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 offl
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-multipart
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. 데이터셋 생성

4.1 샘플 데이터 생성

```
# 샘플 이미지 및 어노테이션 생성
python3 scripts/generate_sample_data.py
```

출력 확인:

```
data/sample/
├── train/
│   ├── images/      # 48개 이미지 (6종 × 8개)
│   ├── annotations/ # 48개 JSON
│   └── labels.tsv
└── test/
    ├── images/      # 12개 이미지 (6종 × 2개)
    ├── annotations/ # 12개 JSON
    └── labels.tsv
```

4.2 YOLO 데이터셋 생성

```
# YOLO 형식 데이터셋 생성
python3 scripts/generate_yolo_dataset.py
```

출력 확인:

```
data/yolo_dataset/
├── images/
│   ├── train/  # 300개 이미지
│   └── val/    # 60개 이미지
├── labels/
│   ├── train/  # 300개 라벨
│   └── val/    # 60개 라벨
└── data.yaml   # YOLO 설정 파일
```

4.3 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)"

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

---

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

```
# 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
```

예상 소요 시간**:** 40분 ~ 1시간 (CPU 기준)

학습 진행 확인**:**

```
Epoch 1/5: eval_accuracy=1.0
Epoch 2/5: eval_accuracy=1.0
...
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
python3 scripts/test_step2_layoutlm.py
```

예상 결과:

```
문서 유형별 정확도:
  진단서: 10/10 (100.0%)
  소견서: 10/10 (100.0%)    ← Step 1과 달리 정확하게 분류
  보험금청구서: 10/10 (100.0%)
  입퇴원확인서: 10/10 (100.0%)
  의료비영수증: 10/10 (100.0%)
  처방전: 10/10 (100.0%)

전체 정확도: 60/60 (100.0%)
```

---

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
curl -X POST "http://localhost:8000/classify?use_mock_ocr=true" \
  -F "file=@data/yolo_dataset/images/val/diagnosis_0040.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

학습 및 테스트

- [ ] 샘플 데이터 생성
- [ ] YOLO 데이터셋 생성
- [ ] YOLO 학습 완료
- [ ] YOLO 테스트 통과
- [ ] LayoutLM 학습 완료
- [ ] LayoutLM 테스트 통과
- [ ] API 서버 실행 확인

작성일: 2026-02-08 작성자: Claude Opus 4.5