🗓️ [HUFS-LAI ML 2025-2 Assignment 6] OCR 기반 캡처 이미지 일정 누적 등록 도구 최종 보고서

1. 프로젝트 개요 및 문제 정의

1.1. 무엇을 만들었나 (프로젝트 주제)

본 프로젝트는 메신저(카카오톡, Slack 등)에서 공유된 **캡처 이미지** 내의 일정 정보를 자동으로 인식하고 추출하여, 여러 이미지에서 파싱된 일정을 **하나의 iCalendar 파일(.ics)**로 통합 및 누적하여 제공하는 **CLI (Command Line Interface) 도구**입니다[1][1].

1.2. 어떤 문제를 해결했나 (문제 정의)

현대 디지털 커뮤니케이션 환경에서 일정 공유는 주로 텍스트나 이미지 캡처 형태로 이루어집니다. 본 프로젝트는 다음과 같은 비효율성을 해소하고자 합니다[2]:

- **수동 입력 오류 해소:** 이미지 속의 날짜, 시간 정보를 캘린더에 **수동으로 입력**할 때 발생하는 오기입 문제를 제거합니다.
- **일정 파편화 통합:** 여러 장의 캡처 이미지에 흩어져 있는 일정을 한 번의 처리로 **하나의 .ics 파일에 통합**하여 캘린더 가져오기(Import)를 단일화합니다[3].

2. 진행 과정 (Implementation Flow)

2.1. 주제 선정 및 문제 정의

초기에는 Slack 텍스트 봇을 통한 실시간 연동을 목표로 했으나, 외부 서버(ngrok)와 Google Calendar API의 복잡한 인증 및 연결 안정성 문제로 인해 구현이 지연되었습니다. 이에 **ML 모델(OCR)의 핵심 파이프라인 시연**과 **유틸리티 기능(ICS 파일 생성)**에 집중하기 위해 주제를 **이미지 기반의 CLI 도구**로 변경했습니다.

2.2. 데이터 수집 및 분석

- **데이터:** 카카오톡 및 Slack 채널의 **실제 일정 공지 캡처 이미지**를 테스트 데이터로 사용했습니다.
- **분석:** 캡처 이미지에는 OCR의 정확도를 저해하는 다양한 요소(말풍선, 배경색, 크기, 낮은 해상도, 한국어 폰트의 복잡성)가 포함되어 있어, 단순 정규식으로는 높은 정확도를 기대하기 어렵다고 분석했습니다.

2.3. ML 모델 학습 및 평가 (Tesseract OCR)

- **ML 모델:** **Tesseract OCR Engine (v4.x)**을 핵심 모델로 채택했습니다[4].
- **학습:** Tesseract는 이미 학습된 모델이므로 별도의 추가 학습 과정 없이, **한글 언

어 팩(tesseract-ocr-kor)**을 설치하여 한국어 인식 환경을 구축했습니다[5].

- **평가:** 테스트 결과, 간단하고 깔끔한 텍스트 배열은 잘 인식했으나, 배경색이 있거나 말풍선 안에 있는 텍스트는 '4110ㅜ' [6666], ' L 38094896641' [7] 등 의미 없는 문자열(OCR 노이즈)로 변환되는 한계가 명확하게 확인되었습니다.

3. 모델을 서비스로 만든 구조 (시스템 구조)

3.1. 시스템 구조 (CLI 아키텍처)

본 도구는 서버 구축 없이 Google Colab의 I/O 기능을 활용하여 **순차적인 명령어 실행 흐름(CLI)**을 따릅니다[8888].

1. **사용자 입력:** google.colab.files.upload() 함수를 통해 이미지 파일을 업로드합니다[999].
2. **ML 모델 (Tesseract):** 업로드된 이미지 파일을 받아 Tesseract OCR Engine이 **텍스트 문자열**로 변환합니다[10].
3. **파싱 로직:** Python의 re 모듈이 OCR 텍스트에서 날짜/시간 패턴을 파싱하여 datetime 객체로 변환합니다[11111111].
4. **ICS 생성/누적:** ics 라이브러리의 Calendar 객체에 추출된 이벤트(Event)를 누적합니다[12121212].
5. **결과 출력:** 모든 이미지 처리가 완료되면 files.download()를 통해 **accumulated_schedule_tesseract.ics** 파일을 사용자에게 전달합니다[13131313].

3.2. 핵심 파싱 로직 상세 (parse_schedule_python)

- **날짜 추출 정규식:** r'(₩d{1,4}[년₩.]?₩s*₩d{1,2}[월₩.]?₩s*₩d{1,2}일?)₩s*([가-힣]{1,2}요일)?(?:...)?' 와 같이 연, 월, 일 단위의 날짜 패턴을 유연하게 인식합니다.
- **종일 이벤트 처리:** 시간 정보가 추출되지 않은 경우 (hour_str이 False인 경우), 이벤트는 **종일 이벤트 (is_all_day = True)**로 자동 설정되며, 시작 시각은 00:00으로, 종료 시각은 **다음 날 00:00**으로 설정됩니다[14]. 이는 ICS 표준에서 종일 이벤트를 정확히 표현하기 위한 필수 처리입니다.
- **제목 설정:** 텍스트에서 **5자 이상 50자 미만**인 첫 번째 유효한 줄을 일정 제목으로 지정하여, OCR 노이즈를 포함하는 짧은 문자열을 필터링했습니다.

4. 실제 사용 결과 및 검증

4.1. 5회 이상 사용 기록 및 결과 분석

process_multiple_uploads_final() 함수를 반복 실행하여 여러 이미지 파일(KakaoTalk, 화면 캡처)을 처리했습니다.

처리 이미지 수	누적된 최종 일정 수	ICS 파일명	주요 추출 제목
1회차 (2회 누적) [15]	2개	accumulated_schedule_tesseract (1).ics	L 38094896641 24학번 S.. [16], 안녕하십니까, 학우 여러분. [17]
2회차 (3회 누적) 181818	3개	accumulated_schedule_tesseract (4).ics [19]	4:10 2 [2020], 4102 [21]
3회차 (3회 누적) 222222	3개	accumulated_schedule_tesseract (2).ics [23]	6 # .42명의 멤버 *4개의 탭 -에시 [24], 4102 all 수 00 [25]

- **누적 검증:** 실행 로그에서 **[현재 누적된 일정 수: X]**가 증가하고, 최종적으로 다운로드된 .ics 파일에 **다수의 BEGIN:VEVENT 블록**이 포함되어 **누적 기능은 성공**했습니다[26].
- **날짜 검증:** DTSTART:20261002T000000Z [27]와 같이 날짜 정보(2026년 10월 2일)가 추출되었고, 시간 정보가 없어 **종일 이벤트**로 처리되었음이 DTEND 필드(20261003T000000Z)에서 확인됩니다[28].

5. 배운 점 및 개선 방향

5.1. 배운 점

- **ML 파이프라인의 민감도:** 텍스트 기반 NER보다 **OCR 기반 파이프라인**이 이미지 품질과 포맷에 **훨씬 더 민감하다**는 것을 확인했습니다. OCR 단계의 낮은 정확도가 후속 파싱 단계의 성공률을 결정하는 병목 지점임을 파악했습니다.
- **환경 문제 해결 능력:** Slack 봇 개발 과정에서 발생했던 SyntaxError (U+00A0) [29], ModuleNotFoundError[30], 그리고 **ngrok URL 고착화 현상**을 포트 변경, 토큰 재설

정 등의 기술적 방법으로 해결하면서 문제 진단 및 우회 능력을 배양했습니다.

5.2. 개선 방향 (향후 계획)

1. **OCR 엔진 업그레이드:** Tesseract 대신 **Google Cloud Vision API**를 사용하여 OCR 정확도를 획기적으로 개선하고, 복잡한 메신저 UI 이미지에서도 텍스트를 안정적으로 추출할 수 있도록 기능을 확장할 것입니다.
2. **LLM 기반 파싱 도입:** 정규 표현식(re 모듈)의 경직성을 극복하고 OCR 노이즈가 포함된 문장도 유연하게 해석하기 위해, **LLM(Large Language Model)**을 도입하여 일정 요약 및 개체명 추출 기능을 통합할 것입니다.
3. **UI/UX 개선:** CLI 방식 대신 웹 인터페이스(Web UI)를 구축하여 사용자 편의성을 높이고, .ics 파일 다운로드 외에 Google Calendar/Outlook에 직접 일정을 등록하는 API 연동 기능을 추가할 계획입니다.