

onlybooks / llm

Code Issues 9 Pull requests 1 Actions Projects Security Insights

Eye icon, Squash icon, Star icon, Arrow-down icon

LLM을 활용한 실전 AI 애플리케이션 개발

www.onlybook.co.kr/entry/llm

187 stars 152 forks 6 watching Branches Activity Tags

Public repository

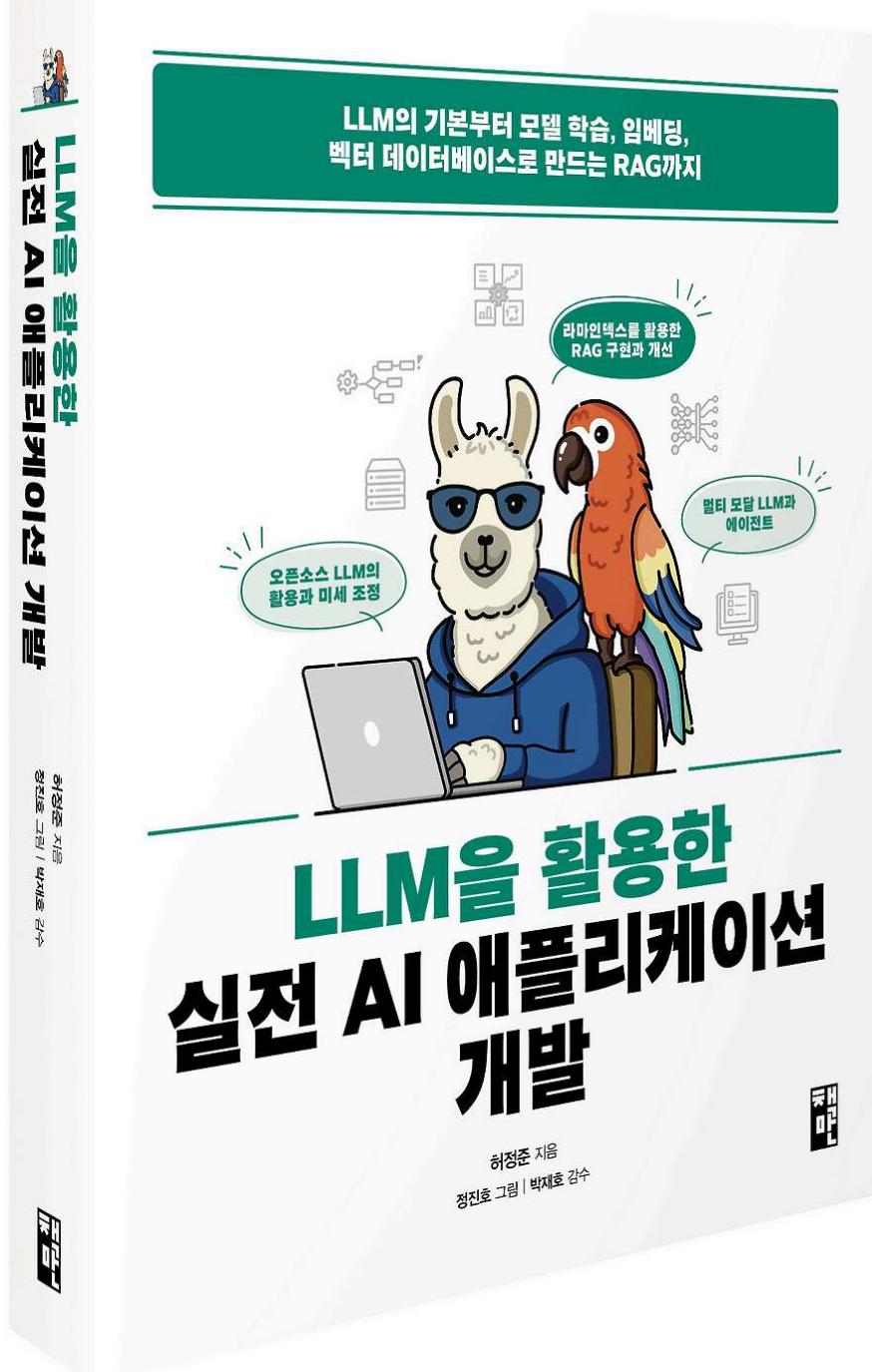
1 Branch 0 Tags Go to file Go to file Add file ...

onlybooks Update README.md b672610 · 7 months ago

번호	작업 내용	날짜
02장	decoder input error correction	9 months ago
03장	upgrade library version	9 months ago
05장	upgrade library version	9 months ago
06장	fix/variable-typo	last year
07장	Add files via upload	last year
08장	세션 재시작 안내 추가	last year
09장	필요한 라이브러리 추가	last year
10장	Add files via upload	last year
11장	Add files via upload	last year
12장	Add files via upload	last year
14장	Add files via upload	last year
15장	Add files via upload	last year
16장	Add files via upload	last year
README.md	Update README.md	7 months ago

README

Edit icon, More icon



LLM을 활용한 실전 AI 애플리케이션 개발

LLM의 기본부터 모델 학습, 임베딩, 벡터 데이터베이스로 만드는 RAG까지

허정준 지음 | 정진호 그림 | 박재호 감수

556쪽 | 32,000원 | 2024년 7월 25일 출간 | 18424027 | ISBN 9791189909703 (93000)

트랜스포머 아키텍처부터 RAG 개발, 모델 학습, 배포, 최적화, 운영까지

라마인덱스(LlamaIndex)와 LLM을 활용한 AI 애플리케이션 개발의 모든 것

★ 정오사항 확인: <https://www.onlybook.co.kr/entry/llm-errata>

이 책에서는 LLM의 기본 아키텍처에서 출발해 애플리케이션의 요구사항에 맞춰 LLM을 길들이고 제한된 컴퓨팅 환경에서 동작하게 경량화해서 원활하게 서빙하게끔 기초를 다진 다음에 RAG라는 LLM의 대표적인 애플리케이션을 만드는 방법을 차근차근 설명한다. 또한, 실제 운영과정에서 부딪히는 어려움을 해소하는 방법과 멀티 모달과 더불어 에이전트와 같은 고급 주제까지 다룬다. LLM 시대를 맞이해 필수적으로 갖춰야 하는 개발 지식을 이론과 실무 양쪽 관점에서 설명하므로, 새로운 패러다임에 적응하고자 하는 개발자들에게 가뭄의 단비와도 같은 책이다.

깃허브 실습 코드 다운로드

실습 코드는 책의 깃허브 저장소(<https://github.com/onlybooks/llm>)에서 확인할 수 있습니다. 깃허브의 코드는 구글 코랩에서 두 가지 방법으로 활용할 수 있다.

1. 로컬에서 업로드하기: 깃허브의 코드를 로컬 환경에 클론하거나 압축 파일 형태로 내려받은 후 진행하려는 실습 폴더의 노트북 파일(ipynb)을 구글 코랩에서 열어 실습을 진행할 수 있다.
2. 깃허브 URL로 열기: 구글 코랩에서 노트 열기(Ctrl+O)를 선택하면 다양한 노트 열기 방식 중 깃허브GitHub 탭에서 코드의 URL을 통해 실습 노트북을 열 수 있다.

| 이 책의 코드 실행 환경 |

이 책의 실습은 구글 코랩에서 실행한다. 구글 코랩은 구글에서 제공하는 노트북 실행 환경으로, 파이썬의 주피터 노트북과 유사한 UI로 브라우저에서 실행할 수 있다. 또 구글 코랩에서는 무료로 T4 GPU(16GB)를 사용할 수 있도록 제공한다. 구글 코랩의 무료 버전은 12시간의 런타임 제한이 있으며, 장시간 사용하지 않으면 연결이 끊길 수 있다.

지은이 허정준

서울대학교 기계항공공학부를 졸업하고 롯데면세점 빅데이터팀 데이터 분석가를 거쳐 현재는 프리랜서 마켓 크몽에서 AI 엔지니어로 일하고 있다. 『파이토치 라이트닝으로 시작하는 딥러닝』을 번역했으며, 최근에는 LLM을 활용한 어시스턴트(에이전트) 개발에 관심이 많다.

차례

[1부] LLM의 기초 뼈대 세우기

1장 LLM 지도

1.1 딥러닝과 언어 모델링

- _1.1.1 데이터의 특징을 스스로 추출하는 딥러닝
- _1.1.2 임베딩: 딥러닝 모델이 데이터를 표현하는 방식
- _1.1.3 언어 모델링: 딥러닝 모델의 언어 학습법

1.2 언어 모델이 챗GPT가 되기까지

- _1.2.1 RNN에서 트랜스포머 아키텍처로
- _1.2.2 GPT 시리즈로 보는 모델 크기와 성능의 관계
- _1.2.3 챗GPT의 등장

1.3 LLM 애플리케이션의 시대가 열리다

- _1.3.1 지식 사용법을 획기적으로 바꾼 LLM
- _1.3.2 sLLM: 더 작고 효율적인 모델 만들기
- _1.3.3 더 효율적인 학습과 추론을 위한 기술
- _1.3.4 LLM의 환각 현상을 대처하는 검색 증강 생성(RAG) 기술

1.4 LLM의 미래: 인식과 행동의 확장

1.5 정리

2장 LLM의 중추, 트랜스포머 아키텍처 살펴보기

2.1 트랜스포머 아키텍처란

2.2 텍스트를 임베딩으로 변환하기

- _2.2.1 토큰화
- _2.2.2 토큰 임베딩으로 변환하기
- _2.2.3 위치 인코딩

2.3 어텐션 이해하기

- _2.3.1 사람이 글을 읽는 방법과 어텐션
- _2.3.2 쿼리, 키, 값 이해하기
- _2.3.3 코드로 보는 어텐션
- _2.3.4 멀티 헤드 어텐션

2.4 정규화와 피드 포워드 층

- _2.4.1 층 정규화 이해하기
- _2.4.2 피드 포워드 층

2.5 인코더

2.6 디코더

2.7 BERT, GPT, T5 등 트랜스포머를 활용한 아키텍처

- _2.7.1 인코더를 활용한 BERT
- _2.7.2 디코더를 활용한 GPT
- _2.7.3 인코더와 디코더를 모두 사용하는 BART, T5

2.8 주요 사전 학습 메커니즘

- _2.8.1 인과적 언어 모델링
- _2.8.2 마스크 언어 모델링

2.9 정리

3장 트랜스포머 모델을 다루기 위한 허깅페이스 트랜스포머 라이브러리

- 3.1 허깅페이스 트랜스포머란
- 3.2 허깅페이스 허브 탐색하기
 - _3.2.1 모델 허브
 - _3.2.2 데이터셋 허브
 - _3.2.3 모델 데모를 공개하고 사용할 수 있는 스페이스
- 3.3 허깅페이스 라이브러리 사용법 익히기
 - _3.3.1 모델 활용하기
 - _3.3.2 토크나이저 활용하기
 - _3.3.3 데이터셋 활용하기
- 3.4 모델 학습시키기
 - _3.4.1 데이터 준비
 - _3.4.2 트레이너 API를 사용해 학습하기
 - _3.4.3 트레이너 API를 사용하지 않고 학습하기
 - _3.4.4 학습한 모델 업로드하기
- 3.5 모델 추론하기
 - _3.5.1 파이프라인을 활용한 추론
 - _3.5.2 직접 추론하기
- 3.6 정리

4장 말 잘 듣는 모델 만들기

- 4.1 코딩 테스트 통과하기: 사전 학습과 지도 미세 조정
 - _4.1.1 코딩 개념 익히기: LLM의 사전 학습
 - _4.1.2 연습문제 풀어보기: 지도 미세 조정
 - _4.1.3 좋은 지시 데이터셋이 갖춰야 할 조건
- 4.2 채점 모델로 코드 가독성 높이기
 - _4.2.1 선호 데이터셋을 사용한 채점 모델 만들기
 - _4.2.2 강화 학습: 높은 코드 가독성 점수를 향해
 - _4.2.3 PPO: 보상 해킹 피하기
 - _4.2.4 RLHF: 멋지지만 피할 수 있다면...
- 4.3 강화 학습이 꼭 필요할까?
 - _4.3.1 기각 샘플링: 단순히 가장 점수가 높은 데이터를 사용한다면?
 - _4.3.2 DPO: 선호 데이터셋을 직접 학습하기
 - _4.3.3 DPO를 사용해 학습한 모델들
- 4.4 정리

[2부 LLM 길들이기]

5장 GPU 효율적인 학습

- 5.1 GPU에 올라가는 데이터 살펴보기
 - _5.1.1 딥러닝 모델의 데이터 타입
 - _5.1.2 양자화로 모델 용량 줄이기
 - _5.1.3 GPU 메모리 분해하기
- 5.2 단일 GPU 효율적으로 활용하기
 - _5.2.1 그레이디언트 누적
 - _5.2.2 그레이디언트 체크포인팅
- 5.3 분산 학습과 ZeRO
 - _5.3.1 분산 학습
 - _5.3.2 데이터 병렬화에서 중복 저장 줄이기(ZeRO)
- 5.4 효율적인 학습 방법(PEFT): LoRA
 - _5.4.1 모델 파라미터의 일부만 재구성해 학습하는 LoRA
 - _5.4.2 LoRA 설정 살펴보기
 - _5.4.3 코드로 LoRA 학습 사용하기
- 5.5 효율적인 학습 방법(PEFT): QLoRA
 - _5.5.1 4비트 양자화와 2차 양자화
 - _5.5.2 페이지 옵티마이저
 - _5.5.3 코드로 QLoRA 모델 활용하기
- 5.6 정리

6장 sLLM 학습하기

- 6.1 Text2SQL 데이터셋
 - _6.1.1 대표적인 Text2SQL 데이터셋
 - _6.1.2 한국어 데이터셋
 - _6.1.3 합성 데이터 활용
- 6.2 성능 평가 파이프라인 준비하기
 - _6.2.1 Text2SQL 평가 방식
 - _6.2.2 평가 데이터셋 구축
 - _6.2.3 SQL 생성 프롬프트
 - _6.2.4 GPT-4 평가 프롬프트와 코드 준비
- 6.3 실습: 미세 조정 수행하기
 - _6.3.1 기초 모델 평가하기
 - _6.3.2 미세 조정 수행
 - _6.3.3 학습 데이터 정제와 미세 조정
 - _6.3.4 기초 모델 변경
 - _6.3.5 모델 성능 비교
- 6.4 정리

7장 모델 가볍게 만들기

- 7.1 언어 모델 추론 이해하기
 - _7.1.1 언어 모델이 언어를 생성하는 방법
 - _7.1.2 중복 연산을 줄이는 KV 캐시
 - _7.1.3 GPU 구조와 최적의 배치 크기
 - _7.1.4 KV 캐시 메모리 줄이기
- 7.2 양자화로 모델 용량 줄이기
 - _7.2.1 비츠앤바이츠
 - _7.2.2 GPTQ
 - _7.2.3 AWQ
- 7.3 지식 증류 활용하기
- 7.4 정리

8장 sLLM 서빙하기

- 8.1 효율적인 배치 전략
 - _8.1.1 일반 배치(정적 배치)
 - _8.1.2 동적 배치
 - _8.1.3 연속 배치
- 8.2 효율적인 트랜스포머 연산
 - _8.2.1 플래시어텐션
 - _8.2.2 플래시어텐션 2
 - _8.2.3 상대적 위치 인코딩
- 8.3 효율적인 추론 전략
 - _8.3.1 커널 퓨전
 - _8.3.2 페이지어텐션
 - _8.3.3 추측 디코딩
- 8.4 실습: LLM 서빙 프레임워크
 - _8.4.1 오프라인 서빙
 - _8.4.2 온라인 서빙
- 8.5 정리

[3부] LLM을 활용한 실전 애플리케이션 개발

9장 LLM 애플리케이션 개발하기

- 9.1 검색 증강 생성(RAG)
 - _9.1.1 데이터 저장
 - _9.1.2 프롬프트에 검색 결과 통합
 - _9.1.3 실습: 라마인덱스로 RAG 구현하기
- 9.2 LLM 캐시
 - _9.2.1 LLM 캐시 작동 원리
 - _9.2.2 실습: OpenAI API 캐시 구현
- 9.3 데이터 검증
 - _9.3.1 데이터 검증 방식
 - _9.3.2 데이터 검증 실습
- 9.4 데이터 로깅
 - _9.4.1 OpenAI API 로깅
 - _9.4.2 라마인덱스 로깅
- 9.5 정리

10장 임베딩 모델로 데이터 의미 압축하기

10.1 텍스트 임베딩 이해하기

_10.1.1 문장 임베딩 방식의 장점

_10.1.2 원핫 인코딩

_10.1.3 백오브워즈

_10.1.4 TF-IDF

_10.1.5 워드투벡

10.2 문장 임베딩 방식

_10.2.1 문장 사이의 관계를 계산하는 두 가지 방법

_10.2.2 바이 인코더 모델 구조

_10.2.3 Sentence-Transformers로 텍스트와 이미지 임베딩 생성해 보기

_10.2.4 오픈소스와 상업용 임베딩 모델 비교하기

10.3 실습: 의미 검색 구현하기

_10.3.1 의미 검색 구현하기

_10.3.2 라마인덱스에서 Sentence-Transformers 모델 사용하기

10.4 검색 방식을 조합해 성능 높이기

_10.4.1 키워드 검색 방식: BM25

_10.4.2 상호 순위 조합 이해하기

10.5 실습: 하이브리드 검색 구현하기

_10.5.1 BM25 구현하기

_10.5.2 상호 순위 조합 구현하기

_10.5.3 하이브리드 검색 구현하기

10.6 정리

11장 자신의 데이터에 맞춘 임베딩 모델 만들기: RAG 개선하기

11.1 검색 성능을 높이기 위한 두 가지 방법

11.2 언어 모델을 임베딩 모델로 만들기

_11.2.1 대조 학습

_11.2.2 실습: 학습 준비하기

_11.2.3 실습: 유사한 문장 데이터로 임베딩 모델 학습하기

11.3 임베딩 모델 미세 조정하기

_11.3.1 실습: 학습 준비

_11.3.2 MNR 손실을 활용해 미세 조정하기

11.4 검색 품질을 높이는 순위 재정렬

11.5 바이 인코더와 교차 인코더로 개선된 RAG 구현하기

_11.5.1 기본 임베딩 모델로 검색하기

_11.5.2 미세 조정한 임베딩 모델로 검색하기

_11.5.3 미세 조정한 임베딩 모델과 교차 인코더 조합하기

11.6 정리

12장 벡터 데이터베이스로 확장하기: RAG 구현하기

- 12.1 벡터 데이터베이스란
 - _12.1.1 딥러닝과 벡터 데이터베이스
 - _12.1.2 벡터 데이터베이스 지형 파악하기
- 12.2 벡터 데이터베이스 작동 원리
 - _12.2.1 KNN 검색과 그 한계
 - _12.2.2 ANN 검색이란
 - _12.2.3 탐색 가능한 작은 세계(NSW)
 - _12.2.4 계층 구조
- 12.3 실습: HNSW 인덱스의 핵심 파라미터 이해하기
 - _12.3.1 파라미터 m 이해하기
 - _12.3.2 파라미터 ef_construction 이해하기
 - _12.3.3 파라미터 ef_search 이해하기
- 12.4 실습: 파인콘으로 벡터 검색 구현하기
 - _12.4.1 파인콘 클라이언트 사용법
 - _12.4.2 라마인덱스에서 벡터 데이터베이스 변경하기
- 12.5 실습: 파인콘을 활용해 멀티 모달 검색 구현하기
 - _12.5.1 데이터셋
 - _12.5.2 실습 흐름
 - _12.5.3 GPT-4o로 이미지 설명 생성하기
 - _12.5.4 프롬프트 저장
 - _12.5.5 이미지 임베딩 검색
 - _12.5.6 DALL-E 3로 이미지 생성
- 12.6 정리

13장 LLM 운영하기

- 13.1 MLOps
 - _13.1.1 데이터 관리
 - _13.1.2 실험 관리
 - ~~13.1.3 모델 저작소~~

Releases

No releases published

Packages

No packages published

Contributors 3

-  **onlybooks** 책만
-  **shangrilar** Jungjun Hur
-  **yunjihyung**

Languages

