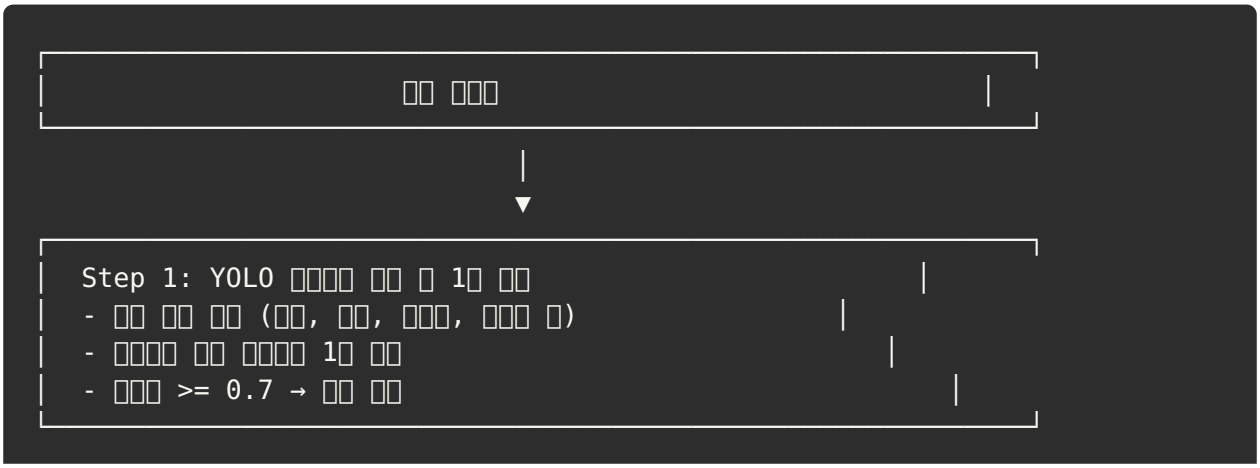


문서 분류 시스템 구현 보고서

작성일: 2026-02-08 버전: 1.0.0

---

1. 프로젝트 개요

1.1 목적

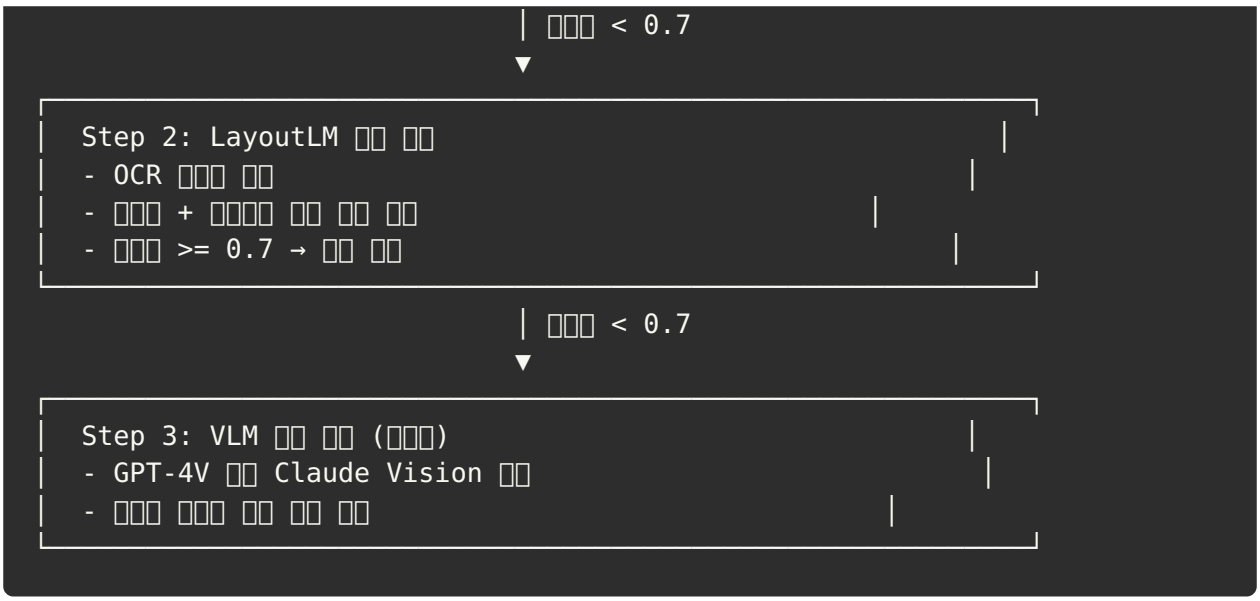
의료/보험 문서를 자동으로 분류하는 3단계 파이프라인 시스템 구축

1.2 분류 대상 문서 (6종)

번호	문서 유형	영문명
1	진단서	Diagnosis
2	소견서	Medical Opinion
3	보험금청구서	Insurance Claim
4	입퇴원확인서	Admission/Discharge Certificate
5	의료비영수증	Medical Receipt
6	처방전	Prescription

1.3 시스템 아키텍처

```
┌─────────────────────────────────────────────────────────────┐
│                         입력 이미지                         │
└─────────────────────────────────────────────────────────────┘
                              │
                              ▼
┌─────────────────────────────────────────────────────────────┐
│  Step 1: YOLO 레이아웃 검출 및 1차 분류                     │
│  - 문서 요소 검출 (도장, 로고, 테이블, 바코드 등)           │
│  - 레이아웃 기반 규칙으로 1차 분류                          │
│  - 신뢰도 >= 0.7 → 분류 완료                                │
└─────────────────────────────────────────────────────────────┘
```



```
                             │ 신뢰도 < 0.7
                             ▼
  ┌──────────────────────────────────────────────────────────┐
  │  Step 2: LayoutLM 정밀 분류                             │
  │  - OCR 텍스트 추출                                      │
  │  - 텍스트 + 레이아웃 기반 정밀 분류                    │
  │  - 신뢰도 >= 0.7 → 분류 완료                           │
  └──────────────────────────────────────────────────────────┘
                             │ 신뢰도 < 0.7
                             ▼
  ┌──────────────────────────────────────────────────────────┐
  │  Step 3: VLM 예외 처리 (미구현)                         │
  │  - GPT-4V 또는 Claude Vision 활용                       │
  │  - 복잡한 문서에 대한 최종 판단                        │
  └──────────────────────────────────────────────────────────┘
```

---

2. 데이터셋 구성

2.1 샘플 데이터 생성

실제 의료 문서 데이터가 없어 합성 데이터를 생성하여 사용

생성 방식

- Python PIL 라이브러리를 사용하여 각 문서 유형별 템플릿 이미지 생성
- 문서 유형별 특징적인 레이아웃 요소 배치:
- 병원 로고, 도장, 서명, 테이블, 바코드, QR코드 등

데이터 규모

구분	문서 수	이미지 수
학습 (train)	6종 × 50개	300개
검증 (val)	6종 × 10개	60개
합계	-	**360개**

2.2 문서 유형별 레이아웃 특성

문서 유형	필수 요소	선택 요소	금지 요소
진단서	도장, 병원로고	서명	테이블, 바코드, QR코드
소견서	도장, 병원로고	서명	테이블, 바코드, QR코드
보험금청구서	테이블, 바코드	도장, 서명, 병원로고	QR코드
입퇴원확인서	도장, 병원로고, 테이블	-	바코드, QR코드
의료비영수증	테이블, 바코드	도장, 병원로고	QR코드
처방전	테이블, QR코드	도장, 병원로고, 서명	바코드

2.3 YOLO 학습 데이터

- 형식: YOLO 형식 (이미지 + 라벨 텍스트 파일)
- 클래스 수: 6개
  `0: stamp (도장) 1: signature (서명) 2: table (테이블) 3: barcode (바코드) 4: qrcode (QR코드) 5: hospital_logo (병원 로고)`

2.4 LayoutLM 학습 데이터

- 형식: 이미지 + OCR 텍스트 + 바운딩 박스
- **OCR** 데이터: 문서 유형별 키워드 기반 Mock OCR 생성
- 진단서: "진단서", "진단명", "상병명", "질병분류", "진단코드" 등
- 소견서: "소견서", "의료소견", "치료소견", "치료경과" 등
- 보험금청구서: "보험금청구서", "청구금액", "계좌번호", "피보험자" 등
- 입퇴원확인서: "입퇴원확인서", "입원일", "퇴원일", "입원기간" 등
- 의료비영수증: "진료비영수증", "본인부담금", "급여", "비급여" 등
- 처방전: "처방전", "처방의약품", "약품명", "투약일수" 등

3. Step 1: YOLO 모델

3.1 모델 구성

- 베이스 모델: YOLOv8n (ultralytics)
- 학습 에포크: 50
- 이미지 크기: 640×640
- 배치 크기: 16

3.2 학습 결과

```
Epoch 50/50 완료
- Box Loss: 0.5765
- Class Loss: 0.3342
- DFL Loss: 0.8823
```

3.3 검증 성능 (Validation)

전체 성능

메트릭	값
mAP50	**99.5%**
mAP50-95	**96.0%**
Precision	98.7%
Recall	98.8%

클래스별 **AP50**

클래스	**AP50**
stamp	99.5%
signature	99.5%
table	99.5%

클래스	**AP50**
barcode	99.5%
qrcode	99.5%
hospital_logo	99.5%

3.4 1차 분류 성능

문서 유형별 정확도

문서 유형	정확도	비고
진단서	10/10 (100%)	-
소견서	0/10 (0%)	진단서로 오분류
보험금청구서	10/10 (100%)	-
입퇴원확인서	10/10 (100%)	-
의료비영수증	10/10 (100%)	-
처방전	10/10 (100%)	-
전체	**50/60 (83.3%)**	-

분석

- 진단서/소견서 혼동: 두 문서 유형이 동일한 레이아웃 특성(도장+병원로고, 테이블 없음)을 가져 YOLO 기반 분류로는 구분 불가
- 해결 방안: Step 2 (LayoutLM)에서 텍스트 기반 분류로 해결

---

4. Step 2: LayoutLM 모델

4.1 모델 구성

- 베이스 모델: microsoft/layoutlmv3-base
- 학습 에포크: 5
- 배치 크기: 2

- 최대 시퀀스 길이: 512
- 학습률: 5e-5 (warmup steps: 30)

4.2 학습 결과

에포크별 성능

Epoch	Train Loss	Eval Loss	Eval Accuracy
1	0.0030	0.0011	100%
2	0.0020	0.0011	100%
3	0.0014	0.0007	100%
4	0.0012	0.0006	100%
5	0.0010	0.0006	100%

4.3 검증 성능

문서 유형별 정확도

문서 유형	정확도
진단서	10/10 (100%)
소견서	10/10 (100%)
보험금청구서	10/10 (100%)
입퇴원확인서	10/10 (100%)
의료비영수증	10/10 (100%)
처방전	10/10 (100%)
전체	**60/60 (100%)**

4.4 핵심 성과

- 진단서/소견서 완벽 구분: Step 1에서 구분 불가했던 두 문서 유형을 텍스트 기반으로 100% 정확하게 분류
- 높은 신뢰도: 대부분의 분류에서 신뢰도 0.99 이상 달성

5. Step 1 vs Step 2 비교

항목	**Step 1 (YOLO)**	**Step 2 (LayoutLM)**
분류 방식	레이아웃 요소 기반	텍스트 + 레이아웃 기반
전체 정확도	83.3%	100%
진단서/소견서 구분	불가능	가능
처리 속도	빠름	상대적으로 느림
적합한 상황	레이아웃이 뚜렷한 문서	텍스트 구분이 필요한 문서

---

6. API 서버

6.1 기술 스택

- 프레임워크: FastAPI
- 서버: Uvicorn
- 문서: Swagger UI (자동 생성)

6.2 엔드포인트

엔드포인트	메서드	설명
`/`	GET	API 정보
`/health`	GET	헬스 체크 및 모델 로드 상태
`/classes`	GET	지원 문서 클래스 목록
`/classify`	POST	단일 문서 분류
`/classify/batch`	POST	배치 문서 분류
`/docs`	GET	Swagger UI

6.3 분류 API 응답 예시

```
{
  "predicted_class": "소견서",
  "confidence": 0.9979,
  "final_step": 2,
  "processing_time": 0.564,
  "step1_result": {
    "predicted_class": "진단서",
    "confidence": 1.0,
    "requires_step2": false
  },
  "step2_result": {
    "predicted_class": "소견서",
    "confidence": 0.9979,
    "all_probabilities": {
      "소견서": 0.9979,
      "진단서": 0.0005,
      ...
    }
  }
}
```

---

7. 프로젝트 구조

```
문서분류기/
├── api/
│   ├── __init__.py
│   └── app.py                  # FastAPI 앱
├── config/
│   └── config.yaml             # 전체 설정
├── src/
│   ├── pipeline.py             # 메인 파이프라인
│   ├── preprocessor/           # 전처리 (기울기 보정 등)
│   ├── step1_yolo/             # YOLO 검출 및 분류
│   ├── step2_layoutlm/         # LayoutLM 분류 및 OCR
│   ├── step3_vlm/              # VLM 예외 처리 (placeholder)
│   └── utils/                  # 유틸리티
├── scripts/
│   ├── train_yolo.py           # YOLO 학습
│   ├── train_layoutlm_simple.py # LayoutLM 학습
│   ├── test_step1_classifier.py # Step 1 테스트
│   ├── test_step2_layoutlm.py   # Step 2 테스트
│   └── test_api.py             # API 테스트
├── data/
│   ├── models/                 # 학습된 모델
```

```
│   ├── yolo_dataset/          # YOLO 학습 데이터
│   └── sample/                # 샘플 데이터
├── run_api.py                 # API 서버 실행
└── requirements.txt           # 의존성
```

---

8. 실행 방법

8.1 환경 설정

```
pip install -r requirements.txt
```

8.2 API 서버 실행

```
python run_api.py
```

8.3 테스트

```
# Step 1 테스트
python scripts/test_step1_classifier.py

# Step 2 테스트
python scripts/test_step2_layoutlm.py

# API 테스트
python scripts/test_api.py
```

---

9. 향후 개선 사항

9.1 단기

- [ ] Step 3 (VLM) 구현: GPT-4V 또는 Claude Vision 연동
- [ ] 실제 OCR 적용: EasyOCR/PaddleOCR 실데이터 학습
- [ ] 전처리 파이프라인 활성화: 기울기 보정, 문서 영역 검출

9.2 중기

- [ ] 실제 의료 문서 데이터 수집 및 학습
- [ ] 모델 경량화 및 추론 속도 최적화
- [ ] Docker 컨테이너화

9.3 장기

- [ ] 추가 문서 유형 지원
- [ ] 다국어 지원
- [ ] 클라우드 배포 (AWS/GCP)

---

10. 결론

3단계 파이프라인 기반 문서 분류 시스템을 성공적으로 구현했습니다.

주요 성과**:** 1. **YOLO** 기반 레이아웃 검출: mAP50 99.5% 달성 2. **LayoutLM** 기반 정밀 분류: 100% 정확도 달성 3. 진단서/소견서 구분 문제 해결: Step 2에서 텍스트 기반 분류로 완벽 구분 4. **REST API** 제공: FastAPI 기반 서비스 API 구현

합성 데이터 기반으로 개발되었으므로, 실제 의료 문서 데이터로 추가 학습 시 성능 검증이 필요합니다.

작성자: Claude Opus 4.5 (Co-authored with BongwooChoi)