

데이터 전처리 인공지능 학습 결과서

산출물 단계	데이터 전처리
평가 산출물	인공지능 학습 결과서
제출 일자	2025-10-01
깃허브 경로	SKN14-Final-1Team
작성 팀원	정민영

[1] 사내 내부 문서 기반 RAG: Qwen 모델 (LoRA 파인튜닝)

1. 학습 개요

1.1 학습 목적

- 회사 내부 문서를 기반으로 한 소형 언어모델(Small Large Language Model) 파인튜닝
- 사내 문서 챗봇 구축을 위한 특화된 모델 개발
- 업무 관련 질문과 일상 질문을 모두 처리할 수 있는 통합 모델 구축

1.2 학습 대상 모델

Qwen3-8B: 메인 학습 모델

Qwen2.5-7B-Instruct: 비교 모델

1.3 학습 데이터

- 데이터셋: qwen3_company_train_dataset_combined.json
- 데이터 구조: 대화 형태의 메시지 데이터
- 팀별 분류: Backend, Frontend, Data_AI, CTO
- 말투 변형: 공손 말투체와 친구 말투체 각각 포함

2. 학습 설정

2.1 하이퍼파라미터

항목	값
학습 에포크	3 epochs
배치 크기	4 (per_device_train_batch_size)
그래디언트 누적	2 steps
학습률	1e-4
최적화기	adamw_torch_fused
데이터 타입	bfloat16
최대 시퀀스 길이	8192

2.2 LoRA 설정

항목	값
LoRA 알파	32
LoRA 드롭아웃	0.1
LoRA 랭크	8
타겟 모듈	["q_proj", "v_proj"]
바이어스	none

2.3 학습 최적화

- 그래디언트 체크포인팅: **True** (메모리 절약)
- 그래디언트 클리핑: **0.3**
- 워밍업 비율: **0.03**
- 학습률 스케줄러: **constant**

3. 학습 과정

3.1 데이터 전처리

- Chat 템플릿 적용: **Qwen** 표준 템플릿 사용
- 토큰화: 최대 **8192** 토큰으로 제한
- 레이블링: **Assistant** 응답 부분만 학습 대상으로 설정
- 패딩: 배치 내 최대 길이에 맞춰 패딩 적용

3.2 학습 모니터링

- 로깅 주기: 10 steps
- 모델 저장: 50 steps마다 체크포인트 저장
- 출력 디렉토리: qwen3-8b-informal-formal

4. 성능 평가 결과

4.1 TOOL CALL 멀티턴 성능 평가

평가 데이터: 학습 데이터와 동일한 방식으로 내용만 다른 테스트 데이터셋 생성

평가 기준: tool_selection, params_selection, params_value_similarity

모델	tool_selection	params_selection	params_value_similarity
Qwen3-8B (기본)	79.22%	79.22%	77.21%
Qwen2.5-7B (기본)	25.97%	25.97%	67.33%
Qwen3-8B (파인튜닝)	98.05%	98.05%	88.50%
Qwen2.5-7B (파인튜닝)	99.68%	99.68%	87.55%

1. 툴 선택 정확도 (tool_selection)

- 목표: 정답과 예측의 TOOL NAME이 일치하는지 평가.
- 계산: 정답에 tool_call 있는 샘플 중, 예측이 맞은 개수 / 총 샘플 수
- 해석: "정확한 도구를 선택했는지" 평가.

2. 파라미터 키 선택 정확도 (params_selection)

- **목표:** 정답과 예측의 파라미터 키(TOOL CALL의 파라미터 키)들이 얼마나 일치하는지 평가.
- **계산:** 정답 키와 예측 키의 매칭 수 / (정답 키 수 + 예측에만 있는 키 수)
- **해석:** "필요한 키(TOOL CALL의 키)를 정확히 선택했는지, 불필요한 키를 추가하지 않았는지" 평가.

3. 파라미터 값 유사도 (params_value_similarity)

- **목표:** 공통 키의 값이 얼마나 유사한지 평가.
- **계산:** 형태소 Jaccard(0.6) + 문자 유사도(0.4)의 평균 유사도 계산.
- **해석:** "선택된 값(TOOL CALL 키의 값)들이 얼마나 정확하게 일치하는지" 평가.

4.2 RAG 평가

평가 방식: 자체 제작 데이터셋(55개) + RAGAS 활용

평가 기준:

- **Context Recall:** 검색된 문맥이 질문과 기준 정답을 얼마나 잘 포괄하는지
- **Faithfulness:** 생성된 답변이 검색된 문맥에 얼마나 충실한지
- **Factual Correctness (mode = f1):** 생성된 답변이 기준 정답과 의미적으로 얼마나 일치하는지 (F1 스코어 기반)

모델	context_recall	faithfulness	factual_correctness(f1)
Qwen2.5-7B (파인튜닝)	0.7697	0.7992	0.3191
Qwen3-8B (파인튜닝)	0.9273	0.8614	0.3709

평가 방식: 자체 제작 데이터셋(55개) + LLM-AS-JUDGE 자체 정량 RAG 평가

평가 기준: 정확도, 재현율, 구체성

- **정확도 (Accuracy):** 시스템 답변이 기준 정답과 얼마나 일치하는지 평가. 점수 (0~3점)
 - 3점 (완벽 일치)
 - 2점 (약간 차이)
 - 1점 (핵심 맞지만 세부 누락)
 - 0점 (큰 차이)
 - 100점 환산: $(\text{정확도 점수} / 3) * 100$
- **재현율 (Recall):** 시스템 답변이 기준 정답에서 중요한 정보를 얼마나 잘 재현했는지 평가. 점수 (0~3점)
 - 3점 (모든 핵심 정보 정확히 재현)
 - 2점 (핵심 정보는 잘 재현되었으나 세부 부족)
 - 1점 (핵심 정보 누락)
 - 0점 (핵심 정보 많이 누락)
 - 100점 환산: $(\text{재현율 점수} / 3) * 100$
- **구체성 (Specificity):** 답변에서 세부 사항(예시, 매개변수 설명 등)을 얼마나 구체적으로 설명했는지 평가. 점수 (0~3점)
 - 3점 (구체적 설명 및 예시 포함)
 - 2점 (핵심 매개변수 설명, 예시 부족)
 - 1점 (기본 설명만 제공)
 - 0점 (세부 사항 누락)
 - 100점 환산: $(\text{구체성 점수} / 3) * 100$
- **최종 평균 점수** = $(\text{정확도} + \text{재현율} + \text{구체성}) / 3$

모델	정확도	재현율	구체성	평균
Qwen2.5-7B (파인튜닝)	67.27	70.91	55.76	64.65
Qwen3-8B (파인튜닝)	70.91	74.55	65.45	70.30

5. 기술적 특징

5.1 메모리 최적화

- bfloat16 사용으로 메모리 절약
- 그래디언트 체크포인팅으로 대용량 모델 학습 가능
- LoRA 기반 효율적 파인튜닝

5.2 대화 처리 능력

- Chat 템플릿 기반 구조화된 대화 처리
- 시스템 메시지와 사용자 메시지 구분
- Assistant 응답만 정확히 라벨링해 학습

5.3 도구 호출 기능

- TOOL CALL / TOOL RESPONSE 구조 학습
- 팀별 특화 검색 도구 활용
- 문서 검색 기반 정확한 답변 생성

6. 학습 성과 분석

6.1 베이스 모델 대비 멀티턴 TOOL CALL 성능 개선

Qwen2.5-7B: 25.97% → 99.68% (+73.71%p)

Qwen3-8B: 79.22% → 98.05% (+18.83%p)

파인튜닝 후: Qwen2.5-7B는 tool_selection이 1% 높고, Qwen3-8B는 params_selection가 1% 높음

6.2 RAG 성능 비교

Qwen3-8B 파인튜닝 모델이 모든 지표에서 우수

RAGAS 평가:

- context_recall: 0.7697(qwen2.5-7b) vs 0.9273(qwen3-8b)
- faithfulness: 0.7992(qwen2.5-7b) vs 0.8614(qwen3-8b)
- factual_correctness(f1): 0.3191(qwen2.5-7b) vs 0.3709(qwen3-8b)

자체 정량적 RAG 평가:

- Qwen2.5-7B [64.65] vs Qwen3-8B [70.30]

6.3 최종 모델 선정

TOOL CALL 성능은 두 모델 모두 충분

RAG 성능은 Qwen3-8B가 우수

최종 선정 모델: Qwen3-8B 파인튜닝 모델

7. 활용 방안

7.1 사내 문서 챗봇

- 팀별 특화 문서 검색 서비스
- 정확한 RAG 기반 업무 답변
- 일상 질문에도 자연스러운 대화 가능

7.2 추가 개발 방향

- 더 다양한 질문 유형 대응
- 실시간 문서 업데이트 RAG 반영
- 사용자 피드백 기반 개선

8. 결론

Qwen3-8B 파인튜닝을 통해 사내 문서 기반 전문 챗봇을 성공적으로 구축하였습니다..

주요 성과:

- TOOL CALL 성능 대폭 향상 (Qwen2.5-7B: +73.71%p, Qwen3-8B: +18.83%p)
- RAG 성능에서 Qwen3-8B가 가장 우수 (평균 70.30)
- 멀티턴 대화에서 안정적 TOOL CALL 처리

최종 선정 모델: Qwen3-8B 파인튜닝 모델

[2] LangGraph 기반 멀티모달 RAG 챗봇: GPT-4 + 하이브리드 검색

2.1 모델 개요 및 목적

모델명: GPT-4(4o, 4o-mini 혼합하여 OpenAI API 사용)

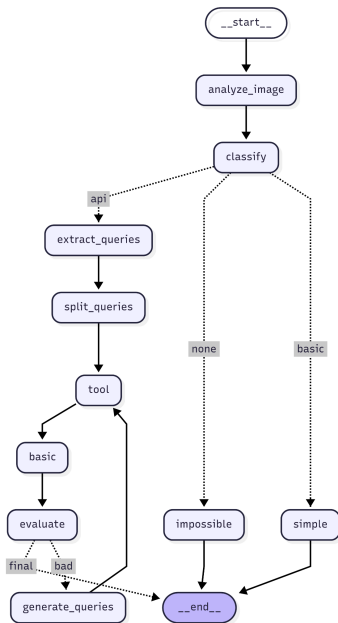
용도: API 문서(현재 구글 api 지원) 기반 멀티모달 질의응답 시스템

선정 이유:

- 멀티모달 처리에 최적: 스크린샷·다이어그램 등 이미지를 텍스트 문맥과 함께 이해
- 구조화된 출력 신뢰성: JSON 모드/함수호출을 활용해 쿼리 추출·분류에 안정적
- 재현성: temperature=0 설정으로 평가·운영 환경에서 일관된 응답
- 생태계·호환성: OpenAI SDK/도구 호환으로 LangGraph 노드와 자연스럽게 연계
- 성능/비용 밸런스: 일상 질문에는 GPT-4o-mini를 조합하여 지연시간과 비용 최적화
- 엔터프라이즈 활용성: 멀티모달+RAG 시나리오에서 높은 사실 충실도와 안전장치 연계 용이

2.2 아키텍처 및 구성

전체 워크플로우:



이미지 분석 → 질문 분류 → 쿼리 추출/분리 → LLM 툴 호출 → 하이브리드 검색 → 답변 생성 → 품질 평가 → (품질 평가에서 'BAD' 나오면) 검색 쿼리 재생성 & 재검색 후 다시 답변

핵심 구성 요소:

구성 요소	설명
멀티모달 입력 처리	GPT-4o API를 통한 이미지 분석 및 텍스트 통합
지능형 질문 분류	api(구글 API/기술), basic(일상), none(답변불가) 3단계 분류
LLM 툴 호출 시스템	11개 구글 API 태그 중 적절한 태그 자동 선택하여 검색
하이브리드 검색	Chroma + BGE-m3 임베딩(가중치 0.8) & BM25 키워드 검색(가중치 0.2)
재시도 검색 전략	초기 검색(text_k=5, qa_k=20) → 재검색(text_k=15, qa_k=30)

2.3 파라미터 및 설정

모델 및 검색 관련 주요 설정:

항목	설정
메인 모델	GPT-4o (temperature=0)
분류 모델	GPT-4o (temperature=0)
쿼리 추출	GPT-4o (JSON 모드)
품질 평가	GPT-4.1 (temperature=0)
임베딩 모델	BAAI/bge-m3 (normalize_embeddings=True)
벡터 DB	Chroma (원문/QA)
BM25 인덱스	태그별 사전 구축 및 로드(pkl파일)
지원 API	11개 구글 API (Maps, Gmail, Drive, Calendar 등)
메모리	MemorySaver (대화 상태 유지)
최대 재시도	1회 (품질 평가 기반)
히스토리 길이	최근 4개 메시지

2.4 주요 기능 및 성능

멀티모달 처리:

- 이미지 업로드 시 자동 분석 및 텍스트 통합
- 이미지-텍스트 연관 질문 처리 가능
- S3 기반 채팅 이미지 저장 및 URL 관리

지능형 검색:

- LLM이 자동으로 API 태그를 선택하여 검색 실행

품질 보장:

- 답변 품질 자동 평가 (good/bad)
- 부정적/회피적 답변 자동 감지
- 대체 쿼리 생성 및 재검색

활용 사례:

- 이 스크린샷에서 나오는 Drive API 오류 해결법은?(이미지 첨부)
- Gmail API에서 라벨별 메일 필터링하는 방법
- BigQuery 데이터셋 생성 시 권한 설정

2.5 활용 예시

입력: 사용자가 Drive API 오류 스크린샷과 함께 “이 오류 어떻게 해결해?”

처리 과정:

1. 이미지 분석: “google drive api에 대한 403 권한 오류가 나와있는 이미지, insufficient permissions 에러 메세지”
2. 질문 분류: api (구글 API 관련)
3. 쿼리 추출: Drive API 권한 오류 해결, 403 insufficient permissions
4. LLM 툴 호출: drive 태그 선택하여 vector_search_tool 호출

5. 하이브리드 검색: **drive** 태그로 필터링된 문서 검색 (**QA** / 원문 벡터 **DB** 검색)
6. 답변 생성: 구체적인 해결 방법 제시
7. 품질 평가: **good** (구체적&신뢰성 있는 정보 제공)

출력:

이 오류는 **Drive API** 접근 권한이 부족해서 발생합니다.

해결 방법:

- 1) Google Cloud Console에서 **Drive API** 활성화 확인
- 2) 서비스 계정에 적절한 **IAM** 역할 부여
- 3) OAuth 스코프에 '<https://www.googleapis.com/auth/drive>' 포함

2.6 향후 개선 계획

검색 고도화:

- 시맨틱 캐시 도입으로 응답 속도 개선
- 임베딩 모델 파인튜닝
- **Cross-encoder** 리랭킹 추가
- 다국어 쿼리 지원 확장

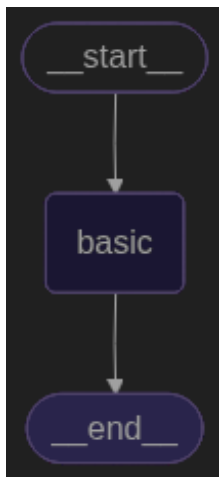
워크플로우 확장:

- 코드 생성 전용 노드 분기 추가
- **API** 테스트 결과 연동

2.7 LangGraph 노드 구조 발전 과정

초기 모델(기본 검색 모델)

- 구조: 단일 노드 구조
- 워크플로우: 질문 → 벡터 DB 검색 → 답변 생성
- 노드 구성: `basic_langgraph_node` 단일 노드

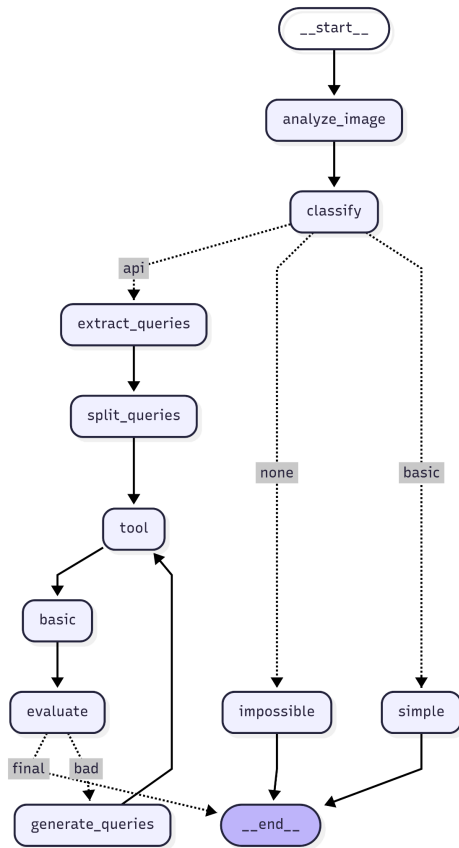


- 특징: 기본적인 RAG 구조, 쿼리 분리 없음, QA 벡터 DB 사용

최종 완성 모델(하이브리드 분기 처리 모델)

- 구조: 복합 조건부 분기 + 품질 평가 + 재시도 메커니즘
- 워크플로우: 이미지 분석 → 질문 분류 → 쿼리 추출 → LLM 톨 호출(메타 필터링 적용) → 하이브리드 검색 → 답변 생성 → 품질 평가 → (최종 답변/쿼리 재생성 & 재검색 후 다시 답변)

· 노드 구성



- 특징: LLM 톨 호출, 하이브리드 검색, 답변 품질 자동 평가, 재시도, 원문/QA 벡터 DB 모두 사용

RAGAS 평가 결과 (버전별 RAG 성능 비교):

평가 기준:

- **Context Recall:** 검색된 문맥이 질문과 기준 정답을 얼마나 잘 포괄하는지
- **Faithfulness:** 생성된 답변이 검색된 문맥에 얼마나 충실한지
- **Factual Correctness (mode = f1):** 생성된 답변이 기준 정답과 의미적으로 얼마나 일치하는지 (F1 스코어 기반)

버전	Context Recall	Faithfulness	Factual Correctness	특징
초기 버전	0.6520	0.7354	0.4940	기본 RAG 구조
최종 버전	0.9000	0.9274	0.6940	최종 완성 모델

2.8 최종 모델 성능 요약

지표	값
Context Recall	0.9000
Faithfulness	0.9274
Factual Correctness	0.6940

실제 운영 환경 적용:

- Django 웹 애플리케이션 통합 완료
 - apichat 앱의 utils 모듈로 구현
 - 실시간 대화형 인터페이스 제공
-

부록

- Qwen3-8B LoRA 학습 로그

```
1 # 학습 시작
2 trainer.train() # 모델이 자동으로 허브와 output_dir에 저장됨
3
4 # 모델 저장
5 trainer.save_model() # 최종 모델을 저장
[14]

The tokenizer has new PAD/BOS/EOS tokens that differ from the model config and generation config and generation config were aligned accordingly, being updated with the tokenizer: {'bos_token_id': None, 'pad_token_id': 151643}.

[888/888 2:35:06, Epoch 3/3]

Step Training Loss
10 0.997300
20 0.800500
30 0.701600
40 0.595900
50 0.582200
60 0.545600
70 0.528400
80 0.523300
```

- QWEN2.5-7B LoRA 학습 로그

```
1 # 학습 시작
2 trainer.train() # 모델이 자동으로 허브와 output_dir에 저장됨
3
4 # 모델 저장
5 trainer.save_model() # 최종 모델을 저장
[14]

The tokenizer has new PAD/BOS/EOS tokens that differ from the model config and generation config and generation config were aligned accordingly, being updated with the tokenizer: {'bos_token_id': None, 'pad_token_id': 151643}.

[888/888 1:28:31, Epoch 3/3]

Step Training Loss
10 0.806300
20 0.708600
30 0.702300
40 0.634900
50 0.616800
60 0.577900
70 0.568900
80 0.573700
90 0.548100
100 0.529700
110 0.505500
120 0.533600
```

- QWEN 모델 파인튜닝 학습 코드 스크립트:

https://github.com/skn-ai/14-250409/SKN14-Final-1Team-AI/tree/docs/SLLM_FINETUNING/250919

- Qwen 모델 학습 데이터(허깅페이스) :
<https://huggingface.co/datasets/SKN14-Final-1Team/qwen-finetuning-data-ko-250919>
- 학습된 QWEN2.5-7B-INSTRUCT 모델(허깅페이스) :
<https://huggingface.co/SKN14-Final-1Team/qwen2.5-7b-informal-formal-merged-09-19>
- 학습된 QWEN3-8B 모델(허깅페이스) :
<https://huggingface.co/SKN14-Final-1Team/qwen3-8b-informal-formal-merged-09-19>
- API 챗봇 LLM LANGGRAPH 소프트웨어(DJANGO 웹 어플리케이션 안에 위치) :
<https://github.com/skn-ai14-250409/SKN14-Final-1Team-Web/tree/develop/apichat/utlis>
- API 챗봇 LLM LANGGRAPH에서 사용하는 qa셋 데이터(허깅페이스) :
[SKN14-Final-1Team/google-api-qa-data-ko · Datasets at Hugging Face](#)