TASTEAI : 맛집 추천 서비스

빅데이터기반분산-딥러닝혁신프로젝트 최종 발표

팀원 :

32202698 오유석

32204041 정다훈

LIST OF CONTENTS

문제상황 정의

1. 지도 기반 정보의 한계

- 기존의 맛집 탐색은 대부분 포털 지도나 네이버 플레이스에 의존 → 사용자가 직접 검색어를 입력하고 원하는 정보를 찾아야 하는 **수동적 구조**
- 지도에 노출된 정보는 **일부 인기 식당에 편중**되어 있어, 실제 이용자들의 다양한 상황을 반영하기 어려움

2. 정보 탐색의 비효율성

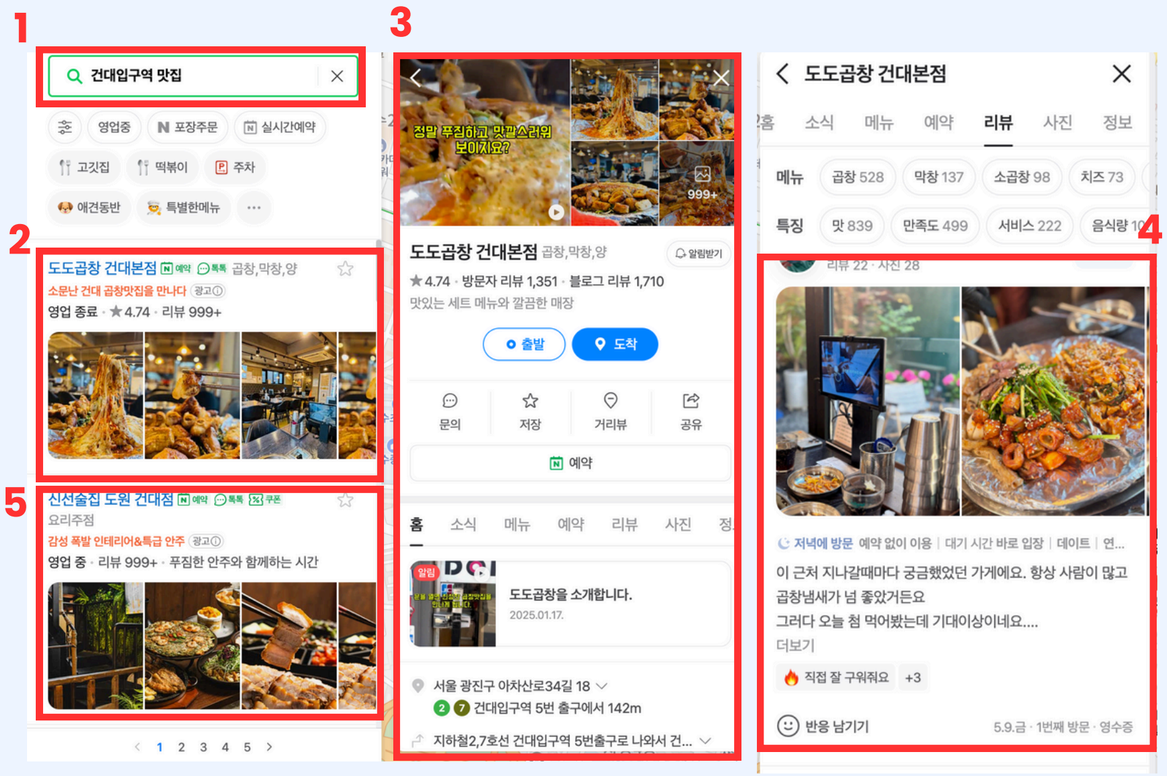
- 사용자들은 목적에 맞는 식당을 찾기 위해 **여러 앱과 웹페이지를 오가며 리뷰를 비교**하고,검색 키워드도 반복적으로 수정해야 하는 번거로움을 겪음
- 이는 시간도 오래 소요되고, **사용자의 피로도를 높이는 주요 원인**

3. 낯선 지역에서의 선택 어려움

- 처음 방문한 지역에서는 **신뢰할 만한 정보를 찾기 더 어렵고**, 리뷰의 진위 여부를 판단하기 힘듦

→ **사용자에게 적절한 맛집을 추천해주는 'AI 기반 맛집 추천 서비스' 개발**

솔루션 제안 (데이터 크롤링)



1. 검색창에 들어갈 검색어 선정 ex) 건대입구역 맛집, 죽전역 맛집

2. 여러 식당 중에서 먼저 첫번째 가게를 클릭

3. 해당 식당의 고유 ID와 가게 이름을 수집

4. 식당 정보를 이용하여 새창을 띄운 후 리뷰데이터를 끝까지 수집

5. 모든 리뷰 데이터가 수집되면 csv로 정리되고 그 다음 식당에 대하여 리뷰 데이터 크롤링 수행

솔루션 제안 (데이터 크롤링)

```python
with open(file_name, mode='w', newline='', encoding='utf-8-sig') as f:
    writer = csv.writer(f)
    writer.writerow(["식당명", "작성일", "방문횟수", "리뷰 내용", "리뷰 태그", "방문자 유형"])

    for r in reviews:
        try:
            # 작성일
            date = r.find_element(By.CSS_SELECTOR, 'span.pui__gfuUIT > time').text.strip()

            # 방문횟수
            revisit_elements = r.find_elements(By.CSS_SELECTOR, 'span.pui__gfuUIT')
            revisit = ''
            for elem in revisit_elements:
                if "번째 방문" in elem.text:
                    revisit = int(elem.text.replace("번째 방문", "").strip())
                    break

            # 더보기 클릭 (리뷰 내용 펼치기)
            try:
                showmore = r.find_element(By.CSS_SELECTOR, 'a[data-pui-click-code="rvshowmore"]')
                driver.execute_script("arguments[0].click();", showmore)
                time.sleep(0.2)
            except:
                pass

            # 리뷰 내용
            content = r.find_element(By.CSS_SELECTOR, 'div.pui__vn15t2').text.strip()

            # 태그
            try:
                tag_area = r.find_element(By.CSS_SELECTOR, 'div.pui__HLNvmI')
                tags = [tag.text.strip() for tag in tag_area.find_elements(By.CSS_SELECTOR, 'span.pui__jhpEyP')]
                tag_str = ", ".join(tags)
            except:
                tag_str = ''

            # 🔻 방문자 유형
            visitor_text = ''
            try:
                visit_block = r.find_element(By.CSS_SELECTOR, '[data-pui-click-code="visitkeywords"]')
                em = visit_block.find_element(By.TAG_NAME, 'em')
                visitor_text = em.text.strip()
            except:
                visitor_text = ''
```

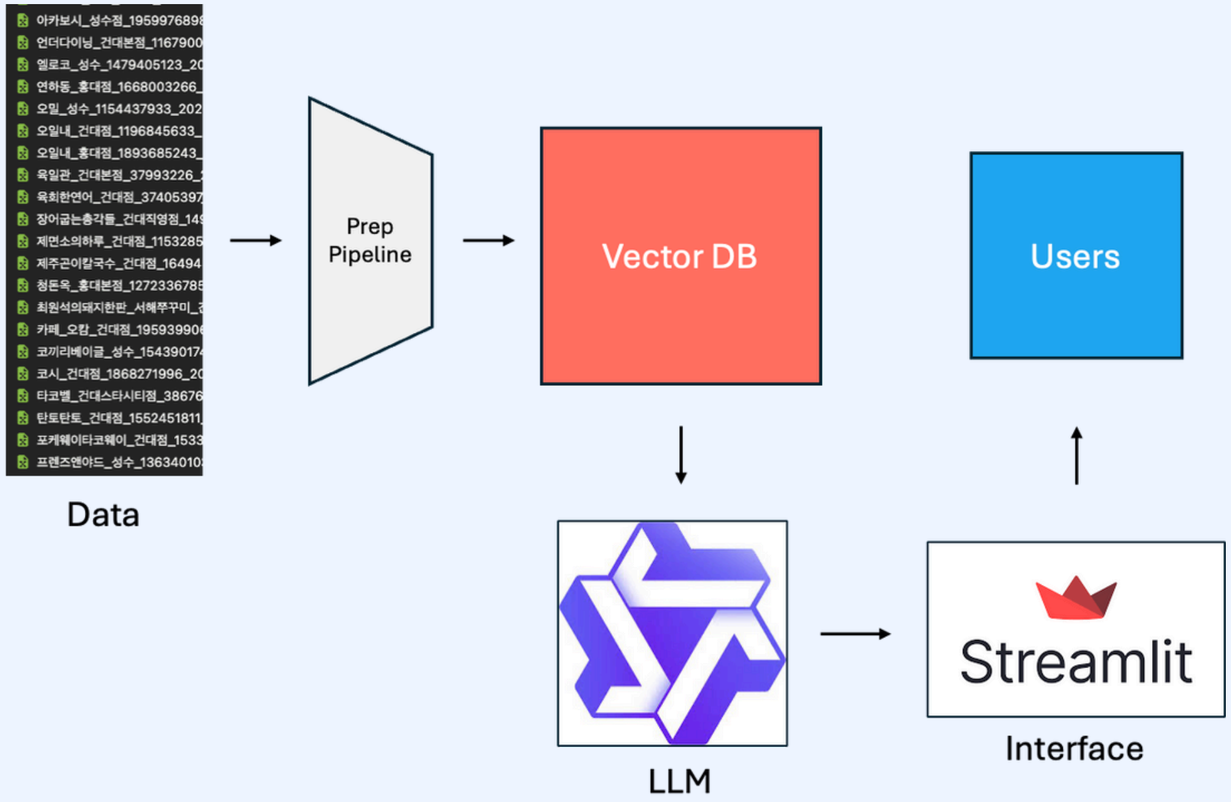
리뷰 데이터 크롤링 방식

	식당명	작성일	방문횟수	리뷰 내용	리뷰 태그	방문자 유형
	교도리 홍대점	5.31.토	1	여기 진짜 미쳤어 숯불향 제대로 배 접기	음식이 맛있어요,	저녁에 방문
	교도리 홍대점	3.4.화	1	스트레스를 왕창	화장실이 깨끗해	저녁에 방문
	교도리 홍대점	4.25.금	1	닭발 좋아하는 사 치킨이랑 야채튀 주먹밥이랑 같이 접기	음식이 맛있어요,	밤에 방문
	교도리 홍대점	6.3.화	1	닭발 매콤하니 맛 홍대에 프차밖에 이런 맛집 숨어있 다음에 또 올거에 팽이버섯 추가 필 접기	친절해요, 매장이	밤에 방문
	교도리 홍대점	5.27.화	1	홍대에 위치한 매 맵찔이 분도 맛있 연인과 함께 꼭 오	음식이 맛있어요,	저녁에 방문

매장별 저장된 리뷰 데이터 CSV

02. SOLUTION

솔루션 제안



1.데이터 전처리 파이프라인 수행

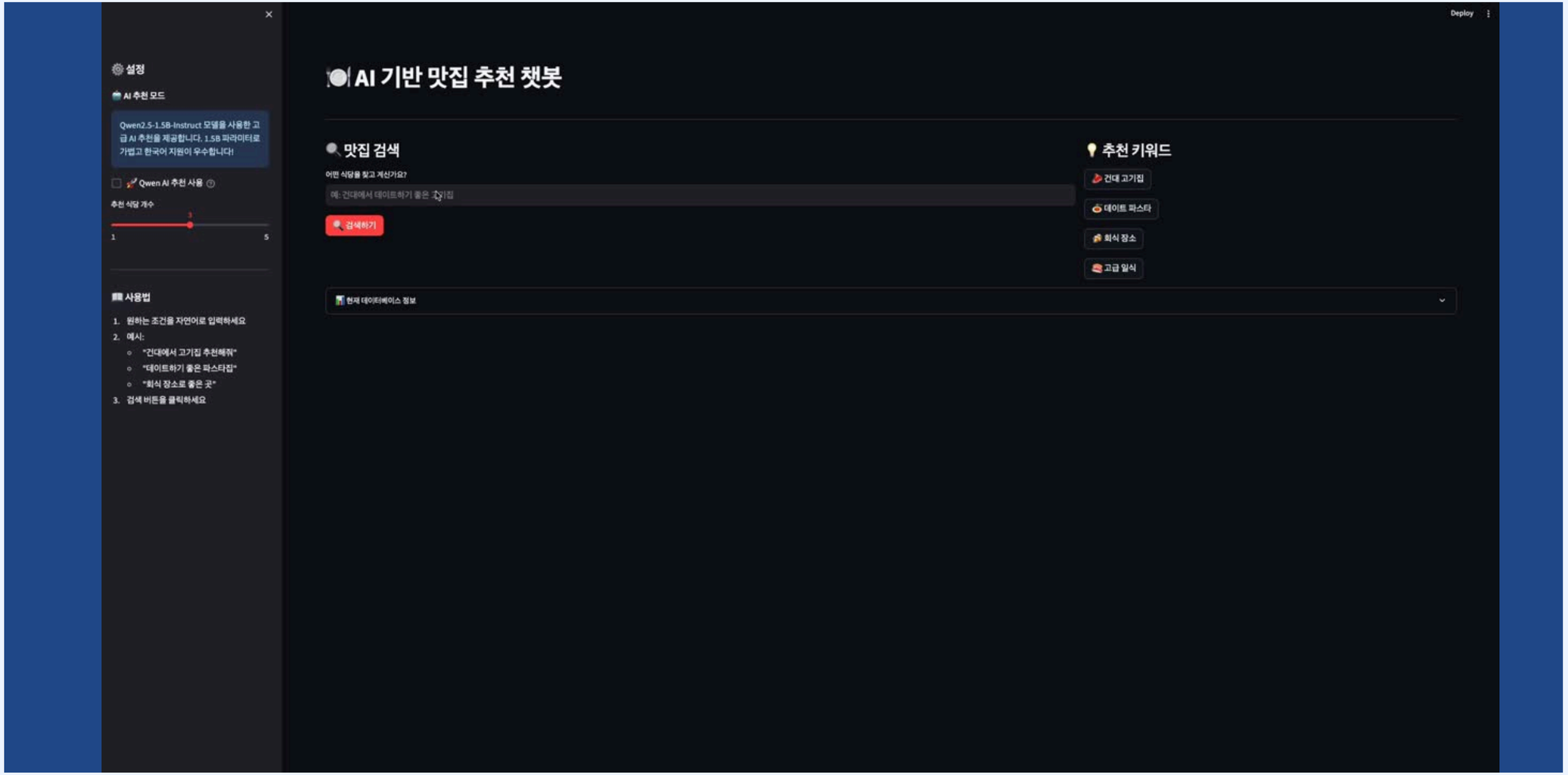
2.임베딩 후 벡터DB에 저장

3.사용자에 입력에 따라 알맞은 데이터를 벡터 DB에서 LLM으로 전달

4.프롬프트 엔지니어링이 완료된 LLM에서 답변 생성

5.사용자 인터페이스에 표시

시연 영상


Deploy
AI 기반 맛집 추천 챗봇
맛집 검색
어떤 식당을 찾고 계신가요?
예: 건대에서 데이트하기 좋은 고기집
검색하기
추천 키워드
건대 고기집
데이트 파스타
회식 장소
고급 일식
현재 데이터베이스 정보
설정
AI 추천 모드
Qwen2.5-1.5B-Instruct 모델을 사용한 고급 AI 추천을 제공합니다. 1.5B 파라미터로 가볍고 한국어 지원이 우수합니다!
Qwen AI 추천 사용
추천 식당 개수
1
5
사용법
1. 원하는 조건을 자연어로 입력하세요
2. 예시:
"건대에서 고기집 추천해줘"
"데이트하기 좋은 파스타집"
"회식 장소로 좋은 곳"
3. 검색 버튼을 클릭하세요

04. CHALLENGES

서비스 확장성 : 다양한 지역 확보 및 병렬성

성능 향상 – 하드웨어 측면 : 더 높은 성능의 LLM을 사용

성능 향상 – 소프트웨어 측면 : 프롬프트 개선을 통한 성능 향상

05. IMPROVMENTS

서비스 확장성

1. 지역 키워드 리스트 기반 순회 구조

- 기존 코드는 하나의 키워드만 수동으로 입력하여 데이터를 수집

```
# ✅ 메인 크롤링
search_url = "https://map.naver.com/p/search/건대입구역 맛집"
driver = webdriver.Chrome()
driver.get(search_url)
time.sleep(4)
```

- **주요 지역들을 리스트로 정리**한 후, 이를 자동으로 순회하여 리뷰데이터를 수집하는 방향으로 개선

```
keywords = ["건대입구역 맛집", "성수역 맛집", "망원역 맛집", "홍대입구 맛집"]
for keyword in keywords:
    수집함수(keyword)
```

2. 비동기 처리 기반 고속 수집 구조 도입

- 만약 리스트 기반 순회로 처리한다면, 순차 처리 방식이므로 시간이 오래 걸릴 수 있음
- **멀티쓰레딩을 이용한 비동기 기반 수집 구조**를 도입하여, 여러 지역의 리뷰를 동시에 빠르게 수집 가능

  기존 구조 → “1지역” → 끝날 때 까지 기다림 → “다음 지역“

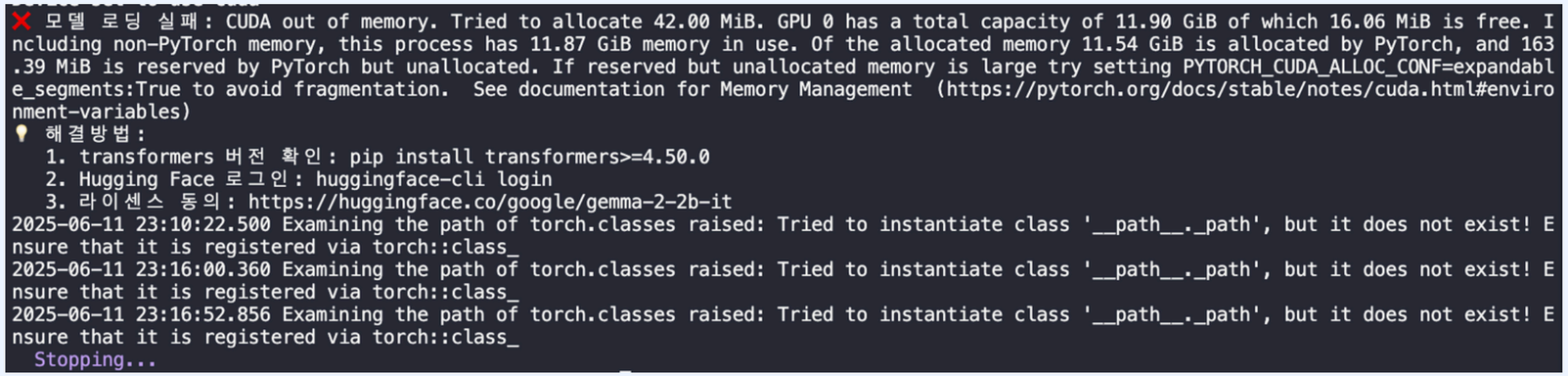
  개선 구조 → “여러 지역 병렬 요청“ → 결과 비동기 수집

3. 상황 기반 키워드 조합

- 실제 사용자들은 “홍대 맛집“이 아닌 “홍대 혼밥 맛집“ 과 같은 상황을 부여하므로 지역명 만으로는 한계가 존재
- **지역 리스트와 상황 리스트를 조합해 동적 키워드**를 생성하는 방식을 도입하여 개선

```
regions = ["홍대", "건대입구", "망원동"]
situations = ["혼밥", "데이트", "비 오는 날"]
keywords = [f"{region} {situation} 맛집" for region in regions for situation in situations]
```

성능 향상 - 하드웨어 측면



Gemma-2b 모델을 불러 왔을 때 생긴 out-of-memory 문제

1. LLM 모델 양자화 (4bit)

- 고성능 LLM을 저사양 환경에서도 구동할 수 있도록, 모델 파라미터를 4bit 정수형으로 압축하는 기술
- VRAM 사용량을 최대 80~90% 절감하면서도, 성능은 대부분 유지

2. LoRA 기반 경량 미세조정

- 기존 LLM을 그대로 유지하면서, 일부 핵심 레이어에만 소량의 파라미터 추가 학습
- 특히 양자화 모델과 함께 사용하면, 적은 자원으로 튜닝까지 가능
- 맛집 추천, 리뷰 요약 등 실제 서비스 목적에 맞게 모델 커스터마이징

05. IMPROVMENTS

성능 향상 - 소프트웨어 측면

1. 역할 지시 Prompoting

- 모델에게 명확한 역할을 부여해 일관된 응답 스타일을 유도하는 방식
- 장점 : 응답의 말투, 표현 방식, 형식이 안정적으로 유지됨

```
prompt = """
당신은 미식가들의 리뷰를 분석해 추천하는 맛집 큐레이터입니다.
사용자가 원하는 조건(지역, 분위기, 상황 등)에 가장 잘 맞는 식당을 3곳 추천해주세요.
"""
```

2. Few-shot Prompoting

- 원하는 출력 형태의 예시를 몇 개 보여주고 비슷한 형식으로 응답하게 하는 방식
- 장점 : 원하는 형식대로 대답하게 만들 수 있음

```
prompt = """
[예시 입력]
지역: 홍대 / 상황: 혼밥
→ 추천: 1. 연하동 홍대점 – 혼밥에 적합한 1인 좌석 보유. 최근 리뷰 다수 언급.
        2. 따롱이훠궈 홍대점 – 조용하고 혼자 먹기 편한 분위기.

[사용자 입력]
지역: 성수 / 상황: 데이트
→ 추천:
"""
```

3. Chain-of-Though Prompoting

- 문제 해결 과정을 단계별로 생각하도록 유도해 추론 정확도를 높이는 방식
- 장점 : 복잡한 조건이나 다단계 판단이 필요한 문제에서 정답률 향상

```
prompt = """
먼저 해당 지역에서 리뷰 수가 많은 식당을 찾고,
그중 혼밥 키워드가 언급된 식당만 걸러내고,
최종적으로 혼자 방문하기 좋은 곳 3곳을 추천해주세요.
"""
```

THANKS FOR WATCHING

https://github.com/Downy-newlearner/2025_DKU_BigDataProject