(8주차) Prompt Engineering

명지대학교
MYONGJI UNIVERSITY

프롬프트 엔지니어링 정의 및 중요성

상세 설명

- LLM: 입력 텍스트 프롬프트를 기반으로 다음 단어 예측 방식으로 작동함
- 프롬프트 엔지니어링: LLM의 정확하고 유용한 결과물 생성을 유도하는 고품질 프롬프트 설계 과정
- 단순 질문 이상으로, 모델, 학습 데이터, 설정, 단어 선택, 문체, 구조 등 고려한 최적 프롬프트 반복 탐색 과정
- 데이터 과학자/ML 엔지니어 아니어도 프롬프트 작성 가능하나, 효과적 작성은 복잡
- 잘못된 프롬프트는 모호/부정확 응답 유도 가능성

프롬프트 엔지니어링 정의 및 중요성

예시

- 단순 질문: "파리 날씨 어때?"
- 프롬프트 엔지니어링 적용: "여행 계획 중. 다음 주 프랑스 파리 예상 최고/최저 기온 및 강수 확률 정보 요청." (더 구체적, 명확 정보 요구)

실습

- 주제: 간단한 정보 검색
- 미션: 동일 정보(예: "대한민국 수도?") 획득 위한 최소 3가지 다른 스타일 프롬프트 작성 및 LLM 응답 차이 비교 수행

LLM 출력 구성 (Output Configuration)

상세 설명

- 프롬프트 외 LLM 출력 제어 설정값(파라미터) 이해 및 조절 중요함
- 출력 길이 (Output Length):
  * 생성될 최대 토큰 수 제한. 응답 길이, 계산 비용, 응답 시간 등 영향 줌
  * 길이 축소는 요약 능력 향상 의미 아니며, 단순히 예측 중단임
  * ReAct 등 특정 기법에서 불필요 토큰 생성 방지에 중요함
- 샘플링 제어 (Sampling Controls):
  * LLM은 다음 토큰 확정 아닌, 가능 토큰 확률 분포 예측함. 샘플링 제어는 이 확률 기반 실제 출력 토큰 선택 방식 조절함
  * 주요 설정: Temperature, Top-K, Top-P

LLM 출력 구성 (Output Configuration)

상세 설명

- Temperature:

  * 토큰 선택 무작위성 제어

  낮을수록(0) 확률 높은 토큰만 선택, 결정론적/일관된 응답 유도 (사실 기반 작업 적합),

  높을수록 다양한 토큰 선택 확률 증가, 창의적/예측 불가능 응답(브레인스토밍, 창작 적합)

  * 과도하게 높으면 관련성 없는 내용 또는 반복 루프(Repetition Loop) 문제 발생 가능성

- Top-K:

  * 예측 확률 높은 상위 K개 토큰 중 다음 토큰 선택

  K 작을수록(1) 결정론적/사실 기반 응답, 클수록 창의적 응답 가능성 높음

LLM 출력 구성 (Output Configuration)

상세 설명

- Top-P (Nucleus Sampling):
  * 예측 확률 누적값 P 초과하지 않는 상위 토큰 중 다음 토큰 선택
    P가 0에 가까우면 확률 높은 토큰만 선택, 1에 가까우면 거의 모든 토큰 고려
- 상호작용 및 조합:
  * Temperature, Top-K, Top-P는 함께 작동하며 상호 영향
  * 한 설정값 극단적 설정 시 다른 설정 무의미해질 수 있음
    (예: Temperature=0이면 Top-K, Top-P 무관)
  * 일반적으로는 Temp 0.2, Top-P 0.95, Top-K 30 (일관성 + 약간의 창의성)
    창의적으로는 Temp 0.9, Top-P 0.99, Top-K 40
    덜 창의적으로는 Temp 0.1, Top-P 0.9, Top-K 20
    단일 정답: Temp 0

LLM 출력 구성 (Output Configuration)

예시

- "옛날 옛날에..." 프롬프트, Temperature 0.1과 0.9 설정 시 각각 나올 이야기 예측 (낮으면 전형적, 높으면 예상 밖 전개)

실습

- 주제: 창의적 글쓰기 (시 또는 짧은 이야기)
- 미션: 동일 시작 프롬프트(예: "비 오는 날 창밖을 보니") 사용 Temperature, Top-K, Top-P 값 다양하게 변경하며 출력 결과물의 창의성, 일관성, 예측 가능성 변화 관찰 및 기록 수행

제로샷 프롬프팅 (Zero-shot Prompting)

상세 설명

- 가장 간단한 형태. 모델에 작업 설명만 제공
  별도 예시(demonstration) 없이 작업 요청 방식
- LLM이 사전 훈련 데이터만으로 작업 이해 및 수행 가능 시 효과적임

예시

- 프롬프트: 영화 리뷰를 긍정(POSITIVE), 중립(NEUTRAL), 부정(NEGATIVE)으로 분류
- 리뷰: "이 영화는 AI가 통제되지 않고 진화할 경우 인류가 향하게 될 방향을 보여주는 불안한 연구임. 이런 걸작 같은 영화가 더 많았으면 좋겠음."
- 감성:
- 예상 출력: POSITIVE

제로샷 프롬프팅 (Zero-shot Prompting)

실습

- 주제: 텍스트 분류
- 미션: 주어진 문장("오늘 날씨가 정말 좋아서 기분이 상쾌함.")의 감정(긍정/부정/중립) 분류 제로샷 프롬프트 작성 및 실행

원샷 & 퓨샷 프롬프팅 (One-shot & Few-shot Prompting)

상세 설명

- 모델의 작업 이해도 향상 위해 프롬프트 내 하나(One-shot) 또는 여러 개(Few-shot) 예시 포함 방식
- 특히 원하는 출력 형식/패턴 존재 시 효과적임
- Few-shot: 요청할 패턴 명확히 제시, 성공 가능성 높임
- 일반적으로 3~5개 이상 예시 사용 권장, 작업 복잡도/모델 입력 길이 제한 따라 조절 필요
- 예시는 작업 관련성 높고, 다양하며, 오류 없이 잘 작성되어야 함

  엣지 케이스(Edge case)* 포함 권장

* '특별하거나', '까다롭거나', '일반적이지 않은 예외적인 상황'의 예시

"나쁘지 않았어요." (부정어가 있지만 의미는 긍정에 가까움 - 까다로운 경우)
"영화." (매우 짧은 입력 - 특별한 경우)
"" (입력이 없음 - 예외적인 경우)
"재미있다고 하기에는 좀 그렇고, 없다고 하기에도 좀 그렇네." (애매모호한 경우)

원샷 & 퓨샷 프롬프팅 (One-shot & Few-shot Prompting)

예시 (피자 주문 파싱)

- 프롬프트:

```
"""
고객 피자 주문 유효 JSON 파싱:
EXAMPLE:
치즈, 토마토 소스, 페퍼로니 들어간 작은 피자 하나.
JSON Response:
{
  "size": "small",
  "type": "normal",
  "ingredients": [["cheese", "tomato sauce", "peperoni"]]
}
```

원샷 & 퓨샷 프롬프팅 (One-shot & Few-shot Prompting)

예시 (피자 주문 파싱)

EXAMPLE:

토마토 소스, 바질, 모짜렐라 들어간 큰 피자 하나 가능?

JSON Response:

```
{
  "size": "large",
  "type": "normal",
  "ingredients": [["tomato sauce", "bazel", "mozzarella"]]
}
```

이제, 큰 피자 하나. 반은 치즈와 모짜렐라, 나머지 반은 토마토 소스, 햄, 파인애플

JSON Response:

```
```

원샷 & 퓨샷 프롬프팅 (One-shot & Few-shot Prompting)

예시 (피자 주문 파싱)

- 예상 출력:

```json
{
  "size": "large",
  "type": "half-half",
  "ingredients": [["cheese", "mozzarella"], ["tomato sauce", "ham", "pineapple"]]
}
```

실습

- 주제: 형식 변환
- 미션: 날짜 형식 "YYYY년 MM월 DD일" → "MM/DD/YYYY" 변환 작업 위한 Few-shot 프롬프트 작성. 최소 3개 예시 포함. (예: "2025년 4월 17일 → 04/17/2025")

시스템, 컨텍스트, 역할 프롬프팅 (System, Contextual, Role Prompting)

상세 설명

- LLM 응답 방식 유도 세 가지 기법, 종종 중첩 사용
- **시스템 프롬프팅 (System Prompting)**:
  * 모델의 전반적 **작동 방식/목표(번역, 분류 등) 설정, 출력 형식/제약 조건** 등
  * 출력 형식 JSON 지정 또는 안전성/독성 제어 위해 사용 가능 (예: "정중하게 답변할 것")
- **역할 프롬프팅 (Role Prompting)**:
  * 모델에 특정 **캐릭터/정체성**(교사, 여행 가이드, 유머 작가 등) 부여
    해당 역할 맞는 어조/스타일/지식 수준 응답 유도함
  * 응답 품질, 관련성, 효과성 향상 가능
    다양한 스타일(직설적, 유머러스, 설득적 등) 적용 가능

시스템, 컨텍스트, 역할 프롬프팅 (System, Contextual, Role Prompting)

상세 설명

- 컨텍스트 프롬프팅 (Contextual Prompting):
  * 현재 대화/작업 관련 구체적 배경 정보/세부 사항 제공, 모델의 질문 뉘앙스 파악 및 맞춤형 응답 생성 도움
  * 매우 동적이며 현재 작업 특화 정보 제공함

예시

- 시스템: (JSON 출력 지정)
- 역할: ("여행 가이드 역할 수행. 내 위치 기반 근처 방문 3곳 추천 요청.")
- 컨텍스트: ("컨텍스트: 80년대 레트로 아케이드 비디오 게임 블로그 글 작성 중. 기사 주제 3가지 설명과 함께 제안 요청.")

시스템, 컨텍스트, 역할 프롬프팅 (System, Contextual, Role Prompting)

실습

- 주제: 복합 프롬프트 작성
- 미션 1 (역할+컨텍스트): "조선 시대 왕 역할, 어려운 백성 삶 문제 해결 위한 현실적 방안 3가지 제시" 프롬프트 작성 및 실행
- 미션 2 (시스템+컨텍스트): "다음 텍스트 3문장 요약 및 핵심 키워드 5개 JSON 형식 반환 텍스트: [긴 뉴스 기사 삽입]" 프롬프트 작성 및 실행

스텝-백 프롬프팅 (Step-back Prompting)

상세 설명

- 복잡한 질문이나 문제에 대해 더 정확하고 논리적인 답변을 생성하도록 유도
- LLM의 관련 배경 지식/추론 과정 활성화, 더 정확하고 통찰력 있는 답변 생성 도움
- LLM의 단순 패턴 매칭 넘어 추론능력 향상, 오류 감소, 구조적인 사고 유도

예시

1. 추상화 (Abstraction): LLM에게 주어진 구체적인 질문(예: "서울에서 부산까지 자전거로 가장 빨리 가는 방법은?") 대신, 이와 관련된 더 일반적인 질문이나 핵심 원칙을 먼저 생각하도록 유도

* 스텝백 질문 예시: "장거리 자전거 여행 경로를 최적화하는 주요 원칙은 무엇인가?"
  또는 "도시 간 이동 시 고려해야 할 일반적인 요소는 무엇인가?"

스텝-백 프롬프팅 (Step-back Prompting)

예시

2. **개념/원칙 도출**: LLM은 이 '스텝백' 질문을 바탕으로 관련된 일반적인 지식이나 원칙 상기
 * 도출된 개념 예시: "최단 거리, 도로 상태(포장/비포장, 경사도), 안전성(자전거 도로 유무), 휴식 지점, 날씨 등을 고려해야 한다."

3. **구체적인 문제 해결**: 이제 LLM은 1단계와 2단계에서 얻은 일반적인 원칙과 개념들을 활용하여, 원래의 구체적인 질문("서울-부산 자전거 최단 경로")에 대한 답변을 논리적으로 추론
 * 최종 답변 추론: "위 원칙들에 따라, 최단 거리를 우선하되 국토종주 자전거길과 같은 안전하고 포장된 경로를 중심으로, 급격한 경사도를 피하고 중간 보급 지점을 고려하여 경로를 계획하는 것이 가장 빠를 것이다. 구체적인 추천 경로는 다음과 같다..."

스텝-백 프롬프팅 (Step-back Prompting)

실습

- 주제: 문제 해결 방안 제시
- 미션: "우리 회사 마케팅 팀 가장 큰 문제점 및 해결 방안?" 질문에 스텝-백 프롬프팅 적용
  1. "성공적 마케팅 팀 핵심 요소?" 등 일반 질문 후 답변 획득
  2. 획득 답변 컨텍스트 활용, 원래 질문 답변 생성 프롬프트 작성 및 실행

생각의 사슬 (Chain of Thought - CoT)

상세 설명

- LLM이 최종 답변 전, 문제 해결 위한 중간 추론 단계 생성 유도 기법임
- 수학 문제, 논리 추론 등 복잡 작업 정확도 크게 향상 가능
- "단계별로 생각해보자 (Let's think step by step)" 문구 추가 등 간단 방식 적용 가능 (Zero-shot CoT)
- Few-shot 방식 결합 시 더 복잡 문제에 효과적임
- 장점: 구현 용이, 효과적, 추론 과정 확인 가능(해석 가능성 높음), 디버깅 용이. 모델 버전 변경 시 비교적 안정적 성능 보임
- 단점: 추론 과정 포함 출력으로 토큰 사용량 증가, 비용/시간 증가

생각의 사슬 (Chain of Thought - CoT)

예시 (나이 계산 문제)

- CoT 미적용 프롬프트: "내가 3살 때, 누나는 내 나이 3배. 지금 나는 20살. 누나 나이?"
  → (오답 가능성 높음, 예: 63세)
- Zero-shot CoT 프롬프트: "내가 3살 때, 누나는 내 나이 3배. 지금 나는 20살. 누나 나이 몇 살인지 단계별로 생각해보자."
- 예상 출력 (추론 과정 포함):

```
1. 내 현재 나이 'x' 가정
2. 내가 3살 때, 내 나이 3살
3. 그때 누나 나이 내 나이 3배, 즉 $3 * 3 = 9$살
4. 지금 나는 20살, 3살 때부터 $20 - 3 = 17$년 지남
```

생각의 사슬 (Chain of Thought - CoT)

예시 (나이 계산 문제)

5. 내가 3살 때 누나 나이 9살, 내 나이 17년 증가했으므로 파트너 나이도 17년 증가
6. 따라서 누나 현재 나이 $9 + 17 = 26$살
결론: 누나는 26살임
“”

- Few-shot CoT: (다른 예시 먼저 제시 후 같은 방식 풀이 유도)

실습

- 주제: 논리 퍼즐 해결
- 미션: 간단한 논리 퍼즐(예: "A(진실), B(거짓), C(번갈아) 세 사람. 각자에게 '옆 사람 진실 말함?' 질문 시 가능 답변 조합?") CoT 프롬프트 이용 해결 유도 및 추론 과정 확인

자기 일관성 (Self-consistency)

상세 설명

- CoT 한계(단일 추론 경로의 취약성, 결과의 일관성 부족 가능성) 보완 기법
  동일 프롬프트 여러 번(높은 Temperature 설정) 추론 경로 생성 후,
  가장 많이 나온 답변 최종 선택 방식임
- 다양한 관점 문제 탐색 및 가장 일관성 있는 결론 도출 및 정확도 제고
- 단계:
  1. 동일 프롬프트로 여러 번 CoT 추론 생성 (높은 Temperature 사용)
  2. 각 결과에서 최종 답변 추출
  3. 가장 빈번 등장 답변 최종 선택

자기 일관성 (Self-consistency)

예시 (이메일 중요도 분류)

- 프롬프트: (해킹 의심 이메일 내용 제시 후) " 위 이메일 중요(IMPORTANT) 또는 비중요(NOT IMPORTANT) 분류, 단계별 생각 및 이유 설명 요청"
- 실행: 해당 프롬프트 Temperature 1.0 설정 후 여러 번 실행
- 관찰: 실행 시마다 추론 과정 및 최종 결론(IMPORTANT 또는 NOT IMPORTANT) 달라질 수 있음
- 최종 결정: 여러 실행 결과 중 가장 많이 나온 결론(예: 3번 중 2번 IMPORTANT) 최종 답변 채택

자기 일관성 (Self-consistency)

실습

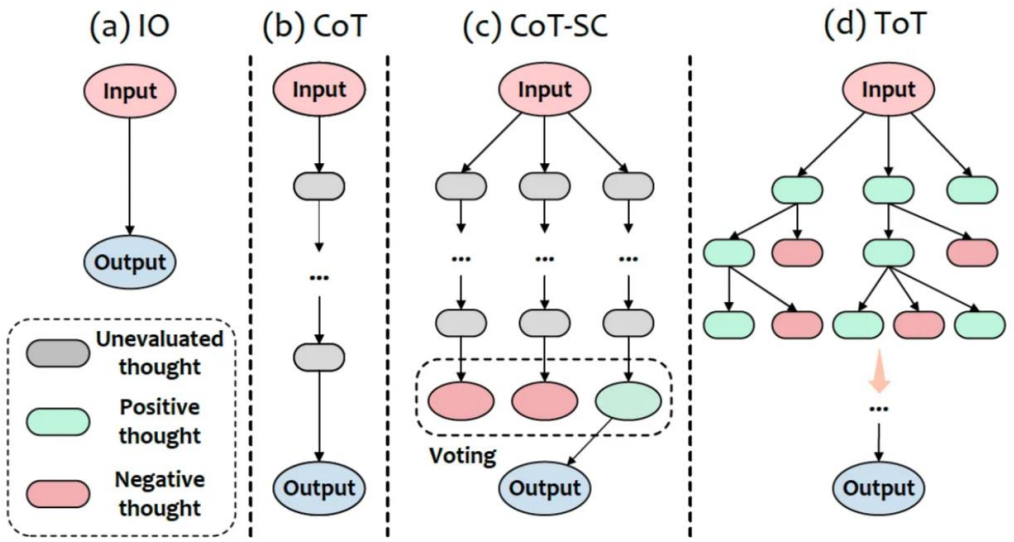
- 주제: 애매한 질문 답변 생성
- 미션: CoT 실습 사용 논리 퍼즐 또는 다른 애매한 질문에 Self-consistency 기법 적용. Temperature 높게 설정, 3~5회 실행하여 각 답변 기록, 가장 많이 나온 답변 최종 결과 선택

생각의 트리 (Tree of Thoughts - ToT)

상세 설명

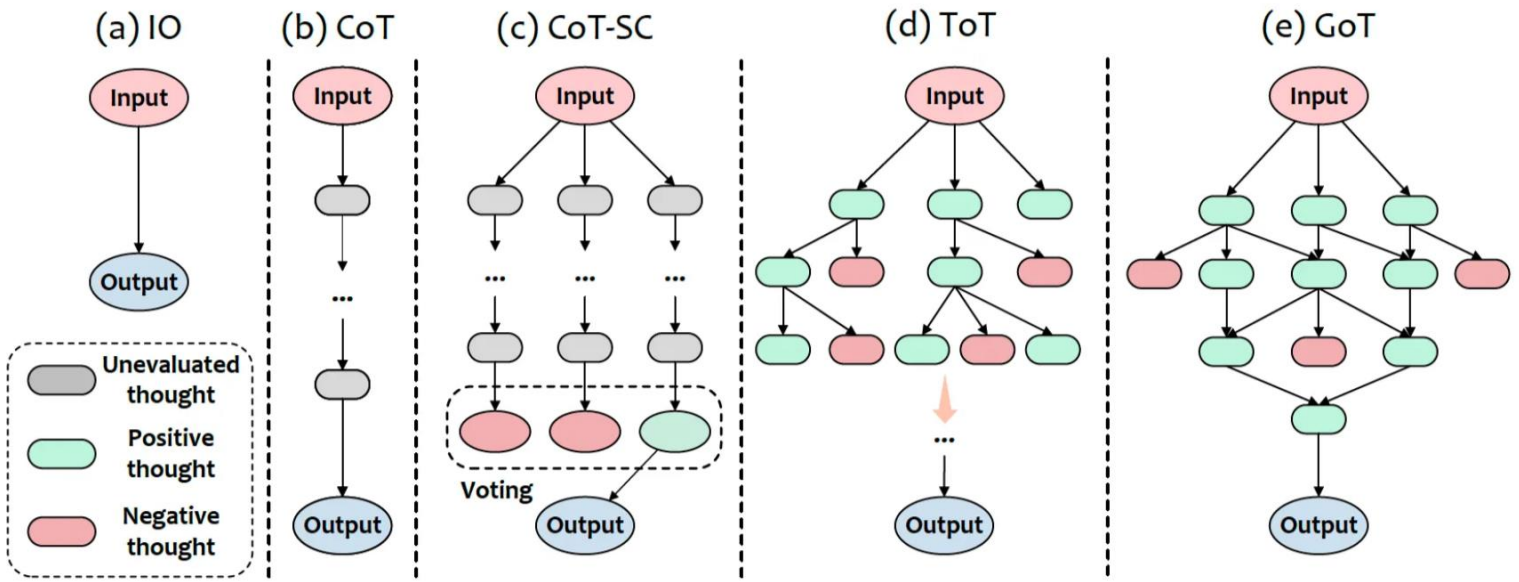
- CoT(단일 경로 추론)와 달리, ToT는 여러 추론 경로 동시 탐색/평가하는 트리 구조 사용함
- 각 노드는 문제 해결 위한 중간 생각(thought) 나타냄, 모델은 여러 분기(branch) 탐색하며 최적 해결책 모색함
- 탐색 필요한 복잡 문제 해결에 더 적합함

예시 (CoT와 ToT 시 각적 비교)



생각의 트리 (Tree of Thoughts - ToT)

예시
(CoT와 ToT 시각적 비교)



실습
(개념 이해)

- 주제: 문제 해결 전략 비교
- 미션: 복잡 문제(예: "24 게임 - 숫자 4개 사칙연산 조합 24 만들기") 해결 가정 시,
  CoT 방식과 ToT 방식 접근 차이점
  (CoT는 단일 시도, ToT는 여러 가능 연산 조합 동시 탐색)

ReAct (Reason & Act)

상세 설명

- LLM 추론(Reason) 능력과 외부 도구/API 사용 행동(Act) 능력 결합 패러다임
- LLM 스스로 필요 정보 검색/코드 실행 등 '행동' 계획/수행, 결과를 바탕으로 다시 '추론'하여 최종 목표 달성함
- 생각-행동-관찰 루프 통해 작동, 최신 정보 검색/계산 등 LLM 자체 한계 보완 효과적임 (에이전트 모델링 기초)
- 실제 구현에는 LangChain 등 프레임워크 및 외부 도구(검색 API 등) 연동 필요함

자동 프롬프트 엔지니어링 (Automatic Prompt Engineering - APE)

상세 설명

- 프롬프트 자체를 LLM 이용 자동 생성/평가/최적화 방법임
- 사람 개입 줄이고 다양한 프롬프트 후보 탐색, 성능 향상 가능
- 단계:
  1. 초기 프롬프트 LLM 제공, 여러 변형 프롬프트 생성
  2. 생성 후보 프롬프트 특정 기준(BLEU, ROUGE 점수 또는 실제 작업 성능 등) 평가
  3. 가장 높은 점수 받은 프롬프트 선택 또는 추가 수정 후 반복

자동 프롬프트 엔지니어링 (Automatic Prompt Engineering - APE)

예시 (밴드 티셔츠 주문 문구 생성)

- 생성 프롬프트: "밴드 상품 티셔츠 웹샵. 챗봇 학습 위해 '메탈리카 티셔츠 S 사이즈 하나요' 주문 다양한 표현 방식 필요. 의미 같게 유지, 10가지 변형 문구 생성 요청"
- LLM 출력 (후보 프롬프트):
  1. 스몰 사이즈 메탈리카 티셔츠 구매 희망
  2. 작은 사이즈 메탈리카 티셔츠 주문 가능? ... (10가지 변형)

실습 (개념 이해)

- 주제: APE 프로세스 설계
- 미션: 특정 작업(예: 고객 리뷰 요약) 위한 프롬프트 APE 방식 최적화 가정 시, 초기 프롬프트 내용, 생성 후보 프롬프트 평가 방식 구체적 계획 수립

모듈 4: 코드 프롬프팅 (Code Prompting)

코드 작성 프롬프트 (Prompts for writing code)

상세 설명	- 원하는 기능/로직 설명, 특정 프로그래밍 언어 코드 생성 요청함 - 반복 작업 자동화 스크립트 작성 등 유용, 개발 생산성 향상 가능 - 생성 코드 반드시 검토 및 테스트 필요함
예시 (Bash 스크립트 생성)	- 프롬프트: "폴더 이름 묻는 Bash 코드 스니펫 작성. 폴더 내용 가져와 내부 모든 파일 이름 앞 'draft_' 붙여 이름 변경하도록 요청" - 출력: (Bash 스크립트 코드 - 주석 포함)
실습	- 주제: 간단한 함수 생성 - 미션: "주어진 숫자 소수(prime number) 판별 Python 함수 작성 요청." 프롬프트 실행 및 생성 코드 검토

모듈 4: 코드 프롬프팅 (Code Prompting)

코드 설명 프롬프트 (Prompts for explaining code)

상세 설명	- 주어진 코드 스니펫 기능, 각 부분 작동 방식 자연어 설명 요청함 - 타인 코드 이해 또는 복잡 로직 파악 도움됨
예시 (Bash 스크립트 설명)	- 프롬프트: "아래 Bash 코드 설명 요청: [이전 예시 생성 Bash 코드 (주석 제거)]" - 출력: (코드 각 단계별 기능 설명)
실습	- 주제: 코드 분석 - 미션: 이전 실습 생성 소수 판별 Python 함수 코드 입력, 해당 코드 설명 요청 프롬프트 실행

모듈 4: 코드 프롬프팅 (Code Prompting)

코드 번역 프롬프트 (Prompts for translating code)

상세 설명	- 특정 프로그래밍 언어 작성 코드 다른 언어 변환 요청함 - 레거시 코드 마이그레이션 또는 다른 환경 코드 재사용 시 유용함
예시	- 프롬프트: "아래 Bash 코드 Python 스니펫 번역 요청: [Bash 파일 이름 변경 스크립트]" - 출력: (동일 기능 Python 코드)
실습	- 주제: 언어 변환 - 미션: 간단한 JavaScript 코드(예: 배열 합계 계산) 작성, Python 코드 번역 요청 프롬프트 실행

모듈 4: 코드 프롬프팅 (Code Prompting)

코드 디버깅 및 검토 프롬프트 (Prompts for debugging and reviewing code)

상세 설명	- 오류 발생 코드 및 오류 메시지 함께 제공, 문제 원인 탐색 및 해결 방법 제안 요청함 - 코드 잠재 문제점/개선 방안 검토 요청 가능

모듈 4: 코드 프롬프팅 (Code Prompting)

코드 디버깅 및 검토 프롬프트 (Prompts for debugging and reviewing code)

예시 (Python 코드 오류 수정 및 개선)

- 프롬프트:

```
"""
아래 Python 코드 오류 발생:
Traceback (most recent call last):
  File "...", line 7, in <module>
    text = toUpperCase(prefix)
NameError: name 'toUpperCase' is not defined
잘못된 부분 디버깅 및 코드 개선 방법 설명 요청.
[오류 있는 Python 코드 전체]
"""
```

- 출력: ('toUpperCase' 함수 미정의 오류 지적, 'prefix.upper()' 사용 제안, 수정 코드 제공 추가 개선 사항 제안 - 확장자 유지, 공백 처리, f-string 사용, 오류 처리 등)

모듈 4: 코드 프롬프팅 (Code Prompting)

코드 디버깅 및 검토 프롬프트 (Prompts for debugging and reviewing code)

실습

- 주제: 오류 해결 및 코드 리뷰
- 미션: 의도적 작은 오류 포함 코드(변수명 오타, 잘못된 함수 호출 등) 작성, 오류 메시지와 함께 LLM에 디버깅/코드 리뷰 요청 프롬프트 실행. LLM 제안 평가

모듈 5: 프롬프트 엔지니어링 모범 사례 및 워크플로우

핵심 모범 사례

상세 설명	- 효과적 프롬프트 설계 위한 주요 원칙 - **예시 제공**: One-shot/Few-shot 매우 효과적임. 원하는 결과물 형태/스타일 명확히 제시 가능 - **단순하게 디자인**: 프롬프트 명확/간결해야 함. 복잡/불필요 정보 지양 행동 나타내는 동사(분류, 생성, 요약 등) 사용 권장 - **출력 구체적 명시**: 원하는 결과물 길이, 형식, 스타일, 포함/제외 내용 등 구체적 지시 - **제약 조건보다 지침 사용**: "하지 마라" 제약보다 "이렇게 하라" 긍정적 지침이 더 효과적 모델에 명확 목표 제시 및 유연성 부여, 제약은 안전성/특정 형식 요구 시 신중 사용 - **최대 토큰 길이 제어**: 설정값 또는 프롬프트 내 지시(예: "트윗 길이로 설명") 통해 응답 길이 관리

모듈 5: 프롬프트 엔지니어링 모범 사례 및 워크플로우

핵심 모범 사례

상세 설명

- **프롬프트에 변수 사용**: 반복 사용/동적 변경 필요 부분 변수 처리
  프롬프트 재사용성/유연성 높임 (애플리케이션 통합 시 유용)
- **입력 형식/작성 스타일 실험**: 동일 목표라도 질문, 명령문, 지시문 등 다양한 형식/어조 시도하며 최적 방식 탐색
- **퓨샷 분류 작업 시 클래스 혼합**: 예시 순서 모델 과적합 방지 위해, Few-shot 예시에서 가능 결과 클래스 골고루 섞어 제시
- **모델 업데이트 적응**: 모델 지속 발전하므로, 새 버전 모델 테스트 및 특성 맞게 프롬프트 조정 필요. Vertex AI Studio 등 도구 활용
- **출력 형식 실험**: 특히 데이터 추출, 분류, 정렬 등 작업에서 JSON/XML 등 구조화 형식 사용 시 일관성 확보, 환각(hallucination) 감소, 후처리 용이성 등 이점 있음

모듈 5: 프롬프트 엔지니어링 모범 사례 및 워크플로우

핵심 모범 사례

실습

- 주제: 프롬프트 개선
- 미션: 다음 '나쁜' 프롬프트 모범 사례 적용 개선
  * (나쁜 프롬프트): "비디오 게임 콘솔에 대해 작성"
  * (개선 방향): 길이(3문단), 내용(상위 5개 콘솔), 스타일(정보성, 대화체) 구체화, 긍정적 지침 사용 등

JSON 활용 심화 (JSON Repair & Schemas)

상세 설명

- JSON Repair: LLM 생성 JSON 출력 토큰 제한 등으로 중간 끊겨 유효하지 않은 JSON 될 수 있음. `json-repair` 등 라이브러리 사용, 깨진 JSON 자동 복구 도움됨
- Working with Schemas: JSON은 출력뿐 아니라 입력에도 유용함. JSON Schema 사용, 입력 데이터 구조/타입 미리 정의 시, LLM이 데이터 의미/관계 명확 파악 및 관련 정보 집중하여 더 정확 결과 생성 유도 가능. 특히 대량 데이터 처리/복잡 애플리케이션 통합 시 유용함

예시 (상품 설명 생성 시 Schema 활용)

```
{
  "name": "Wireless Headphones",
  "category": "Electronics",
  "price": 99.99,
  "features": ["Noise cancellation", "Bluetooth 5.0", "20-hour battery life"],
  "release_date": "2023-10-27"
}
```

JSON 활용 심화 (JSON Repair & Schemas)

실습 (개념 이해)

- 주제: 구조화된 입/출력 설계
- 미션: '회의록 요약 및 참석자별 할 일 목록 추출' 작업 자동화 가정 시, LLM 입력으로 정의할 Schema 및 제공 데이터, 출력으로 기대할 JSON 구조 설계

협업 및 문서화 (Collaboration & Documentation)

상세 설명

- 협업: 여러 사람 함께 프롬프트 작성/테스트 시 다양한 접근 방식 시도, 성능 편차 확인, 더 나은 결과 획득 가능
- 문서화: 프롬프트 엔지니어링은 반복 과정이므로, 각 시도 상세 기록 매우 중요함. 모델, 설정값, 프롬프트 내용, 출력 결과, 성공 여부, 피드백 등 체계적 기록 시 추후 재검토, 디버깅, 모델 버전 변경 시 비교 등에 큰 도움됨
- 문서화 템플릿: (Name, Goal, Model, Settings, Prompt, Output 등 포함)
- 코드베이스 관리: 완성 프롬프트는 코드와 분리된 파일 관리 유지보수 용이함
- 자동화된 테스트: 이상적으로 자동화 테스트/평가 절차 통해 프롬프트 일반화 성능 검증 필요

협업 및 문서화 (Collaboration & Documentation)

예시

Goal	Write a storyline for a level of a first-person shooter video game.		
Model	gemini-pro		
Temperature	1	Token Limit	1024
Top-K	40	Top-P	0.8
Prompt	`Write a one paragraph storyline for a new level of a first-person shooter video game that is challenging and engaging.`		
Output	The level begins with the player's squad being ambushed in a dense urban area by a relentless enemy faction. Players must fight their way through a maze of back alleys and dilapidated buildings, utilizing stealth and precision to eliminate enemy forces and avoid detection. The player must use their environment to their advantage, setting traps, utilizing cover, and making calculated moves to survive. Along the way, they uncover intel suggesting the enemy is planning a devastating attack, and the player must race against time to gather evidence and relay it back to their headquarters before the enemy can carry out their plans.		

실습

- 주제: 프롬프트 시도 기록
- 미션: 금일 실습 시도 프롬프트 중 하나 선택, 제공된 문서화 템플릿 형식 맞춰 상세 내용 기록

CoT 모범 사례

상세 설명	- 답변은 추론 뒤에: CoT는 추론 과정 먼저 생성, 최종 답변 마지막 제시 필요. 추론 과정 자체가 후속 토큰 생성 영향 미침 - 답변 추출 용이성: 프롬프트 설계 시 추론 과정과 최종 답변 명확히 구분, 쉽게 추출 가능하도록 설계 필요 (Self-consistency 적용 시 중요) - Temperature는 0으로: CoT는 일반적 단일 정답 찾는 추론 사용되므로, 가장 확률 높은 경로 따라가도록 Temperature 0 설정 권장 (greedy decoding)

모듈 6: 종합 실습 및 마무리

종합 실습 (워크샵)

주제	- 특정 시나리오 기반 프롬프트 설계
미션 (예시)	- 시나리오 1: 고객 서비스 챗봇 FAQ 답변 생성 프롬프트 최적화 (Few-shot, Role Prompting, JSON 출력 활용) - 시나리오 2: 주어진 연구 논문 요약 및 핵심 기여점 CoT 방식 설명 프롬프트 작성 - 시나리오 3: 최신 IT 기술 동향 블로그 게시물 초안 ReAct(개념적) 또는 Step-back 방식 작성 프롬프트 설계
진행	- 조별/개인별 시나리오 선택, 프롬프트 설계 및 Vertex AI Studio 등 테스트 통한 반복 개선. 과정/결과 문서화 및 발표/공유

(참고) BLEU(Bilingual Evaluation Understudy)

BLEU

사용분야	- 기계 번역 품질 평가
핵심 아이디어	생성된 번역문이 사람이 생성한 참조 번역문들과 얼마나 유사한 단어 순서(n-gram)를 포함하고 있는지 측정 정밀도(Precision)에 중점 생성된 문장의 단어/구절이 얼마나 참조 문장에도 등장하는가에 대한 평가
계산방식	- 생성된 문장과 참조 문장(들) 간에 겹치는 n-gram 계산 - 단순히 겹치는 수를 세면 특정 단어만 반복해서 점수를 높일 수 있으므로, 참조 문장에 등장한 횟수만큼만 카운트하는 등 보정 (Modified Precision) - 짧은 문장을 생성하면 겹치는 단어 비율(정밀도)이 높아지기 쉬우므로, 생성된 문장이 참조 문장보다 너무 짧으면 페널티를 부여 (Brevity Penalty) - 일반적으로 1-gram부터 4-gram까지의 정밀도 점수를 기하 평균, 짧음 페널티를 곱하여 최종 점수를 계산

(참고) ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

ROUGE

사용분야	- 텍스트 요약 품질 평가 (기계 번역 등 다른 생성 과제에도 사용)
핵심 아이디어	생성된 요약문이 사람이 만든 참조 요약문의 중요한 정보(단어/구절)를 포함 여부 측정 재현율(Recall)에 중점, F1-Score 사용 참조 문장에 있는 단어/구절이 얼마나 생성된 문장에도 등장하는가에 대한 평가 BLEU와 유사하게 n-gram 비교에 기반하지만, 재현율 계산에 초점
계산방식	- ROUGE-N: 참조 요약문에 있는 n-gram 중 얼마나 많은 비율이 생성된 요약문에도 나타나는지를 측정(ROUGE-1은 unigram, ROUGE-2는 bigram 비교) 보통 재현율(Recall)/정밀도(Precision)의 조화 평균인 F1-Score 특정 - ROUGE-L: 단어의 순서를 고려하는 최장 공통 부분 서열(Longest Common Subsequence) 을 사용, 연속되지 않더라도 순서가 맞는 가장 긴 단어 시퀀스를 찾아 길이를 비교 문장 수준의 구조적 유사성을 더 잘 반영 - ROUGE-S/SU: 단어 쌍(skip-bigram, 중간에 다른 단어 포함가능)의 겹침을 측정

(참고) JSON (JavaScript Object Notation)

JSON 예시

예시

```
{
  "상품명": "맛있는 제주 감귤",
  "가격": 25000,
  "무료배송": true,
  "옵션": [
    "선물 포장 요청",
    "영수증 발행",
    "리뷰 작성 이벤트 참여"
  ],
  "재구매의사": false,
  "평점": 4.5,
  "원산지": "대한민국 제주특별자치도",
  "판매자정보": {
    "상호명": "제주 행복 농장",
    "대표자명": "김감귤",
    "사업자번호": "123-45-67890",
    "주소": "제주특별자치도 서귀포시 감귤로 123",
    "고객센터운영": true
  }
}
```