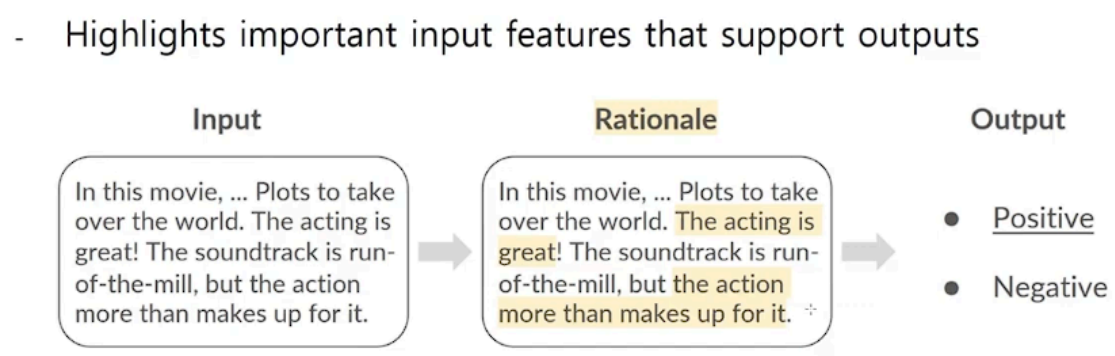
06_Explaining Generative LLMs: Prompting-based Explanations

1. Extractive Rationales / Feature Attributions

우선 각각이 무엇인지 대략적으로 알아보자.

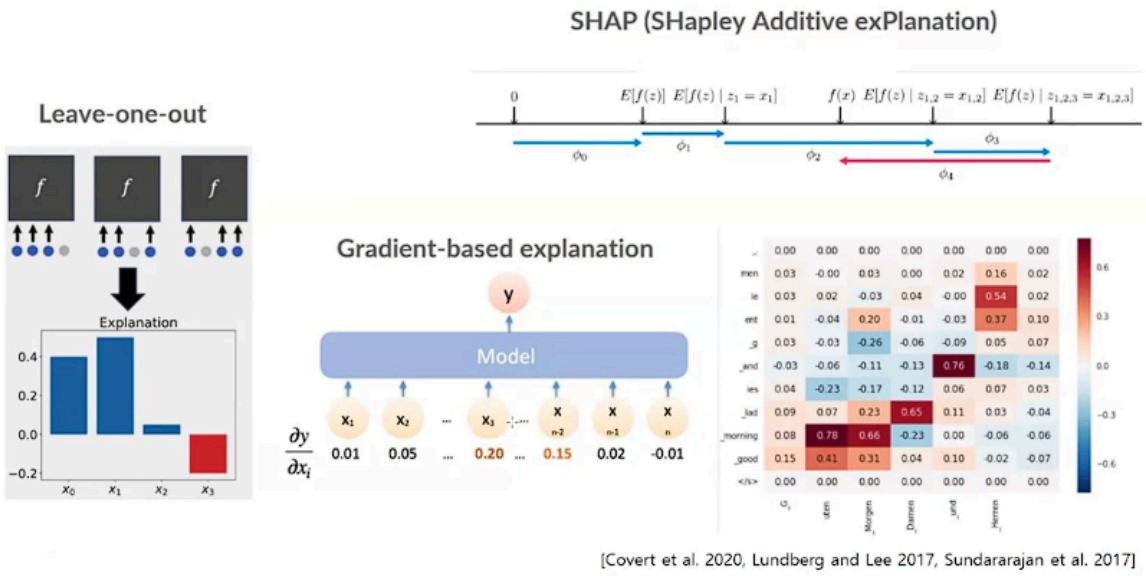
먼저 Extractive Rationales는 모델이 출력에 영향을 미친 중요한 입력 특성(단어나 문장 부분 등)을 강조하는 방식이다.

- input : 원래 문장
- Rationale : 모델이 중요하다고 판단한 부분을 강조
- Output : 긍정적인 감정을 나타내는 요소를 분석하여 최종 예측(P or N)



다음으로 Feature Attributions은 모델이 어떤 피처를 기반으로 예측을 수행하는지 분석하는 방법이다. 아래와 같은 종류들이 존재한다.

- Leave-One-Out (LOO)
- Gradient-Based Explanation
- SHAP



[Covert et al. 2020, Lundberg and Lee 2017, Sundararajan et al. 2017]

Challenges for LLMs

하지만, 위와 같은 방법들을 적용하기에는 대형 언어 모델의 문제점들이 존재한다.

1. **Computation Cost (연산 비용)**
   - LLM은 계산 비용이 많이 들며, 특히 대규모 데이터를 다룰 때 비용이 급격히 증가함.
2. **Low Efficiency in Long Context (긴 문맥에서의 비효율성)**
   - 긴 문맥을 유지할 때 모델의 성능이 저하되며, 문맥을 효율적으로 활용하는 것이 어려움.
3. **No Access to API-based models (API 기반 모델 접근 제한)**
   - 일부 API 기반의 모델에서는 내부 구조(e.g. gradient, attention score)에 접근할 수 없어 XAI 기법을 적용하기 어려움.

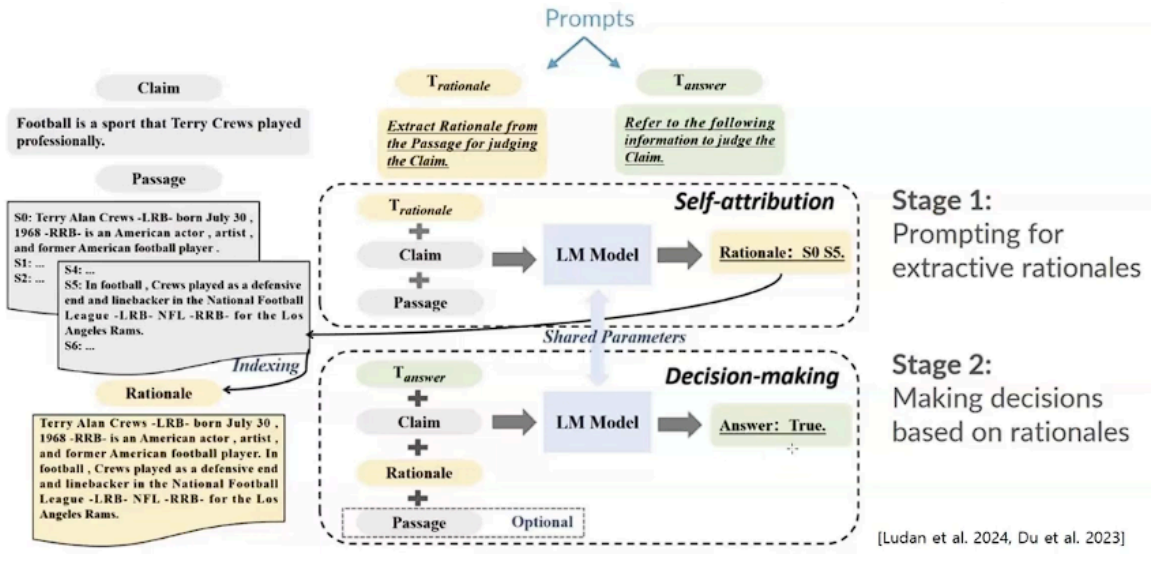
➡️ 위의 문제를 해결하기 위해 **Prompting-based Extractive Rationales/Feature Attributions**
을 활용할 수 있다!

Self-Attribution & Decision-Making

아래 이미지의 Prompts를 활용한 Extractive Rationales/Feature Attribution 예제를 보자.
이 접근법은 LLM이 특정한 이유(근거, Rationale)를 제공하도록 프롬프트를 설계하여 모델의 판단 과정을 설명하도록 하였다.

1. **Stage 1: Prompting for Extractive Rationales (추출형 근거 생성)**
   - 모델이 특정 주장을 판단할 때 어떤 정보(문장, 문맥 등)를 사용하여 근거로 삼는지를 추출하는 과정.
   - (Claim + Passage)에다가 $T_{rationale}$를 함께 제공하여 특정한 문장에서 S0, S5가 가장 중요한 근거로 선택되었다.
2. **Stage 2: Making Decisions Based on Rationales (근거를 바탕으로 최종 판단)**
   - 추출된 근거를 기반으로 최종적으로 모델이 True or False와 같은 결론을 내리는 과정.

➡️ 이 방식은 **프롬프트를 활용**하여 모델이 신뢰할 수 있는 방식으로 설명 가능한 출력을 제공하고, 기존의 그래디언트나 어텐션 기반 XAI 방법론과 달리, **모델의 내부 구조를 직접 활용하지 않고도 설명 가능성을 높일 수 있다!**



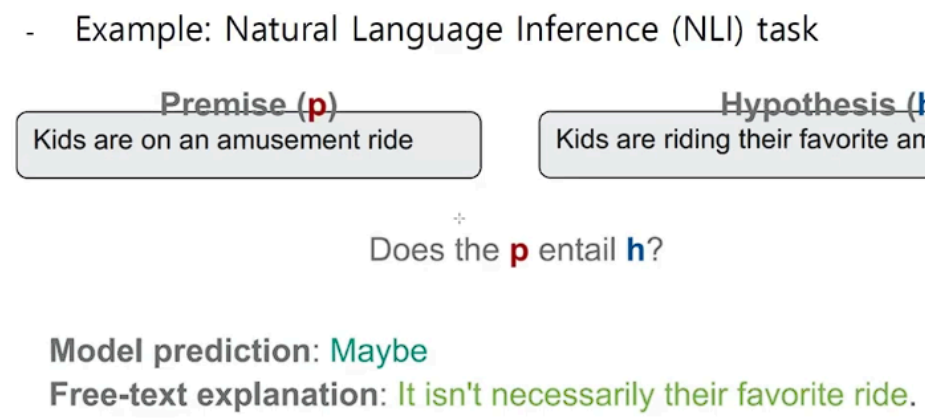
[Ludan et al. 2024, Du et al. 2023]

2. Free-Text Explanations

Free-Text Explanations에 대해 알아보기 전에, NLP의 대표적인 task인
Natural Language Inference (NLI)의 예제를 살펴보자.

“Kids are on an amusement ride” 전제 p가 “Kids are riding their favorite amusement ride”
가설 h를 포함하는가?
이 질문에 모델의 예측은 Maybe (애매함)이다.

전체 문장에서 아이들이 놀이기구를 타고 있다고만 했지, 가장 좋아하는 놀이기구를 타고 있다는 정보는 없기 때문이다. 이러한 모델의 Free-text Explanation을 통해 왜 모델이 애매한 답을 냈는지 설명할 수 있다.

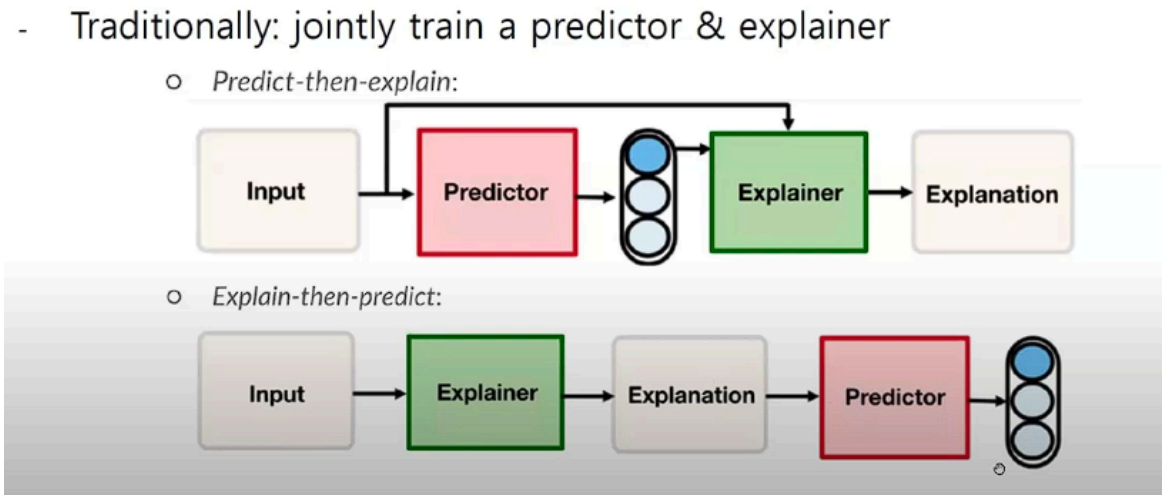


How to Generate Free-Text Explanations?

기존의 모델은 설명을 생성하는 방법으로 두 가지 접근 방식을 사용한다.

1. **Predict-then-Explain**
   - 먼저 예측 모델이 출력을 생성한 후
   - 설명 모델이 이에 대한 설명을 제공
   - 즉, 예측이 먼저, 설명이 나중
2. **Explain-then-Predict**
   - 먼저 설명 모델이 주어진 입력에 대한 설명을 생성하고
   - 이를 기반으로 예측 모델이 출력을 생성

✅ 위 방식은 설명 기반의 추론이 가능해지고, 모델이 결정에 대한 근거를 더 명확하게 제공할 수 있다.



❌ 하지만, 위와 같은 전통적인 방법을 통해 설명을 생성하려면 인간이 작성한 많은 예제가 필요하다. 대부분의 경우, 모델이 잘못된 예측을 해도 그 이유를 설명할 수 있는 신뢰할 만한 데이터가 부족하다.

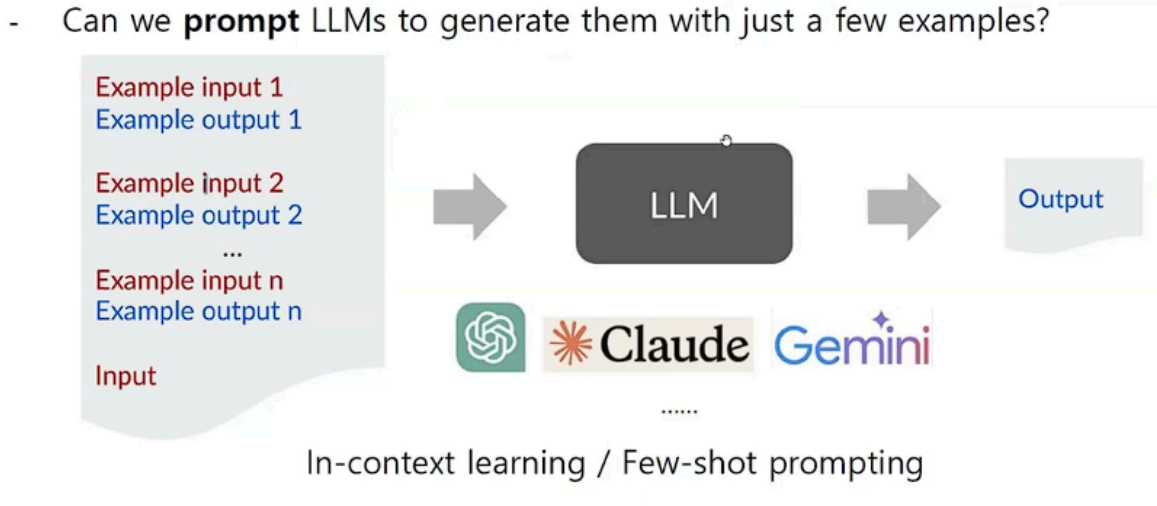
따라서 **Prompting**을 이용한 새로운 방식이 연구되고 있다.

Can We Prompt LLMs for Explanations?

In-context learning은 사전 훈련된 모델의 가중치를 변경하지 않고, 컨텍스트 내 예제만으로 학습할 수 있는 방법이다.

이는 제공되는 예제의 개수에 따라 크게 3가지로 구분되는데,
(Zero-shot, One-shot, Few-shot) Learning으로 구분된다.

이를 통해 인간이 설명 데이터를 직접 생성할 필요없이, 모델이 스스로 생성할 수 있고,
다양한 task에 적용 가능하다.



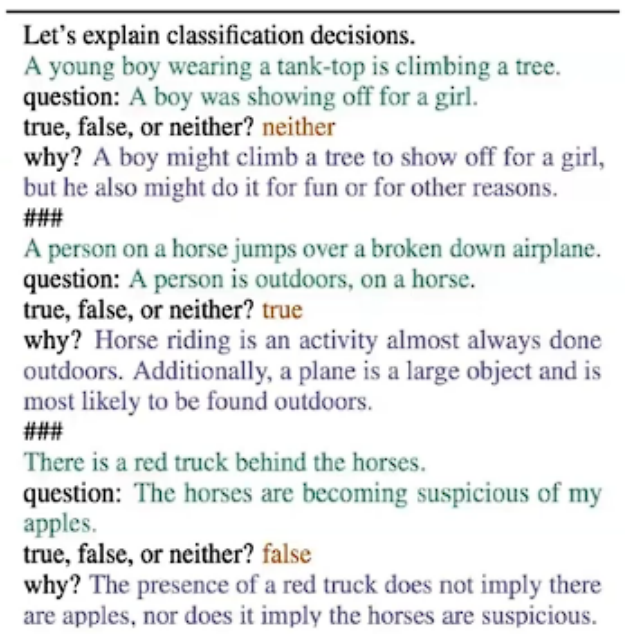
Prompting for Explanations

GPT-3 수준의 LLM은 단순한 task에서 Free-text Explanation을 생성할 수 있다.

- NLI (자연어 추론)
- Commonsense QA (상식 질의응답)
- Social Bias Detection (사회적 편향 감지)

위와 같은 예시들이 있다.

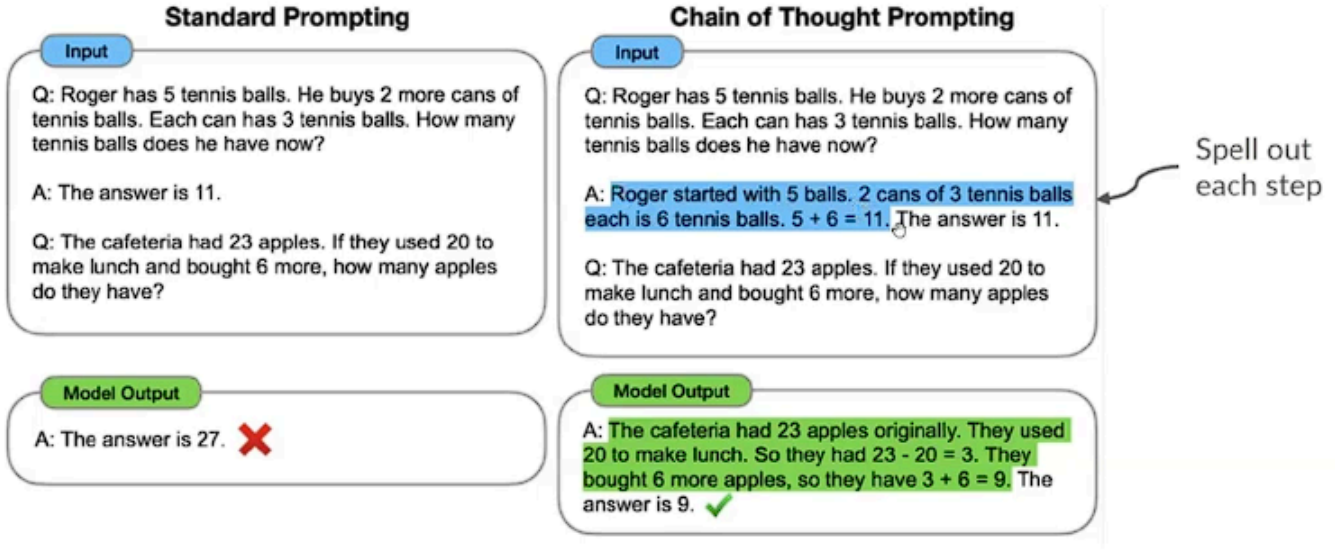
❌ 하지만, **Multi-step Reasoning**이 필요한 문제에서는 한계가 존재한다!


Let's explain classification decisions.
A young boy wearing a tank-top is climbing a tree.
question: A boy was showing off for a girl.
true, false, or neither? neither
why? A boy might climb a tree to show off for a girl, but he also might do it for fun or for other reasons.
###
A person on a horse jumps over a broken down airplane.
question: A person is outdoors, on a horse.
true, false, or neither? true
why? Horse riding is an activity almost always done outdoors. Additionally, a plane is a large object and is most likely to be found outdoors.
###
There is a red truck behind the horses.
question: The horses are becoming suspicious of my apples.
true, false, or neither? false
why? The presence of a red truck does not imply there are apples, nor does it imply the horses are suspicious.

Chain of Thought (CoT) Prompting

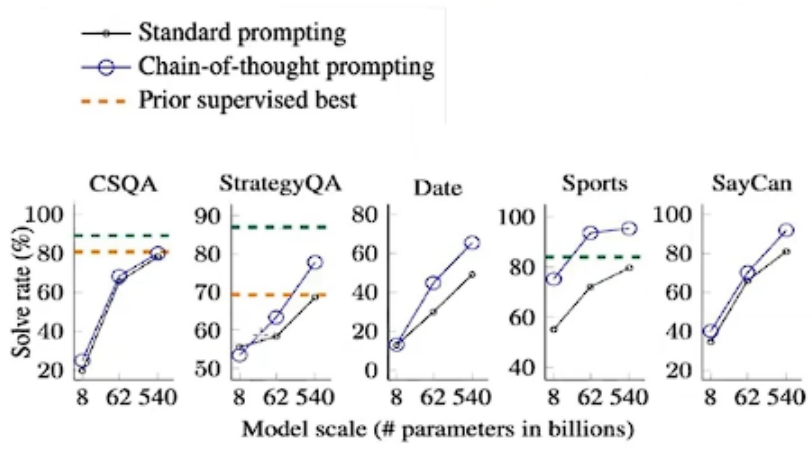
아래의 그림을 통해 CoT를 이해해보자.

- **Standard Prompting**
  - 단순히 질문을 입력하고, 모델이 정답을 예측.
  - 복잡한 문제에서는 논리적 추론 없이 정답을 내기 때문에 오류가 많다.
- **Chain of Thought (CoT) Prompting**
  - 모델이 추론 과정을 단계별로 서술하도록 유도함.
  - 다단계 reasoning이 필요한 문제에서 성능이 뛰어남.
  - 수학 문제, 논리적 reasoning, 자연어 추론(NLI)에서 큰 효과를 보인다.



아래 그림을 보면, CoT 방식을 사용했을 때 benchmark dataset에 대한 모델의 성능이 증가한 것을 알 수 있다. 또한, 모델 크기가 커질수록 CoT 방식이 더욱 효과적임을 알 수 있다.

다단계 추론을 요구하는 다양한 task(CSQA, StrategtQA 등)에서 CoT 방식이 전통적인 방법보다 더 높은 정확도를 달성하였다.

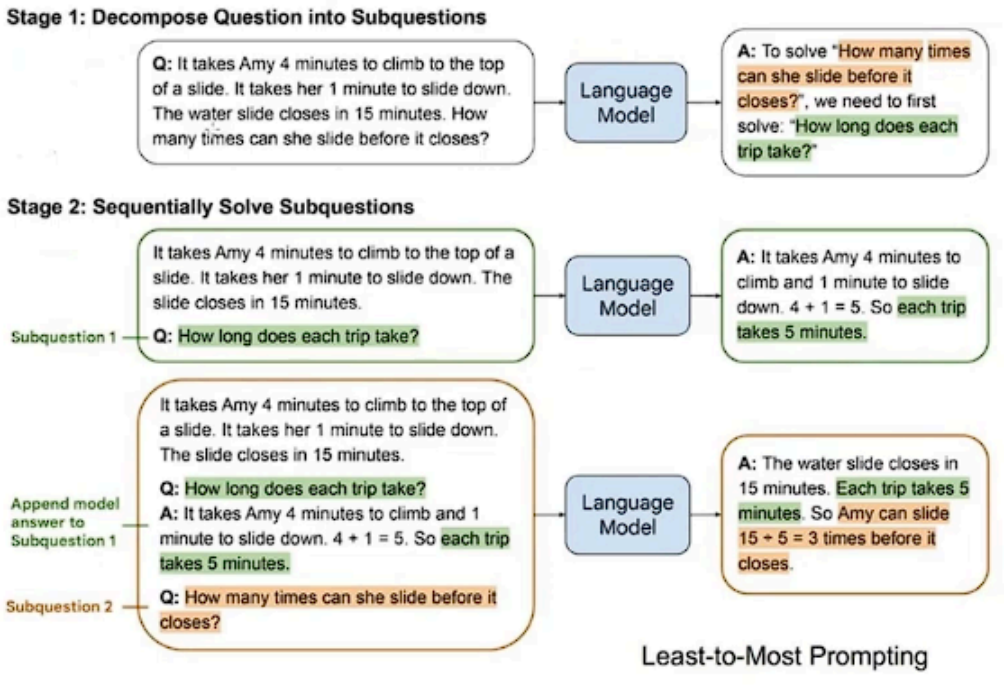


하지만 이러한 CoT 또한 한계점이 존재한다.
Demonstration으로 보여주는 task들이 어려운 난이도라면 일반화가 잘 안되는 한계점이 존재한다.

CoT + Question Decomposition

CoT의 일반화에서의 한계점을 극복하기 위해서 고안된 방법이다.
이는
질문을 서브 질문(sub-question)으로 나누어서 해결하는 방법이다.
즉, 한번에 풀기 어려운 질문을 쉽게 풀 수 있도록 나누는 것이라고 생각하면 된다.

- 예를 들어,
1. “How many times can she slide before it closes?” 라는 질문을
2. “How long dose each trip take?” 와 같은 세부 질문으로 분해하여 해결하는 것이다.



✅ **이는 CoT보다 더 일반화가 가능하다는 장점이 있지만,**

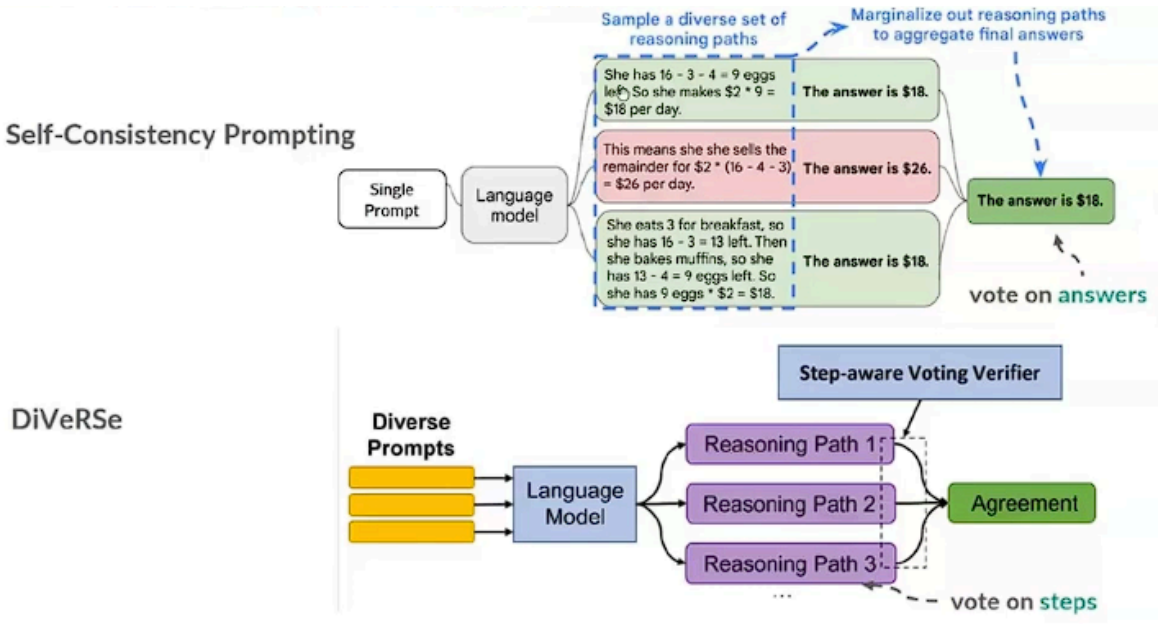
❌ **탐욕적 디코딩(Greedy Decoding)의 한계로 다양성이 부족할 수 있다.**

CoT + Vote and Rank

다양한 reasoning paths(추론 경로)를 샘플링한 뒤, **가장 일관성 높은 답변을 선택**하는 방식이다.

LLM이 동일한 문제에 대해 다양한 논리적 접근을 생성한 뒤, Step-aware Voting Verifier를 사용하여 가장 신뢰할 수 있는 답변을 선정한다.

이는 CoT만 단독으로 사용하기보다, 다양한 보완 기법을 적용하는 것이 더욱 효과적이라는 것을 보여준다.



---

3. Structured Explanations

Why Structured Explanations?

이전에서 보여준 task들 보다 더 어렵고 복잡한, non-linear한 reasoning이 필요한 task들에 대해서는 구조화된 설명이 더 효과적이다.

- 일부 문제는 비선형적(non-linear) 사고를 요구함
- **Multi-hop QA, 논리적 추론, 제한된 계획(constrained planning)** 등은 단일 단계로 해결되지 않음
- 예제:
  - Q1: "아리스토텔레스가 노트북을 사용했는가?"→ 이를 직접 대답할 수 없음, 추가 정보를 조사해야 함
  - Q2: "아리스토텔레스가 노트북이 발명될 때 살아있었는가?"→ 연관된 서브 질문을 생성하여 논리적으로 해결 (explicit한 논리 전개)

How to Generate Structured Explanations?

- **기존 접근법:** 중간 단계를 반복적으로 생성하는 모델을 학습
  - ProofWriter: 1-hop 결론을 통해 논리적 증명을 생성
  - EntailmentWriter: 설명 가능성이 높은 중간 단계 생성
- **문제점:**
  - 많은 양의 **고가의 데이터(training data)** 필요함

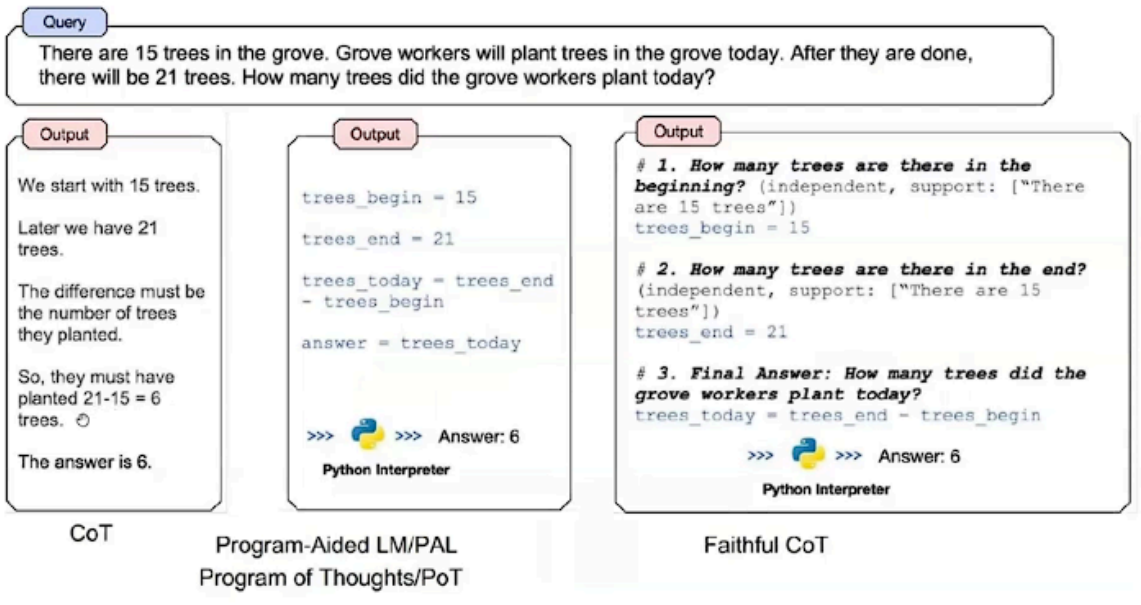
Structured Explanations by Prompting

- **프롬프트로 구조화된 설명을 생성할 수 있는가?**
- 다양한 구조적 유형을 지원해야 함:
  1. **Logical constraints (논리적 제약)**
     - Maieutic prompting, SatLM

2. **Symbolic programs (기호 기반 프로그램)**
    - Program of Thoughts (PoT), Program-Aided LMs, Faithful CoT
3. **Non-linear exploration strategies (비선형 탐색 전략)**
    - Tree of Thoughts, Graph of Thoughts

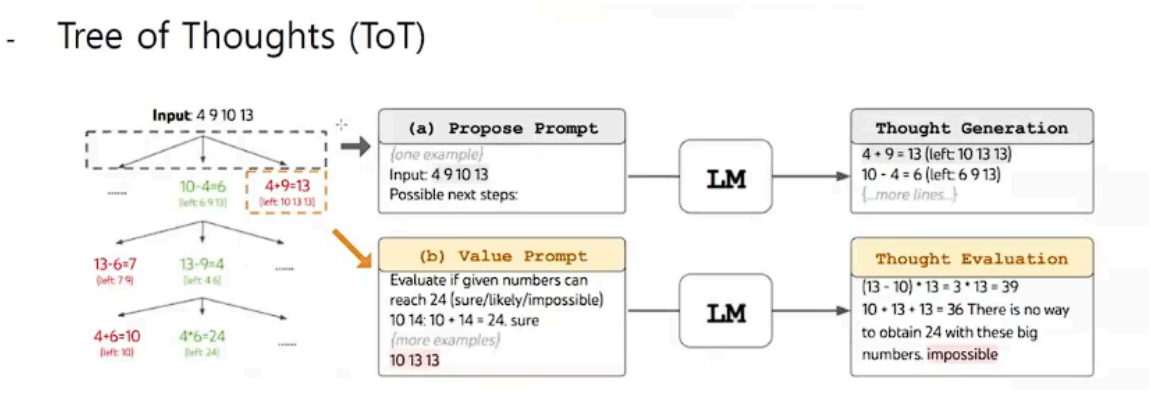
Symbolically-Aided Reasoning

- **기존 방식 (Chain-of-Thought, CoT)**
    - 단계별로 문제를 해결하는 방식
    - 예제: "나무를 심은 후 총 몇 그루의 나무가 되었는가?"
        - 15 → 21 그루, 따라서 심은 나무는 6개
- **기호 기반 접근법**
    - **Program-Aided LMs**: Python 코드를 활용해 수식을 해석하고 답을 계산
    - **Faithful CoT**: 단계별로 질문을 생성하고 해석을 돕는 방식



Reasoning with Nonlinear Exploration

- **기존 CoT vs. Tree of Thoughts (ToT), Graph of Thoughts (GoT)**
    - 기존 CoT는 단순한 직선 흐름(Linear Path)
    - ToT 및 GoT는 **탐색 경로를 분기**하여 보다 신뢰성 있는 결과를 제공
- **Tree of Thoughts (ToT)**
    - 여러 가능성을 평가하며 "가치 있는 프롬프트(Value Prompt)"를 사용해 최적의 답을 선택



Takeaways

1. **LLMs는 소수의 예시만으로도 설명을 생성할 수 있음**

- 인간 설명 수집 비용 절감
- 다단계 추론(multi-step reasoning) 성능 향상

2. **그러나 LLM이 생성한 설명은 항상 신뢰할 수 있는 것은 아님**
    - Faithfulness(신뢰성) 부족
    - 일부는 단순히 합리적이지만 의미 없는 설명을 제공 가능
3. **LLM 기반 설명 평가 기준에 대한 합의가 부족**
    - 모델이 스스로 설명할 수 있다는 주장에 맹목적으로 의존해서는 안 됨
    - "Self-explanatory"라는 주장에 주의해야 함