Ch 04. Model

상태	완료
최종 편집 일시	@2025년 11월 16일 오전 11:59

https://wikidocs.net/233772

왜 모델인가

> 모델 혹은 LLM(Large Language Model) 단계는 **이전 프롬프트 단계에서 구성된 입력을 기반으로 대규모 언어 모델을 활용하여 응답을 생성하는 과정**입니다. 이 단계는 RAG 시스템의 핵심적인 부분으로, **언어 모델의 능력을 최대한 활용하여 사용자의 질문에 대해 정확하고 자연스러운 답변을 생성**합니다.

필요성

> 왜 LLM 단계가 필요할까?
>
> 1. **사용자 의도 이해**
>
>   : **자연어 이해(NLU)와 자연어 생성(NLG) 능력이 결합**되어, 보다 자연스럽고 유익한 응답을 제공할
>
> 2. **문맥적 적응성**: 주어진 **문맥을 고려하여 응답을 생성**

01. 다양한 LLM 모델 활용

OpenAI의 모델 파라미터

1. `temperature` : 샘플링 온도 설정 (0~2. float) → 일종의 창의성

   높은 값은 출력을 더 **무작위**하게 만들고, 낮은 값은 출력을 더 **집중되고 결정론적**으로 만든다.

2. `max_tokens` : 채팅 완성에서 생성할 토큰의 최대 개수 → 결국 비용 관리에 초점을 둔다
   - 모델이 한 번에 출력할 수 있는 **"말"의 양을 제한**한다.
   - 너무 길거나 무한하게 이어지는 응답을 방지해서 시스템 자원 낭비와 응답 지연을 막는다.
   - 대화 흐름을 관리하거나, API 사용량 (비용)을 효율적으로 통제할 수 있습니다

   

   토큰이란 LLM에서 어떤 의미를 가지는가?
   LLM 기반 챗봇이 한번의 응답에서 생성할 수 있는 최대 텍스트 길이를 설정하는 것을 `max_token`을 통해 설정할 수 있다.

3. `model_name` : 적용할 모델

모델명	설명	컨텍스트 길이	최대 출력 토큰	학습 데이터
gpt-4o	GPT-4 터보보다 저렴하고 빠른 최신 다중모드 플래그십 모델	128,000 토큰	16,384 토큰	2023년 10월까지
gpt-4-turbo	최신 GPT-4 터보 모델. 비전 기능, JSON 모드, 기능 호출 지원	128,000 토큰	4,096 토큰	2023년 12월까지
gpt-4o-mini	GPT-3.5 터보보다 더 우수한 성능의 작은 모델	128,000 토큰	16,384 토큰	2023년 10월까지
o1-preview	다양한 도메인의 어려운 문제 해결을 위한 추론 모델	128,000 토큰	32,768 토큰	2023년 10월까지
o1-mini	코딩, 수학, 과학에 특화된 빠른 추론 모델	128,000 토큰	65,536 토큰	2023년 10월까지
gpt-4o-realtime	실시간 API용 오디오 및 텍스트 입	128,000 토큰	4,096 토큰	2023년 10월까지

모델명	설명	컨텍스트 길이	최대 출력 토큰	학습 데이터
	력 처리 모델 (베타)			

주요 LLM 모델 및 서비스 요약

- **OpenAI (GPT-4o, GPT-4-turbo 등)**
  - 다양한 모델명, 토큰 컨텍스트 길이, 출력 토큰 수, 학습 데이터 최신 시기, 가격 및 성능에 따라 활용 가능
  - 대표 옵션: temperature(무작위성), max_tokens(출력 길이), model_name(모델 선택)
  - 주요 활용 코드 예제 제공
- **Anthropic (Claude 3 시리즈)**
  - 안전성과 윤리에 중점을 둔 AI 스타트업
  - 다양한 Claude 모델(3.5, 3 Sonnet, Haiku 등)이 API와 클라우드 플랫폼(AWS, GCP 등)에 배포됨
  - 각 모델명별 활용 예시 및 API 정보 제시
- **Perplexity**
  - 검색·생성융합 AI 서비스, 다양한 LLM 채택, 파일/PDF/이미지 분석 지원
  - API 및 Pro 서비스 안내, 모델 옵션·매개변수 상세 설명, 예시 코드 제공
- **Cohere**
  - 기업용 AI (Command R+, Aya 등), 긴 컨텍스트 윈도우 및 다국어 지원·비영리 오픈모델 개발
  - 모델별 특징, 실제 사용 코드를 안내
- **Upstage**
  - 국내 기업, Solar LLM 및 문서처리 솔루션 소개
  - OCR, 검색 추천, 챗봇 등 다양한 기능
- **Xionic**
  - 한국어에 특화된 AI 모델, 상업적 라이선스
  - 대표 서비스와 실제 활용 코드 안내
- **LogicKor**

- 한국어 LLM 성능 벤치마크 리더보드 운영
- 국내외 모델의 사고력·성능 비교, 오픈소스 평가 시스템

---

02. 캐싱(Cache)

> Langchain은 LLM을 위한 선택적 캐싱 레이어를 제공한다.
>
> 1. 동일한 완료를 요청하는 경우 발생하는 API 호출 횟수로 인한 비용 절감
> 2. API 호출 횟수 절감을 통한 애플리케이션의 속도 향상

InMemoryCache란 무엇인가

> **인메모리 캐시(In-memory cache)**란 데이터나 결과를 **프로그램 실행 중 메모리(RAM)에 저장해두고** 같은 요청이 반복될 때 디스크, 데이터베이스, API 등 원래 데이터 소스에 다시 접근하지 않고 미리 저장해둔 값을 바로 반환하도록 하는 **임시 저장소**

- 빠르고 비용 절감 효과가 있지만 캐시는 휘발성이기 때문에 프로세스 재시작할 시 데이터가 사라지고, 큰 용량의 데이터 저장, 장기저장에 부적합하다
- 빠르지만 안정적이지 않다

SQLiteCache란 무엇인가

- SQLite란?

  > **가볍고 빠른** 데이터를 저장 및 관리할 수 있는 **파일 기반의 관계형 데이터베이스** (RDBMS)

  - Windows, macOS, Linux 등 거의 모든 **운영체제**에서 동일하게 사용 가능
  - Python, Java, C 등의 다양한 언어에서 **동일한 방식**으로 DB를 생성 및 접근 가능
  - 모바일 (Android, iOS), 임베디드 기기 등 다양한 플랫폼에서도 잘 작동

  → 호환성이 좋고 각 OS 별 경로만 잘 따른다면 문제없이 활용할 수 있다.

- SQLiteCache란 이런 SQLite에 정보를 저장해서 빠르게 불러온다.

특징	InMemoryCache (인메모리 캐시)	SQLite Cache (디스크 기반 캐시)
저장 위치	프로그램 실행 중 "메모리(RAM)"에 저장	로컬 디스크의 "SQLite DB(파일)"에 저장

특징	InMemoryCache (인메모리 캐시)	SQLite Cache (디스크 기반 캐시)
지속성	프로그램 종료/재시작 시 캐시 내용 사라짐 (휘발성)	디스크에 영구 저장, 프로그램 재시작 후에도 유지
속도	매우 빠름 (RAM 접근)	메모리보다는 느림 (디스크 파일 접근)
용량 제한	메모리 한계에 따라 캐시 크기가 제한적임	디스크 공간 사용 가능, 대용량 캐시 가능
사용 예시	빠른 실험, 일회성 처리 또는 재시작될 서비스	여러 번 재실행하거나 서버/클러스터 환경, 장기 저장 필요할 때

- 이 캐싱 전략에서 프롬프트의 '동일성'은 어떻게 판단하는가?

  LangChain에서 **캐싱의 '동일함' 기준**

  - **입력 프롬프트(문자열)가 완전히 '동일'할 때 (**프롬프트 템플릿, 입력 값 등)
  - 프롬프트가 의미상 같더라도 사소한 문자 차이가 있으면 캐시에서 조회되지 않는다.

    (예: '한국'과 '한 국'은 다르다)

  캐시 키 생성 방식은 보통 다음과 같은 요소의 조합:

  - 프롬프트의 직렬화된 **문자열 값**
  - 사용한 **LLM(예: 모델명, 파라미터 등)** 정보

  시맨틱 캐시(semantic cache) 옵션을 사용하면,

  - 프롬프트를 임베딩 (문장 벡터)으로 변환한 뒤, **문장 유사도(semantic similarity)**를 기준으로 유사한 질문에 대해 기존 캐시를 활용한다 → **문장 의미 기반**
  - https://www.mongodb.com/ko-kr/docs/atlas/ai-integrations/langchain/memory-semantic-cache/

03. 모델 직렬화(Serialization) - 저장 및 불러오기

직렬화(Serialization)

> 모델을 재사용 가능하게 저장 가능하게 하자

목적

- 모델 재사용 (재훈련 없이)

- 모델 배포 및 공유 용이
- 계산 리소스 절약

장점

- 빠른 모델 로딩
- 버전 관리 가능
- 다양한 환경에서 사용 가능

> 모델 클래스, llm 객체, 체인은 모두 직렬화가 가능하다. (`is_lc_serializable` 클래스 메서드로 확인 가능)

1. 체인을 `dumps` , `dumpd` 를 활용해서 Dict 또는 JSON 문자열로 직렬화할 수 있다.
2. Python 객체를 바이너리 형태로 직렬화하는 포맷인 Pickle을 활용해서 직렬화 할 수 있다.

   특징

   - Python 전용 (다른 언어와 호환 불가)
   - 대부분의 Python 데이터 타입 지원 (리스트, 딕셔너리, 클래스 등)
   - 객체의 상태와 구조를 그대로 보존

   장점

   - 효율적인 저장 및 전송
   - 복잡한 객체 구조 유지
   - 빠른 직렬화/역직렬화 속도

   단점

   - 보안 위험 (신뢰할 수 없는 데이터 역직렬화 시 주의 필요)
   - 사람이 읽을 수 없는 바이너리 형식

주요 용도

1. 객체 캐싱
2. 머신러닝 모델 저장
3. 프로그램 상태 저장 및 복원

사용법

- `pickle.dump()` : 객체를 파일에 저장
- `pickle.load()` : 파일에서 객체 로드

04. 토큰 사용량 확인

> `with get_openai_callback()` 구문안에서 실행되는 모든 토큰 사용량/요금이 추적된다.

```
# callback을 사용하여 추적합니다.
with get_openai_callback() as cb:
    result = llm.invoke("대한민국의 수도는 어디야?")
    result = llm.invoke("대한민국의 수도는 어디야?")
    print(f"총 사용된 토큰수: \t\t{cb.total_tokens}")
    print(f"프롬프트에 사용된 토큰수: \t{cb.prompt_tokens}")
    print(f"답변에 사용된 토큰수: \t{cb.completion_tokens}")
    print(f"호출에 청구된 금액(USD): \t${cb.total_cost}")
✓ [37] 19ms

총 사용된 토큰수:          46
프롬프트에 사용된 토큰수:   30
답변에 사용된 토큰수:  16
호출에 청구된 금액(USD):   $0.00023500000000000002
```

05. 구글 생성 AI(Google Generative AI)

→ 랭체인 버전 오류를 해결하지 못해 스킵