Parser 실험 설명서

1. 기본 세팅

- conda activate pystudy_env
- pip install -U pip
- pip install -r requirements.txt

2. 기본 실행 방법

• 실행:

bash 복사

```
python test.py --pdf_dir ./pdf_dir --out_dir ./outputs \
  --parsers unstructured,pymupdf,pdfminer,pdfplumber,llamaindex
```

현재 ./pdf_dir은 my_pdfs로 ./outputs는 parser_outputs로 바꾸면 됨

3. 출력물 설명

A) JSONL (원문 파싱 결과)

- 경로: ./outputs/<원본파일명>.<parser>.jsonl
- 스키마:

json 복사

```
{
  "type": "paragraph | heading | table | list | footnote | title | raw",
  "text": "<추출 텍스트>",
  "page": <페이지 번호(1-base)>,
  "meta": { "raw_type": "...", "heuristic": "...", "ocr": "...", "src": "..." }
}
```

- 용도: 디버깅, 특정 페이지 품질 점검, 청킹/정규화 파이프라인 입력

B) CSV 리포트 3종

1. summary.csv — **모든 실행 결과(전수)**
    - **주요 칼럼 의미**
        - parser: 파서 이름
        - pdf: 원본 PDF 파일명

- elapsed_sec: 전체 처리 시간(초)
- jsonl: 결과 JSONL 경로
- n_elements: 추출된 전체 요소 개수
- n_tables: **표 탐지 개수** (표-heavy 문서에서 중요)
- n_headings: 제목/헤딩 탐지 개수
- n_paragraphs: 문단 개수
- total_tokens: 추출 텍스트 토큰 총합(대략적인 정보량)
- avg_chunk_tokens: 청크 평균 토큰 길이
- clause_heading_ratio: 금융 문서 조항("제 n 조") 패턴 비율(헤딩 대비)
- total_pages: PDF 전체 페이지 수
- page_coverage: **페이지 커버리지**(추출에 등장한 페이지 비율, 0~1)
- order_breaks: 페이지 순서 역전 횟수(레이아웃 깨짐 신호)
- duplicate_ratio: 중복 청크 비율(0~1)
- noise_ratio: 너무 짧거나 특수문자 위주 청크 비율(0~1)
- std_chunk_tokens: 청크 길이 표준편차(길이 분포 안정성)
- elapsed_per_page: 페이지당 평균 처리 시간(초)
- avg_table_rows: 표 행수 평균(근사)
- numeric_cell_ratio: 표의 숫자 셀 비율(근사, 0~1)
- error: 실패 시 에러 메시지

2. winners_by_pdf.csv — **PDF별 1위 파서**
    - 각 PDF마다 스코어가 가장 높은 파서 1개를 뽑아 요약
3. overall_ranking.csv — **파서별 평균 랭킹**
    - 파서 단위로 score, elapsed_sec, page_coverage 등 주요 지표 평균/합계를 집계

5. 스코어(score) 산식

summary.csv의 각 행(row)에 대해 아래 가중합을 산출합니다(필요 시 조정 가능).

```
score =
  + 2.0 * n_tables
  + 1.5 * clause_heading_ratio
  + 0.5 * n_headings
  + 1.0 * page_coverage
  + 0.5 * numeric_cell_ratio
  - 0.2 * elapsed_per_page
  - 0.5 * noise_ratio
  - 0.5 * duplicate_ratio
```

- **가점**: 표/조항/헤딩/커버리지/숫자셀
- **감점**: 느림(페이지당), 노이즈/중복
- 표가 많은 금융 문서에서 **실무적으로 중요한 요소**를 우선시하도록 설계됨