일상 속의 OCR

금융·교육·행정 등 다양한 분야의 활용 사례 및 성능 비교

OCR 개요

OCR의 정의 및 원리

OCR(Optical Character Recognition)은

이미지나 스캔된 문서 속의 문자(글자)를 인식하여 디지털 텍스트로 변환하는 기술

💡 예시 : 스캔된 영수증 → “총 금액 13,200원” 텍스트 추출

원리는 **이미지에서 문자 패턴을 인식하고 이를 학습된 문자 데이터와 비교해 디지털화**하는 구조로

딥러닝 기반 OCR은 **글자 모양(CNN)** + 문맥 흐름(LSTM/Transformer)을 함께 고려 함

OCR 활용 분야

분야별 활용 사례 - 금융

활용 목적

- 종이 문서 입력 자동화
- 신분증, 통장 사본, 영수증 등에서 데이터 자동 추출

KakaoBank/Toss	OCR로 신분증 사진을 촬영하면 자동으로 이름·주민번호·주소 인식 → 회원가입 정보 자동 입력
KB 국민은행	통장 OCR을 통해 계좌번호·예금주 자동 인식 → 비대면 계좌 개설 시간 단축
카드사	영수증 스캔 → 금액, 가맹점, 날짜를 자동 인식해 지출 내역 자동 분류

→ 입력시간 단축, 인적 오류 최소화, 비대면 인증 및 자동화 확대

OCR 활용 분야

분야별 활용 사례 - 교육

활용 목적

- 시험지, 과제, 필기 내용 자동 디지털화
- 학습 데이터 수집 및 분석

에듀테크 서비스(Class123, 뤼이드)	수기 답압 OCR 인식으로 자동 채점 시스템 구현
대학교 시험 관리 시스템	답안지 스캔 후 OCR로 학번·이름 자동 인식
강의 필기 앱(Nebo, GoodNotes)	손글씨를 OCF로 변환 → 검색 가능한 텍스트로 저장

→ 교사·학생의 채점 시간 절약, 디지털 학습 콘텐츠 자동 생성

OCR 활용 분야

분야별 활용 사례 - 행정

활용 목적

- 종이 서류 디지털 전환
- 민원 처리 자동화 및 전자문서화

정부24·국세청	제출 서류(주민등록증, 사업자등록증 등) 자동 인식 → 전산 시스템 연동
지방자치단체	종이 민원 서류 OCF → 행정문서 자동 분류 및 전자화
경찰청/출입국관리소	여권 및 외국인 등록증 OCR로 신원 확인 자동화

→ 공공문서 검색·보관 효율 극대화, 민원 처리 속도 향상(수기입력 제거)

OCR 활용 분야

분야별 활용 사례 - 의료

활용 목적

- 환자 기록, 처방전, 검사 결과 자동 입력
- EMR(Electronic Medical Record) 시스템과 연계

삼성서울병원/세브란스	진료기록 OCR → 진료 요약 및 보험청구 자동화
보험사(한화·삼성화재)	병원 영수증 OCRF → 진단서, 수납내역 자동 인식 후 보험청구 자동화
의약품 라벨 인식	약품명·용량 OCR → 투약 관리 정확도 향상

→ 의무기록 전산화로 오류 감수, 환자 데이터 분석·AI 진단 지원 기반 마련

OCR 활용 분야

분야별 활용 사례 - 법률

활용 목적

- 방대한 문서(판례, 계약서) 자동 디지털화
- 검색 및 키워드 분석 자동화

법원 전자소송 시스템	종이 판결문 OCR → 전자문서 변환 후 검색 가능
로앤컴퍼니(로톡)	계약서 OCR 분석 → 주요 조항 자동 추출
로펌 내부시스템	클라이언트 문서 OCR → 분류 및 텍스트 검색 자동화

→ 수작업 문서 검토 시간 절감, 리컬테크(LegalTech)기반 자동화 분석 실현

OCR 활용 분야

분야별 활용 사례 - 물류 및 유통

활용 목적

- 운송장, 송장번호, 제품 라벨 자동 인식
- 입출고, 재고 관리 자동화

CJ대한통운/쿠팡로지스틱스	송장 OCR → 운송장 번호 자동 인식으로 분류기 제어
공항·항만 물류시스템	컨테이너 번호 OCR → 자동 통관·입출항 처리
리테일 매장	상품 바코드 인식 실패 시 OCR로 제품명 대체 인식

→ 물류 속도 향상, 분류 정확도 향상 및 인건비 절감

OCR 처리 과정

1️⃣ 이미지 입력

- 스캐너·카메라로 문서 이미지 획득
- 해상도·조명 품질이 인식률 결정

⇩

2️⃣ 전처리 (Preprocessing)

- 이진화, 노이즈 제거, 기울기 보정
- 깨끗한 입력이 정확도 향상 핵심

⇩

3️⃣ 문자 검출 (Detection)

- 글자 영역 식별 (EAST, CRAFT 알고리즘)
- 배경과 문자 분리

⇩

4️⃣ 문자 인식 (Recognition)

- CNN·LSTM 기반 문자 판독
- 다양한 폰트·손글씨 인식

⇩

5️⃣ 후처리 (Postprocessing

- 언어모델로 문맥 보정
- 띄어쓰기·맞춤법 교정

OCR 인식률 평가 방법

평가의 목적

OCR 모델이 **얼마나 정확하게 문자(텍스트)를 인식했는지**를 수치로 표현하기 위해서

기본 개념

지표	의미	계상 방식
Accuracy (정확도)	전체 문자 중 올바르게 인식된 비율	정확히 인식된 문자 수 ÷ 전체 문자 수 × 100
Precision (정밀도)	인식된 문자 중 실제로 맞은 비율	TP ÷ (TP + FP)
Recall (재현율)	실제 문자 중 인식에 성공한 비율	TP ÷ (TP + FN)
Levenshtein Distance (편집 거리)	원본과 결과의 유사도(삽입·삭제·교체 최소 횟수 기준)	1 - (편집거리 ÷ 원본 문자 수)

📌 TP(True Positive): 올바르게 인식된 문자, FP(False Positive): 잘못 인식된 문자, FN(False Negative): 인식되지 못한 문자.

OCR 인식률 평가 방법

예시 코드(Python)

```python
import Levenshtein

gt = "국민은행 1002-123-456789"    # 정답 텍스트 (ground truth)
pred = "국민은해 1002-123-456788"  # OCR 결과

acc = 1 - Levenshtein.distance(gt, pred) / len(gt)
print(f"OCR 인식률: {acc * 100:.2f}%")
```

출력 예시 : OCR 인식률: 96.43%

OCR 인식률 결과 및 원본 비교 그래프

실험 개요

목적	서로 다른 OCR 엔진의 인식 정확도 비교
대상	동일 이미지(sample.png) 및 PDF(sample.pdf)
모델	Tesseract, OCR.space API, PyMuPDF+Tesseract
평가지표	Levenshtein Distance 기반 Accuracy (%)

OCR 인식률 결과 및 원본 비교 그래프

실험 결과(OCR 인식 결과 -1)

* 원본 이미지(sample.png)의 텍스트

안녕하세요. OCR 인식 테스트 입니다.
이 문장은 Tesseract와 OCR.space의 성능을 비교합니다.
Accuracy test with Korean + English
MIXED text.

1. Tesseract 결과

```
============================================================
🧠 이미지 내 텍스트 (Tesseract)
============================================================
OFS SEAS. OCR GIA! IAS SUCH.

0| ZAYE Tesseract2t OCR.space2| SSS
비교합니다.

Accuracy test with Korean + English
MIXED text.

➡️정확도: 68.29%
```

2. OCR.space 결과

```
============================================================
☁️이미지 내 텍스트 (OCR.space)
============================================================
안녕하세요. OCR 인식 테스트 입니다.
이 문장은 Tesseract와 OCR.space의 성능을
비교합니다.
Accuracy test with Korean + EngIish
MIXED text.

➡️정확도: 91.30%
```

OCR 인식률 결과 및 원본 비교 그래프

실험 결과(OCR 인식 결과 - 2)

* 원본 파일(sample.pdf)의 텍스트

OCR 비교 실험용 문서입니다.
이 문장은 PDF 내부의 텍스트로 존재합니다.

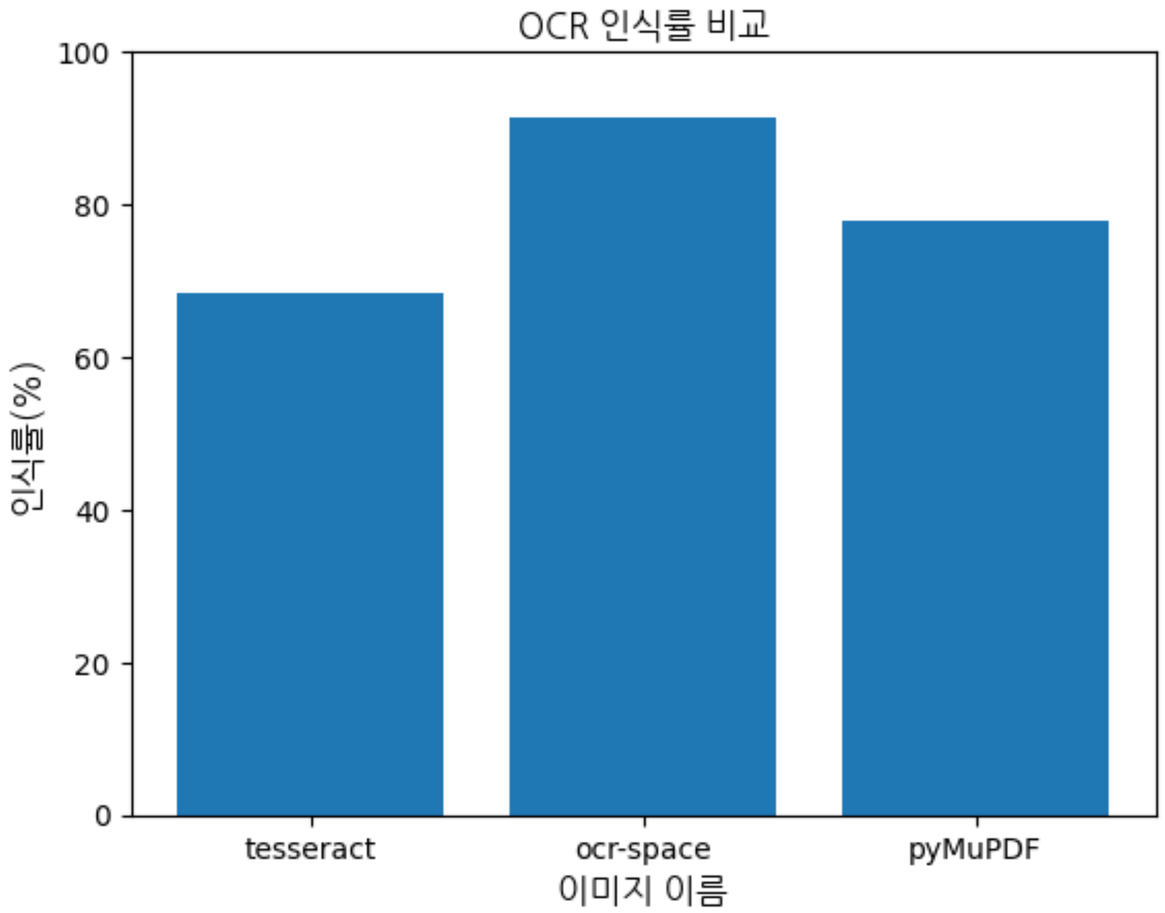
안녕하세요. OCR 인식 테스트 입니다.
이 문장은 Tesseract와 OCR.space의 성능을 비교합니다.
Accuracy test with Korean + English MIXED text.

1. PyMuPDF+ Tesseract 결과

```
==================================================
📄 PDF 텍스트 및 이미지 내 텍스트 (PyMuPDF + Tesseract)
==================================================
OCR 비교 실험용 문서입니다.
이 문장은 PDF 내부의 텍스트로 존재합니다.

[이미지 내 텍스트]
OFS SEAS. OCR GIA! IAS SUCH.

O| ZAYE Tesseract2t OCR.space2| SSS
비교합니다.

Accuracy test with Korean + English
MIXED text.

➡️정확도: 77.90%
```

OCR 인식률 결과 및 원본 비교 그래프

실험 결과(인식률 비교 그래프)



결론 및 향후 발전 방향

📝 결론

- OCR.space API가 Tesseract보다 더 높은 인식률(약 2~3% 차이)을 보임
- **한·영 혼합 텍스트, 문장부호, 띄어쓰기** 인식에서 Tesseract는 오차 발생
- PyMuPDF + Tesseract 조합은 **PDF 내 텍스트 추출에는 우수**, 하지만 **이미지 내 문자 인식 정확도는 낮음**
- **전반적으로 OCR.space가 노이즈 처리 및 문맥 인식 면에서 우수**

🌟 향후 발전 방향

- **다중 OCR 혼합 시스템 구성** → Tesseract(로컬 속도) + OCR.space(클라우드 정확도) 결합
- **이미지 전처리 강화** → 대비 향상, 기울기 보정, 이진화 등을 추가로 적용해 인식률 향상 가능
- **후처리 알고리즘 적용** → Levenshtein 기반 오타 교정 및 언어모델로 문맥 보정
- **데이터셋 확장 및 Fine-tuning** → 한글·영문 혼합, 손글씨 포함 데이터로 모델 학습 개선
- **실시간 OCR 응용** → 모바일 카메라 기반 문서 인식 자동화로 확장 가능