한글 파싱 실험 101

한글 파싱의 어려운 점

1. 독자적인 바이너리 포맷 (`OLE Compound File` , `HWP Binary`)
2. 압축/암호화 옵션
3. 공개 표준이 아니기 때문에 파싱에 대한 문서화가 제한적

방법론

1. `HWP > docx` 변환 후 파싱

장점

- 한글프로그램으로 HWP 파일을 Word로 변환
- Word 변환 이후 부터는 쉬움

단점

- 한컴오피스 설치 필요 (수작업 or 매크로)
- 포맷 변환 시 일부 기능 손실 (특수 스타일, 머리글/바닥글 등)
- 대량 자동처리 어려워짐

2. `HWP Binary` 직접 파싱

HWP 포맷 바이너리 구조를 분석해 Python으로 해석하는 방법

장점

- 원본 데이터를 획득하므로 자유도가 매우 높음

단점

- 난이도가 어려워 성공한 사람이 없음
- 포맷 규격 문서 (한글 5.0 포맷 PDF 문서) 일부만 공개

프로세스

- Olefile + 수작업 파싱

- OLE stream 구조 탐색
- Body Text/ Section- stream 열어 바이너리 해석
- zlib 압축 해제 후 struct.unpack으로 레코드 단위 분석
  시행착오가 많아지게 되는 해석 노가다

3. `PDF` 변환 후 `OCR`

4. 윈도우 COM 자동화

Window 환경에서 한컴오피스를 COM 개체로 제어

장점

- 자동화 가능

단점

- 윈도우 환경
- 한컴오피스 라이선스 필요
- 자동화 스크립트 난이도
- 공식문서의 부족함 (개발자 가이드가 존재하지만 최신버전은 배포가 제한적)
- 버전에 따라 개발해야되는 불편함
- 참고: 보안 모듈 등록 필수

5. Linux에서의 변환

`wine` 으로 한컴 실행하여 변환 시도

단점

- 리눅스 활용의 번거로움

6. 시행착오 사례

1. `pyhwp` : 압축된 section 스트림 글자 깨져서 나옴
2. `olefile` : 헤더 오프셋 해석 오류
3. `ocr` : 텍스트 인식률 70% 이하
4. COM 자동화: 한컴 버전에 따라 호환성 다름

7. 기타

1. RUST/Python 혼합

Rust 로 바이너리 파서 작성, python 에서 호출??

2. Java Apache POI 활용

Java 코드로 파싱 후 Python 과 연동

3. Go 프로그램 작성

- OLE/Compound parsing 에 강점 있는 라이브러리
- mscfb 같은 라이브러리 활용해 stream 추출 쉬워질 수 있음
- Go 할 줄 모름

방법론 로드맵

1. HWP > docx 변환 시도
2. COM 자동화 스크립트 작성 시도
3. pyhwp + olefile : 바이너리 직접 파싱 시도
4. PDF + OCR 시도