip install ko-pii[classifier]) - 검토 트리거용
from ko_pii.classifier import HybridAnonymizer, HybridMode
HybridAnonymizer(rule, clf, mode=HybridMode.REVIEW_FLAG, classifier_threshold=0.5)
```

- 우회 차단: 전각(0 1 0 → 010)·제로폭 문자 정규화 후 검출 (offset 복원)
- 통합: Presidio recognizer · MCP 서버 ([presidio] / [mcp])

성능 ① 검출 정확도 (F1)

평가셋	구분	ko-pii	openai/PF	Presidio
생성 평가셋 540 (검증 gold)	자체	0.790	0.451	0.483
확장셋 1,938 (견고성 대조)	자체	0.825	0.538	0.478
KDPPII (일상 대화)	외부	0.660	0.264	0.273
KLUE (신문 인명)	외부	0.419	0.155	0.000

- 결정적 PII(주민·카드·여권·계좌)는 체크섬 검증 → **F1 ≈ 1.000**
- 견고성: 3.6배 큰 확장셋(1,938)에서도 ko-pii 우위 유지·확대 (LLM은 자기 gold 순환이라 제외)
- 참고선 (LLM 상한·크기효과) — **Gemma-4-31B KDPPII 0.796** (ko-pii 룰만으로 그 83%). 단 최소 **Gemma-4-E4B** 는 **0.613** — 대화체에선 ko-pii(0.660) 미만, 큰 모델만 룰을 능가
자체 = LLM 생성 + 검증(540은 2단계 감사) / 외부 = 인간 라벨 공개 데이터 · 동일 매처(match_forms_overlap)

성능 ② 속도 / 처리량

	처리 시간	처리량 (1코어)
ko-pii (규칙)	0.56 ms / 문서	~1,800 문서/초
신경망 모델 (RoBERTa)	42.6 ms / 문서	~23 문서/초

- 모델 로딩·GPU 불필요 → 메모리 가볍고 콜드스타트 없음
- 대량 처리: `ko-pii ./docs --batch --workers N` (병렬)
 - ※ 같은 CPU. 신경망 쪽은 예/아니오 분류 기준이라 정확히 같은 작업은 아님

NEW ① 퍼지의 약점은 ML 로 — 어떤 모델? (백본 실측)

룰의 약점 = 이름·주소 같은 퍼지 카테고리 → 토큰분류 NER 로 보완. 후보 전부 동일 레시피로 학습:

베이스	크기	KDPII	판정
klue/roberta-large (한국어 인코더)	336M	0.950	✅ 에폭 1에 0.92, 3ep 포화
openai/privacy-filter (PII 전용 MoE)	1.4B	0.633	lr 5배로도 klue 미달
gemma-4-E2B (신형 디코더 LLM, 직접 튜닝)	4.6B	0.353	두 lr 모두 ~0.33 정체

- "더 새롭고 큰 모델"로 뒤집히는 문제가 아님 — 한국어 양방향 인코더가 정답, 병목은 데이터
- 학습 15~30분/GPU 1장 · 전 과정 아카이브 공개([experiments/ner/](#) — 학습 9런·평가 17런·레지스트리)

NEW ② 평가를 의심하기 — gold 가 가짜 번호였다

인분포 벤치마크에선 ML 단독(0.949)이 최강으로 보였다. 그런데:

- 평가셋 gold ID 가 대부분 "형식만 맞는 가짜" — KDPII 카드 gold 의 9%만 Luhn 유효, 생성셋도 주민 15%·사업자 13%·법인 19%만 체크섬 유효
- 룰은 실존 불가 번호를 거부(올바른 설계)했는데 벤치마크가 FN 으로 처벌 — 사업자 recall 0.129 = gold 유효율 13% 와 정확히 일치 (스모킹 건)

→ 외부 검증셋 v2 를 새로 구축 (미사용 도메인 · ID 100% 체크섬 유효 · 신규 주입 패턴):

외부 검증 (400문서)	F1	ID recall
하이브리드 (룰=ID, ML=퍼지)	0.968	1.00 × 4종
ML 단독	0.929	0.85~0.94
룰 단독	0.892 ← 0.659 에서 복권	1.00 × 4종

아티팩트를 제거하니 순위가 원복 — 역할 분담 하이브리드가 진짜 챔피언

NEW ③ v1.14.0 — 하이브리드를 라이브러리 기능으로

```
ml = HFTokenNERAdapter("out/ner_fuzzy/final") # 직접 학습한 NER (레시피 공개)  
anon = Anonymizer(secondary_detector=ml, merge_mode="role_split")
```

- **role_split 병합 모드**: 합산(union)이 아닌 **교체** — 약한 쪽 FP 가 강한 쪽을 오염 못 함
(union 은 항상 role_split 이하 — 실측)
- 인간 라벨 외부 데이터(KLUE-NER 5,000문장)에서도 퍼지는 ML 압승: **0.84 vs 룰 0.38**
- 함께 배포: RRN 공백 표기 검출(880101 - 1234568), README 전 예제 실행 검증,
실험 스크립트 이식성(레포만으로 재현)

NEW ④ 남은 작업 — 가중치 공개 게이트

항목	상태
라이선스 검토	✅ 완료 — KDPII CC-BY-4.0(원출처 확인) · 생성셋 Apache 2.0 · 베이스 KLUE CC-BY-SA → 공개 시 가중치 CC-BY-SA-4.0
모델카드 초안	✅ 작성 완료 (한계 정직 고지 포함)
공공문서 인간 라벨 검증	❑ 마지막 게이트 — 키트·웹 UI 준비 완료, 라벨링 반나절 → 점수로 공개 판정

- 공개 전에도 사용자 동선은 닫혀 있음: 레시피 + 레포 데이터로 직접 학습 → **role_split** 두 줄
- 그 외: 도메인 꼬리 케이스(별명·은어) 보강, cross-encoder reranker (후순위)

정리

- 33종 한국 PII 검출 + 가명화 — 코어는 룰+사전+체크섬, ML 없이·외부 의존성 0
- 정확도: 결정적 PII $\approx 100\%$ · 외부 벤치 해외 도구 우위 · NER 하이브리드 외부 검증 0.97
- 속도: 0.56 ms/문서 (~1,800/초), GPU 불필요 (하이브리드는 opt-in)
- 품질: 792 테스트 · CI(3.10~3.13) · PyPI v1.14.0 · 웹 데모 · 실험 전 과정 아카이브

```
pip install ko-pii          # 코어 (외부 의존성 0)
pip install ko-pii[file]    # + HWP/PDF 파서
```

github.com/Marker-Inc-Korea/ko-pii

부록 — 이름 점수기 전체 (Q&A)

본문 슬라이드는 대표 신호만 단순화한 것. 실제 점수기는 **13개 신호 + 다단계 패스**:

분류	신호 (가중치)
문맥	필드라벨 +0.50 · 직책 +0.35 · 조사 +0.35 · 결정적PII +0.40/+0.20 · 성씨 +0.15 · 기관 +0.10
이름 자체	토큰내직책 +0.35 · 이름끝음절 +0.10 · 음절통계(가변) · 나이/성별 +0.30
감점	일반어 -0.40 · 1글자 -0.30 · 영문/숫자 -0.20
누적	같은 문서 재등장 이름 사전 부스트 (가변)

파이프라인

- **Pass 0** — 매크로 기관+이름+직책 정규식 → 0.95 즉시 확정
- **Pass A** — 후보 점수화 + 문장 내 공출현 부스트 +0.15 → 합 ≥ 임계값이면 검출
- **Pass B** — 누적 사전으로 약한 후보 재평가
- **동적 임계값** — 기본 0.50, 2자 + 이름끝 약하면 0.60 (엄격)

→ 각 검출은 evidence 필드에 **발동한 신호 목록**을 남겨 추적·감사 가능