상세 파이프라인

날짜	@August 10, 2025 12:26 PM
단계	개발 단계
유형	참고문서
작성자	㉿ 종철 이

1.1. 프로젝트 초기화

- 1.1.1. Git 저장소 생성 및 Monorepo 폴더 구조(data, models, packages, scripts) 설정

```
/
├── data/              # 📦 데이터 자산 폴더 (Git LFS 또는 .gitignore 관리)
│   ├── raw/           # 원본 외부 데이터 (PDF, TXT 등)
│   ├── processed/     # 정제 및 청킹된 데이터 (chunks.jsonl)
│   ├── finetune/      # 합성된 Q&A 학습 데이터셋
│   └── knowledge_base/  # 생성된 FAISS 인덱스 파일
│
├── models/            # 🧠 모델 가중치 폴더 (Git LFS 또는 .gitignore 관리)
│   ├── student/       # 최종 파인튜닝 및 양자화된 학생 모델
│   └── teacher/       # (로컬 실행 시) 다운로드한 교사 모델
│
├── packages/          # 🐍 핵심 파이썬 소스 코드 패키지
│   ├── preprocessing/   # 데이터 전처리 및 지식 베이스 구축 관련 모듈
│   ├── training/      # Distill-M 2 모델 훈련 관련 모듈
│   └── inference/     # 최종 추론 파이프라인 관련 모듈 (분류, 검색, 생성 등)
│
├── scripts/           # 🛠️ 유틸리티 및 실행 스크립트
│   ├── run_preprocessing.py
│   ├── run_training.py
│   └── run_inference.py  # 최종 제출용 추론 스크립트 (inference.py)
│
```

```
├── tests/              # 🧪 테스트 코드
│   ├── unit/           # 단위 테스트
│   └── integration/    # 통합 테스트 (전체 파이프라인 검증)
│
├── requirements.txt    # 📜 Python 라이브러리 의존성 및 버전 관리
└── README.md           # 📄 프로젝트 설명 및 실행 방법 문서
```

- 1.1.2. requirements.txt 파일 생성 및 초기 라이브러리 버전 고정

```
# -- Core ML & Frameworks (대회 지정 및 필수 라이브러리) --
torch==2.1.0
transformers==4.41.2
accelerate==0.30.1
peft==0.11.1
trl==0.9.4
vllm==0.5.4

# -- RAG: Embedding & Vector DB --
sentence-transformers==2.7.0
faiss-cpu==1.8.0  # GPU 메모리 확보를 위해 CPU 버전 사용
rank-bm25==0.2.2

# -- Model Optimization --
auto-gptq==0.7.1
bitsandbytes==0.43.1

# -- Data Processing & Utilities --
pandas==2.2.2
langchain==0.2.1
PyMuPDF==1.24.1
beautifulsoup4==4.12.3
lxml==5.2.2
ftfy==6.2.0
chardet==5.2.0
konlpy==0.6.0
```

```
# -- Monitoring & Testing --
tqdm==4.66.4
wandb==0.17.0
pytest==8.2.0

# -- 기타 의존성 --
# (예: unstructured, clean-text 등 추가 전처리 라이브러리)
```

- 1.1.3. Linter (Ruff/Flake8) 및 Formatter (Black) 설정 파일 추가

1.2. 데이터 수집 및 전처리

- 1.2.1. Hugging Face Datasets API를 사용해 후보 데이터셋 다운로드 스크립트 작성
- 1.2.2. 데이터 라이선스 검증 및 출처 기록 자동화 스크립트 작성
- 1.2.3. PyMuPDF를 이용한 PDF 텍스트 및 표 추출 파서(Parser) 구현
- 1.2.4. BeautifulSoup4를 이용한 HTML 파서 구현
- 1.2.5. 국영문 혼합 텍스트 정제 클래스 (KoreanEnglishTextProcessor) 구현

1.3. RAG - 청킹 및 임베딩

- 1.3.1. RecursiveCharacterTextSplitter를 사용한 청킹 로직 구현
- 1.3.2. KoNLPy를 이용한 선택적 형태소 분석 함수 구현
- 1.3.3. sentence-transformers (BGE-M3) 모델 로드 및 텍스트 청크 임베딩 함수 구현

1.4. RAG - 지식 베이스 구축

- 1.4.1. 임베딩 벡터를 FAISS 인덱스에 추가하는 스크립트 작성
- 1.4.2. 생성된 FAISS 인덱스를 디스크에 저장/로드하는 기능 구현

1.5. 학습 데이터 준비

- 1.5.1. Teacher Model 로드 및 추론 환경 설정
- 1.5.2. Q&A 생성을 위한 프롬프트 템플릿 정의

- 1.5.3. Teacher Model을 사용하여 SyntheticQAPair 데이터셋을 생성하는 스크립트 작성
- 1.5.4. 생성된 Q&A 데이터셋 자동 품질 평가 및 필터링 스크립트 작성

단계 2: Epic 2 - 핵심 모델 파인튜닝 (Distill-M 2 준비)

2.1. 응답 생성 (Logits Generation)

- 2.1.1. vllm 라이브러리 설치 및 환경 설정
- 2.1.2. Teacher Model을 사용하여 SyntheticQAPair 질문에 대한 응답(logits) 생성/저장 스크립트 작성
- 2.1.3. Student Model을 사용하여 동일 질문에 대한 응답(logits) 생성/저장 스크립트 작성

단계 3: Epic 3 - 최종 훈련, 추론 및 최적화 🚀

3.1. 최종 훈련 (Distill-M 2)

- 3.1.1. 생성된 교사-학생 응답 쌍을 TRL 라이브러리 형식으로 재포맷하는 스크립트 작성
- 3.1.2. DistiLLMTrainer 및 distillm_v2 손실 함수를 사용한 최종 훈련 스크립트 작성
- 3.1.3. wandb 연동 및 훈련 로그 기록 기능 추가

3.2. 추론 파이프라인 구축

- 3.2.1. Question Classifier 컴포넌트 구현 (객관식/주관식 분류 로직)
- 3.2.2. Cache Layer 컴포넌트 구현 (diskcache 등 사용)
- 3.2.3. Multi-Stage Retriever 컴포넌트 구현 (BM25 + FAISS 검색 로직)
- 3.2.4. Inference Orchestrator 컴포넌트 구현 (전체 추론 흐름 제어)

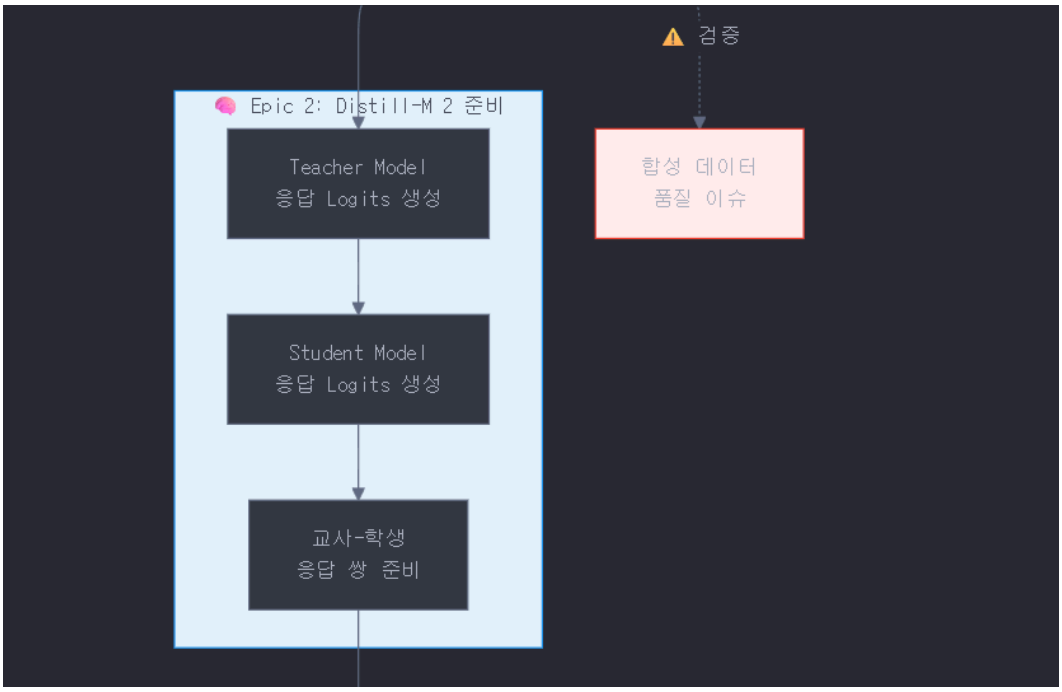
3.3. 예측 및 제출

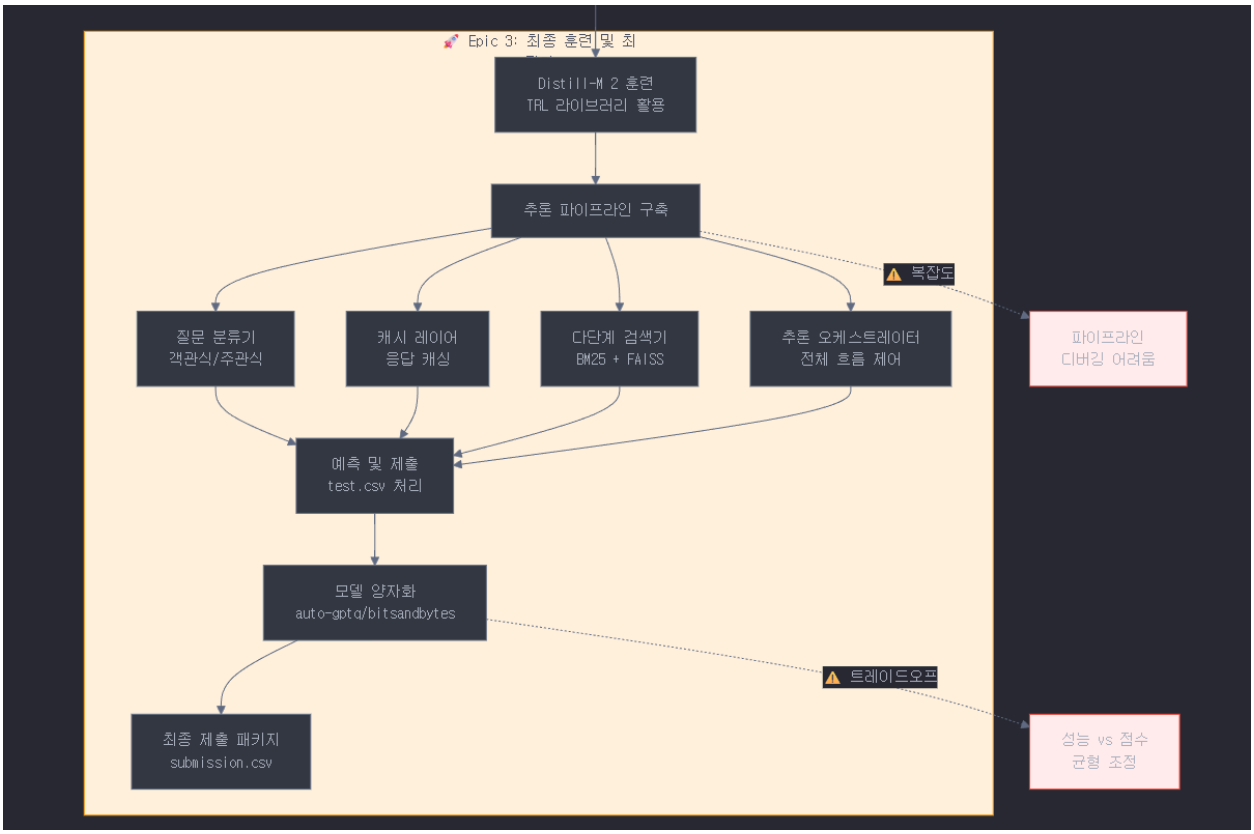
- 3.3.1. test.csv를 배치 단위로 처리하는 로직 구현

- 3.3.2. Fallback Handler 구현 (Timeout 또는 Low Confidence 시 대체 답변 생성 로직)
- 3.3.3. 최종 결과를 submission.csv 형식으로 저장하는 기능 구현

3.4. 최종화 (Finalization)

- 3.4.1. auto-gptq 또는 bitsandbytes를 사용한 모델 양자화 스크립트 작성
- 3.4.2. 양자화 전후 모델 성능 및 속도 비교/평가 스크립트 작성
- 3.4.3. 최종 제출용 패키지를 자동으로 생성하는 스크립트 작성
  이 상세 작업 목록이 팀의 실제 개발을 위한 구체적인 가이드가 될 것입니다.





RAG + Distill-M 2 파이프라인 스토리 📖

아주 오래전, 세상의 모든 금융과 IT 보안 지식이 담긴 **'위대한 도서관'** 이 있었습니다. 이 도서관의 지식을 완벽하게 이해하여 어려운 시험에 통과하는 것이 우리의 목표입니다.

1장: 도서관 정리와 카드 목록 제작 (Epic 1: 데이터 파이프라인) 📚

시험을 준비하기 전, 먼저 도서관부터 정리해야 합니다.

- **'성실한 사서 팀'** (`Data Preprocessing Component`) 이 도서관에 도착합니다. 도서관에는 오래된 양피지(PDF), 두루마리(HTML), 석판(TXT) 등 온갖 형태의 책들이 뒤죽박죽 섞여 있습니다.
- 사서들은 이 모든 책을 펼쳐 먼지를 털고(`텍스트 정제`), 내용을 깔끔한 **'지식 카드'** (`DocumentChunk`) 로 한 장씩 옮겨 적습니다.
- 마지막으로, 어떤 질문에도 즉시 관련 카드를 찾을 수 있도록, 모든 카드의 위치와 핵심 내용을 담은 마법의 **'카드 색인집'** (`FAISS Index`) 을 만듭니다.

이제 도서관은 완벽하게 정리되었습니다.

2장: 거장의 특별한 훈련법 (Epic 2 & 3: Distill-M 2 훈련) 🧠

도서관에는 모든 것을 아는 **'거장'** (`Teacher Model`) 이 있지만, 그는 너무 거대해서 작은 시험장에는 들어갈 수 없습니다. 그래서 재능 있는 **'영재 견습생'** (`Student Model`) 을 대신 시험에 내보내기로 하고, 특별한 훈련을 시작합니다.

1. **모의고사:** '사서'가 가져온 '지식 카드'를 보고 '견습생'이 먼저 문제의 답을 풀어봅니다. 중요한 것은, 답뿐만 아니라 그 답을 고른 **'생각의 과정'** (`Student Logits`) 까지 상세히 기록하는 것입니다.
2. **거장의 해설:** 같은 문제를 '거장'도 풀어봅니다. 그리고 자신의 완벽한 답과 함께, 왜 그렇게 생각했는지에 대한 **'생각의 과정'** (`Teacher Logits`) 을 남깁니다.
3. **1:1 특별 과외** (`DistiLLMTrainer`)**:** 이제 **'특별 교사'** 가 등장합니다. 그는 견습생의 답과 거장의 답을 비교하며 이렇게 가르칩니다.

> "견습생, 자네는 정답을 맞혔지만, 거장께서는 자네가 생각하지 못한 다른 가능성도 5% 정도 고려하셨네. 그 미묘한 차이가 진짜 실력일세."

단순히 정답을 알려주는 것이 아니라, 두 사람의 **'생각의 과정(로짓)'이 어떻게 다른지를 비교 분석**하여, 견습생이 무엇을 더 배워야 하는지를 알려줍니다. 이것이 바로 **'대조적 증류**

(Contrastive Distillation)' 입니다.

4. **깨달음과 성장:** '견습생'은 이 특별 과외를 통해, 단순히 지식을 암기하는 것을 넘어 **'거장처럼 생각하는 법'** 을 배우며 성장합니다.

3장: 결전의 날 (Epic 3: 최종 추론) 🏆

드디어 시험 당일, 완벽하게 훈련된 '견습생'과 그의 팀이 시험장에 들어섭니다.

1. **문제 분석 (`Question Classifier`):** 시험관이 문제를 나눠주자마자, '분석가'가 즉시 문제가 객관식인지 주관식인지 파악합니다.
2. **자료 수집 (`Multi-Stage Retriever`):** '자료 조사팀'이 마법의 '카드 색인집(FAISS)'과 키워드 검색(BM25)을 동시에 사용하여, 질문과 관련된 가장 정확한 '지식 카드'들을 순식간에 찾아냅니다. (이전에 찾아본 자료는 `캐시` 에서 더 빨리 꺼내옵니다.)
3. **답안 작성 (`Inference Orchestrator` & `Student Model`):** '감독관'이 질문과 찾아낸 지식 카드들을 '견습생'에게 전달합니다. '견습생'은 거장처럼 생각하는 훈련을 통해, 지식 카드를 근거로 완벽한 답안을 작성합니다.
4. **최종 제출:** 감독관은 답안지를 채점 규정에 맞게 `submission.csv` 라는 공식 답안지로 옮겨 적어 제출합니다.

이 이야기처럼, 우리 파이프라인은 체계적인 데이터 준비, 독창적인 훈련 방식, 그리고 효율적인 추론 팀플레이를 통해 최종 목표를 달성하도록 설계되었습니다.