프롬프트의 중요성

1. 문맥 설정
- 특정 문맥에서 작동하도록 설정
2. 정보 통합
- 서로 다른 문서의 내용을 통합하고 모델이 활용할 수 있는 형식으로 조정
3. 응답 품질 향상
- 프롬프트 구성에 따라 응답 품질 달라짐

프롬프트 구조

- 지시사항
- 사용자 입력 질문
- 질문 결과 출력

1. From_template() 사용해서 promptTemplate 객체 생성하기
- 치환될 변수를 { }로 묶어서 정의
2. PromptTemplate 객체 생성과 동시에 prompt 생성하기
- Prompt 안에 template 존재
- 유효성 검사를 위해 input_variables 지정

Partial_variables : 부분 변수 채움

Partial은 함수를 부분적으로 사용하기 위함.

항상 공통된 방식으로 가져오고 싶은 변수가 있는 경우 – 날짜, 시간 등

Prompt.format(n=?): 프롬프트 생성

파일로부터 템플릿 읽어오기

ChatPromptTemplate

: 대화 목록을 프롬프트로 주입하고자 할 때

메시지는 튜플 형식, (role, message)로 구성

Role: system- 시스템 설정 메시지 human- 사용자 입력 메시지 ai – ai 답변 메시지

MessagePlaceholder

- 포맷하는 동안 렌더링할 메시지를 완전히 제어
  *렌더링: 저장되어 있는 **데이터나 포맷을 사람이 볼 수 있는 형태로 변환하는 과정**. 즉, **출력 단계** 또는 **시각화 과정**

FewshotPromptTemplate

: LLM에게 단순히 "질문만" 주는 것이 아니라, 예시 몇 개를 함께 제공해 모델이 답변 패턴을 학습하도록 유도하는 프롬프트 방식

Example sector

예제가 많은 경우 프롬프트에 포함할 예제 선택시 작업 담당 클래스

Langchain Hub에서 프롬프트 받아서 실행

- Repo의 아이디, 혹은 commit id로 특정 버전에 대한 프롬프트 가져오기

Hub.pull("repo id or commit id")

Prompt hub에 자신의 프롬프트 등록도 가능

Hub.push("location", prompt)

이거를 langchain 아이디 입력후 불러오기도 가능

Stuff document

- Rag 파이프라인에서 검색된 문서를 그대로 한번에 붙여넣기

Map

- 검색된 여러 문서 각각에 대해 개별적으로 요약 또는 답변 생성

- 문서 단위로 1차 처리하는 프롬프트

Reduce

- Map단계에서 생성된 요약, 답변들을 하나로 통합
  = 요약의 요약

Metadata tagger

- 문서에 추가적인 속성을 자동으로 태깅하는 프롬프트
- 검색 성능 향상, 쿼리시 필터링 조건 사용, 요약 contecxt 제공 등

Chain of density

- LLM이 요약한 것을 밀도있게 만들어가는 과정적 프롬프트 기법