폐쇄망 환경 설치 가이드

목적: 인터넷이 차단된 폐쇄망에서 문서 분류 시스템을 구축하는 완전한 가이드

목차

---

1. 사전 준비 (인터넷 환경에서)

1.1 프로젝트 코드 다운로드

```
# GitHub에서 프로젝트 클론
git clone https://github.com/BongwooChoi/doc-classifier.git
cd doc-classifier
```

1.2 Python 패키지 오프라인 다운로드

```
# 오프라인 패키지 저장 디렉토리 생성
mkdir -p offline_packages

# 모든 의존성 패키지를 wheel 파일로 다운로드
pip download -r requirements.txt -d offline_packages

# 추가 패키지 다운로드 (requirements.txt에 없는 것들)
pip download torch torchvision --index-url https://download.pytorch.org/whl/cpu -d
pip download ultralytics -d offline_packages
pip download transformers -d offline_packages
pip download easyocr -d offline_packages
```

```
pip download fastapi uvicorn python-multipart -d offline_packages
pip download pillow numpy opencv-python -d offline_packages
pip download pyyaml tqdm -d offline_packages
pip download accelerate datasets -d offline_packages
```

1.3 Hugging Face 모델 다운로드

```
 # 모델 저장 디렉토리 생성
mkdir -p offline_models/layoutlmv3-base

# Python으로 모델 다운로드
python3 << 'EOF'
from transformers import LayoutLMv3Processor, LayoutLMv3ForSequenceClassification

model_name = "microsoft/layoutlmv3-base"
save_path = "offline_models/layoutlmv3-base"

# 프로세서 다운로드
processor = LayoutLMv3Processor.from_pretrained(model_name, apply_ocr=False)
processor.save_pretrained(save_path)

# 모델 다운로드 (분류용으로 초기화)
model = LayoutLMv3ForSequenceClassification.from_pretrained(
    model_name,
    num_labels=6
)
model.save_pretrained(save_path)

print(f"모델 저장 완료: {save_path}")
EOF
```

1.4 YOLO 사전학습 모델 다운로드

```
 # YOLO 모델 다운로드
python3 << 'EOF'
from ultralytics import YOLO

# YOLOv8n 다운로드 (자동으로 yolov8n.pt 생성)
model = YOLO("yolov8n.pt")
print("YOLO 모델 다운로드 완료: yolov8n.pt")
EOF
```

```
# 모델 파일을 offline_models로 복사
cp yolov8n.pt offline_models/
```

1.5 EasyOCR 모델 다운로드

```
 # EasyOCR 모델 다운로드 (한국어 + 영어)
python3 << 'EOF'
import easyocr

# 한국어+영어 모델 다운로드 (첫 실행 시 자동 다운로드)
reader = easyocr.Reader(['ko', 'en'], gpu=False)
print("EasyOCR 모델 다운로드 완료")

# 모델 위치 확인
import os
easyocr_model_path = os.path.expanduser("~/.EasyOCR/model")
print(f"EasyOCR 모델 경로: {easyocr_model_path}")
EOF

# EasyOCR 모델 복사
mkdir -p offline_models/easyocr
cp -r ~/.EasyOCR/model/* offline_models/easyocr/
```

1.6 다운로드 파일 목록 확인

```
 # 최종 구조 확인
find offline_packages -name "*.whl" | head -20
ls -la offline_models/
ls -la offline_models/layoutlmv3-base/
ls -la offline_models/easyocr/
```

1.7 전체 파일 압축

```
 # 프로젝트 전체를 압축 (패키지 + 모델 포함)
cd ..
tar -czvf doc-classifier-offline.tar.gz doc-classifier/
```

압축 파일에 포함되는 내용: - `doc-classifier/` - 프로젝트 코드 - `doc-classifier/offline_packages/` - Python 패키지 (~2-3GB) - `doc-classifier/offline_models/` - 사전학습 모델 (~1-2GB)

---

2. 폐쇄망으로 파일 이동

2.1 이동할 파일 목록

파일/폴더	크기 (약)	설명
`doc-classifier-offline.tar.gz`	3-5GB	전체 압축 파일

2.2 이동 방법

- USB 드라이브
- 내부 파일 서버
- 승인된 데이터 전송 시스템

---

3. 폐쇄망에서 설치

3.1 압축 해제

```
# 작업 디렉토리로 이동
cd /path/to/workspace

# 압축 해제
tar -xzvf doc-classifier-offline.tar.gz
cd doc-classifier
```

3.2 Python 가상환경 생성 (권장)

```
# 가상환경 생성
python3 -m venv venv

# 가상환경 활성화 (Linux/Mac)
source venv/bin/activate

# 가상환경 활성화 (Windows)
# venv\Scripts\activate
```

3.3 오프라인 패키지 설치

```
# 오프라인 패키지에서 설치 (인터넷 사용 안 함)
pip install --no-index --find-links=offline_packages torch torchvision
pip install --no-index --find-links=offline_packages ultralytics
pip install --no-index --find-links=offline_packages transformers
pip install --no-index --find-links=offline_packages easyocr
pip install --no-index --find-links=offline_packages fastapi uvicorn python-multipa
pip install --no-index --find-links=offline_packages pillow numpy opencv-python
pip install --no-index --find-links=offline_packages pyyaml tqdm
pip install --no-index --find-links=offline_packages accelerate datasets

# 또는 한 번에 설치
pip install --no-index --find-links=offline_packages -r requirements.txt
```

3.4 설치 확인

```
# 주요 패키지 확인
python3 << 'EOF'
import torch
print(f"PyTorch: {torch.__version__}")

import ultralytics
print(f"Ultralytics: {ultralytics.__version__}")

import transformers
print(f"Transformers: {transformers.__version__}")

import easyocr
print("EasyOCR: OK")

import fastapi
print(f"FastAPI: {fastapi.__version__}")

print("\n모든 패키지 설치 완료!")
EOF
```

3.5 오프라인 모델 배치

```
# YOLO 사전학습 모델 복사
cp offline_models/yolov8n.pt ./
```

```
# EasyOCR 모델 복사 (홈 디렉토리로)
mkdir -p ~/.EasyOCR/model
cp -r offline_models/easyocr/* ~/.EasyOCR/model/
```

---

4. 실제 데이터 준비

폐쇄망에서는 합성 데이터가 아닌 실제 의료/보험 문서 이미지를 사용합니다.

4.1 필요한 문서 유형 (6종)

번호	문서 유형	파일명 prefix	권장 수량 (최소)
1	진단서	`diagnosis_`	50장 이상
2	소견서	`opinion_`	50장 이상
3	보험금청구서	`insurance_`	50장 이상
4	입퇴원확인서	`admission_`	50장 이상
5	의료비영수증	`receipt_`	50장 이상
6	처방전	`prescription_`	50장 이상

> 중요: 파일명이 문서 유형 prefix로 시작해야 합니다. LayoutLM 학습 시 파일명에서 문서 유형을 자동 추출합니다.

4.2 이미지 파일 준비

```
# 이미지 형식: JPG 또는 PNG
# 파일명 규칙: {문서유형prefix}_{번호}.jpg
# 예시:
#   diagnosis_0001.jpg, diagnosis_0002.jpg, ...
#   opinion_0001.jpg, opinion_0002.jpg, ...
#   insurance_0001.jpg, insurance_0002.jpg, ...
```

4.3 YOLO 데이터셋 구성

```
# 디렉토리 구조 생성
mkdir -p data/yolo_dataset/images/train
mkdir -p data/yolo_dataset/images/val
```

```
mkdir -p data/yolo_dataset/labels/train
mkdir -p data/yolo_dataset/labels/val
```

4.3.1 이미지 배치

문서 이미지를 train/val로 분할합니다 (권장 비율 8:2).

```
 # 예시: 진단서 50장 중 40장은 train, 10장은 val
cp diagnosis_0001.jpg ~ diagnosis_0040.jpg  data/yolo_dataset/images/train/
cp diagnosis_0041.jpg ~ diagnosis_0050.jpg  data/yolo_dataset/images/val/
# 나머지 문서 유형도 동일하게 분할
```

4.3.2 YOLO 라벨 파일 작성

각 이미지에 대해 문서 요소의 바운딩박스 라벨 파일(`.txt`)을 작성합니다.

YOLO 라벨 형식: `클래스ID x_center y_center width height` (0~1 정규화 좌표)

검출 대상 클래스:

ID	클래스명	설명
0	stamp	도장
1	signature	서명
2	table	테이블
3	barcode	바코드
4	qrcode	QR코드
5	hospital_logo	병원 로고

라벨 파일 예시 (`diagnosis_0001.txt`):

```
 5 0.175000 0.122727 0.225000 0.045455
1 0.662500 0.863636 0.125000 0.027273
0 0.850000 0.836364 0.137500 0.100000
```

> Tip: labelImg, CVAT 등의 라벨링 도구를 사용하면 편리합니다. 라벨링 도구도 사전에 오프라인으로 준비하세요.

4.3.3 data.yaml 작성

```
 cat > data/yolo_dataset/data.yaml << 'EOF'
path: data/yolo_dataset
train: images/train
val: images/val

names:
  0: stamp
  1: signature
  2: table
  3: barcode
  4: qrcode
  5: hospital_logo
EOF
```

4.4 데이터 확인

```
 # 이미지 개수 확인
echo "Train images: $(ls data/yolo_dataset/images/train/*.jpg 2>/dev/null | wc -l)"
echo "Val images: $(ls data/yolo_dataset/images/val/*.jpg 2>/dev/null | wc -l)"

# 라벨 개수 확인 (이미지 수와 일치해야 함)
echo "Train labels: $(ls data/yolo_dataset/labels/train/*.txt 2>/dev/null | wc -l)"
echo "Val labels: $(ls data/yolo_dataset/labels/val/*.txt 2>/dev/null | wc -l)"

# data.yaml 내용 확인
cat data/yolo_dataset/data.yaml
```

4.5 개인정보 비식별화

실제 의료 문서를 사용할 경우 반드시 개인정보를 비식별화해야 합니다.

비식별화 대상	처리 방법
환자 성명	마스킹 또는 가명 처리
주민등록번호	완전 삭제 또는 마스킹
연락처	마스킹
주소	마스킹
의사 성명	가명 처리 (선택)

> 주의: 비식별화 후에도 문서의 레이아웃 구조와 핵심 키워드(진단서, 소견서 등)는 보존되어야 모델 학습에 유효합니다.

---

5. 모델 학습

5.1 Step 1: YOLO 학습

```
# YOLO 학습 실행
python3 scripts/train_yolo.py
```

예상 소요 시간: 30분 ~ 1시간 (CPU 기준)

학습 진행 확인:

```
Epoch 1/50 ...
Epoch 2/50 ...
...
Epoch 50/50 완료!
```

학습 완료 후 모델 위치:

```
runs/detect/train/weights/best.pt
```

5.2 YOLO 모델 복사

```
# 학습된 모델을 지정 경로로 복사
mkdir -p data/models/yolo_document_layout/weights
cp runs/detect/train/weights/best.pt data/models/yolo_document_layout/weights/
```

5.3 Step 1 테스트

```
# YOLO 분류 테스트
python3 scripts/test_step1_classifier.py
```

예상 결과:

```
문서 유형별 정확도:
진단서: 10/10 (100.0%)
```

```
 소견서: 0/10 (0.0%)       ← 진단서로 오분류 (예상됨)
 보험금청구서: 10/10 (100.0%)
 입퇴원확인서: 10/10 (100.0%)
 의료비영수증: 10/10 (100.0%)
 처방전: 10/10 (100.0%)

전체 정확도: 50/60 (83.3%)
```

5.4 Step 2: LayoutLM 학습

LayoutLM은 이미지 + OCR 텍스트로 학습합니다. 학습 시 EasyOCR로 실제 텍스트를 추출하여 사용합니다.

```
 # LayoutLM 학습 실행 (오프라인 모델 사용)
python3 scripts/train_layoutlm_simple.py
```

중요: 오프라인 환경에서는 모델 경로 수정 필요

학습 스크립트 수정 (`scripts/train_layoutlm_simple.py`):

```
 # 기존 (온라인)
# model_name = "microsoft/layoutlmv3-base"

# 수정 (오프라인)
model_name = "offline_models/layoutlmv3-base"
```

또는 스크립트 실행 전에 환경변수 설정:

```
 export TRANSFORMERS_OFFLINE=1
export HF_DATASETS_OFFLINE=1
```

참고: LayoutLM의 문서 유형 라벨

LayoutLM 학습 시 문서 유형은 별도의 라벨 파일 없이 파일명에서 자동 추출됩니다.

파일명 prefix	매핑되는 문서 유형
`diagnosis_`	진단서
`opinion_`	소견서
`insurance_`	보험금청구서

파일명 prefix	매핑되는 문서 유형
`admission_`	입퇴원확인서
`receipt_`	의료비영수증
`prescription_`	처방전

따라서 이미지 파일명만 올바르게 지정하면 됩니다.

예상 소요 시간: 40분 ~ 2시간 (CPU 기준, 데이터 양에 따라 상이)

학습 진행 확인:

```
 Epoch 1/5: eval_accuracy=0.95
Epoch 2/5: eval_accuracy=0.98
...
Epoch 5/5: eval_accuracy=1.0
```

학습 완료 후 모델 위치:

```
 data/models/layoutlm_classifier/best/
```

5.5 Step 2 테스트

```
 # LayoutLM 분류 테스트 (Mock OCR 기반)
python3 scripts/test_step2_layoutlm.py

# 실제 OCR vs Mock OCR 비교 테스트
python3 scripts/test_real_ocr.py
```

실제 데이터에서의 예상 결과: - 실제 문서는 합성 데이터보다 OCR 품질이 높아 더 나은 성능을 기대할 수 있음 - 진단서/소견서와 같이 레이아웃이 유사한 문서도 텍스트 기반으로 정확하게 구분

---

6. 테스트 및 검증

6.1 통합 파이프라인 테스트

```
 # Step 1 + Step 2 통합 테스트
python3 scripts/test_pipeline_integration.py
```

6.2 개별 이미지 테스트

```
 python3 << 'EOF'
import sys
sys.path.insert(0, '.')

from src.step1_yolo.detector import YOLODetector
from src.step1_yolo.classifier import YOLOClassifier

# YOLO 분류기 로드
detector = YOLODetector(
    model_path="data/models/yolo_document_layout/weights/best.pt"
)
classifier = YOLOClassifier(detector=detector)

# 테스트 이미지 분류
result = classifier.classify("data/yolo_dataset/images/val/diagnosis_0040.jpg")
print(f"예측: {result['predicted_class']}")
print(f"신뢰도: {result['confidence']:.2f}")
EOF
```

---

7. API 서버 실행

7.1 서버 시작

```
 # API 서버 실행
python3 run_api.py
```

출력:

```
 ============================================================
문서 분류 API 서버 시작
============================================================

접속 URL: http://localhost:8000
API 문서: http://localhost:8000/docs
```

7.2 API 테스트

터미널 1: 서버 실행 유지

터미널 2: 테스트 실행

```
 # API 테스트 스크립트
python3 scripts/test_api.py
```

7.3 curl로 테스트

```
 # 헬스 체크
curl http://localhost:8000/health

# 단일 파일 분류 (실제 OCR 사용 - 기본값)
curl -X POST "http://localhost:8000/classify" \
  -F "file=@data/yolo_dataset/images/val/diagnosis_0001.jpg"
```

7.4 브라우저에서 테스트

1. 브라우저에서 `http://localhost:8000/docs` 접속
2. `/classify` 엔드포인트 클릭
3. "Try it out" 클릭
4. 이미지 파일 업로드
5. "Execute" 클릭
6. 결과 확인

---

8. 트러블슈팅

8.1 패키지 설치 오류

문제: `No matching distribution found`

```
# 해결: 누락된 패키지를 인터넷 환경에서 추가 다운로드
pip download [패키지명] -d offline_packages
```

8.2 CUDA/GPU 관련 오류

문제: `CUDA not available`

```
# 해결: CPU 모드로 실행 (기본 설정)
# config/config.yaml에서 확인
yolo:
  device: "cpu"  # 또는 "auto"
```

8.3 모델 로드 오류

문제: `Can't load model from 'microsoft/layoutlmv3-base'`

```
# 해결: 오프라인 모델 경로 사용
# scripts/train_layoutlm_simple.py 수정
model_name = "offline_models/layoutlmv3-base"
```

8.4 메모리 부족

문제: `Out of memory`

```
# 해결: 배치 크기 줄이기
# scripts/train_layoutlm_simple.py에서
per_device_train_batch_size=1  # 2 → 1로 변경
```

8.5 EasyOCR 모델 오류

문제: `Model file not found`

```
# 해결: EasyOCR 모델 경로 확인
ls ~/.EasyOCR/model/

# 모델 파일 재복사
cp -r offline_models/easyocr/* ~/.EasyOCR/model/
```

---

9. 폴더 구조 최종 확인

```
doc-classifier/
├── offline_packages/          # 오프라인 Python 패키지
├── offline_models/            # 사전학습 모델
│   ├── layoutlmv3-base/
│   ├── easyocr/
│   └── yolov8n.pt
├── data/
│   ├── models/                # 학습된 모델
│   │   ├── yolo_document_layout/weights/best.pt
│   │   └── layoutlm_classifier/best/
│   ├── yolo_dataset/          # YOLO 학습 데이터
│   └── sample/                # 샘플 데이터
├── src/                       # 소스 코드
├── scripts/                   # 학습/테스트 스크립트
├── api/                       # FastAPI 앱
├── config/                    # 설정 파일
├── run_api.py                 # API 서버 실행
└── requirements.txt           # 의존성 목록
```

---

10. 체크리스트

사전 준비 (인터넷 환경)

- [ ] 프로젝트 코드 다운로드
- [ ] Python 패키지 오프라인 다운로드
- [ ] LayoutLMv3 모델 다운로드
- [ ] YOLOv8n 모델 다운로드
- [ ] EasyOCR 모델 다운로드
- [ ] 전체 압축

폐쇄망 설치

- [ ] 압축 해제
- [ ] 가상환경 생성
- [ ] 오프라인 패키지 설치
- [ ] 모델 파일 배치

데이터 준비

- [ ] 실제 문서 이미지 수집 (6종, 유형당 50장 이상)
- [ ] 개인정보 비식별화
- [ ] 파일명 규칙에 맞게 이름 변경 (`{유형prefix}_{번호}.jpg`)
- [ ] train/val 분할 (8:2)
- [ ] YOLO 라벨 파일 작성 (바운딩박스 어노테이션)

학습 및 테스트

- [ ] YOLO 학습 완료
- [ ] YOLO 테스트 통과
- [ ] LayoutLM 학습 완료
- [ ] 실제 OCR 테스트 통과 (`test_real_ocr.py`)
- [ ] API 서버 실행 확인

작성일: 2026-02-10 작성자: Claude Opus 4.6 (Co-authored with BongwooChoi)