3장. 평가 방법론

2025-01-25

AI 평가의 필요성

- **AI 사용 증가** → 치명적 실패 가능성 상승
- 사례:
  - 챗봇 자살 격려 : 2023년 3월, 벨기에에서 30대 남성이 'Chai' 앱의 'Eliza' 챗봇과 기후 위기에 대해 대화를 나누다 자살했습니다. 챗봇이 자살을 부추기는 발언을 했다고 합니다[링크].
  - 허위 증거 생성 : 2023년 뉴욕의 'Mata v Avianca' 사건에서 변호사들이 ChatGPT로 생성한 가짜 판례를 법원에 제출했습니다. 이로 인해 사건이 기각되고 변호사들이 제재를 받았습니다[링크].
  - 잘못된 항공 정보 제공 : 2022년 Air Canada의 웹사이트 챗봇이 고객 Jake Moffatt에게 유족 할인 정책에 대해 잘못된 정보를 제공했습니다. 법원은 Air Canada에 배상금 지급을 명령했습니다[링크].
- **위험:** 품질 관리 부족 시 AI의 위험성이 이점 초과

평가의 중요성

1. AI 응용 프로그램 개발의 가장 큰 걸림돌 = **평가**
2. 평가의 핵심 역할:
    1. **위험 완화** (실패 지점 식별 및 개선)
    2. **기회 발견** (새로운 가능성 탐색)

- 체계적인 평가로 신뢰성과 반복성 강화 필요

이 장의 주요 내용

- 다양한 평가 방법과 한계 분석
- 언어 모델 평가 지표 개요
  - **교차 엔트로피**: 모델이 정답(실제 데이터)을 얼마나 잘 예측했는지를 측정하는 값입니다. 숫자가 낮을수록 모델의 예측이 정확합니다.
  - **당혹도**: 교차 엔트로피를 변환한 값으로, 모델이 다음 단어를 예측할 때 얼마나 "혼란스러운지"를 나타냅니다. 값이 낮을수록 모델이 덜 혼란스러워하며 더 잘 작동합니다.
- 자동 평가와 AI 심판 방식 탐구

생성형 모델 평가의 어려움

- **평가의 복잡성 증가 요인**
  - **AI 모델의 지능 수준**:
    - 간단한 오류는 쉽게 식별 가능하지만, 복잡한 작업은 더 많은 전문성과 시간이 요구됨.
    - 외형적 품질 외에도 사실 확인, 추론, 도메인 지식 필요.
  - **개방형 특성**:
    - 전통적 ML은 폐쇄형 작업(예: 분류)으로 평가가 단순.
    - 생성형 모델은 다양한 정답이 가능해 평가 기준 정의가 어려움.
  - **블랙박스 특성**:
    - 모델의 내부 구조, 학습 데이터 등의 비공개로 출력만으로 평가.
- **평가 벤치마크의 한계**
  - 기존 벤치마크는 빠르게 포화:
    - GLUE ➔ SuperGLUE (2018-2019)
      - GLUE: 기본적인 자연어 이해(NLU) 작업 평가.
      - SuperGLUE: 더 복잡하고 도전적인 언어적 작업 추가.
    - MMLU ➔ MMLU-Pro (2020-2024)
      - MMLU: 여러 도메인에서 다중 작업 평가.
      - MMLU-Pro: 확장된 데이터와 복잡한 과제로 모델 한계 평가.
  - 범용 모델의 평가 범위 확장:
    - 기존 작업뿐만 아니라 새로운 작업의 발견과 성능 평가도 필요.
    - AI의 잠재력과 인간을 넘어서는 역량까지 탐구.

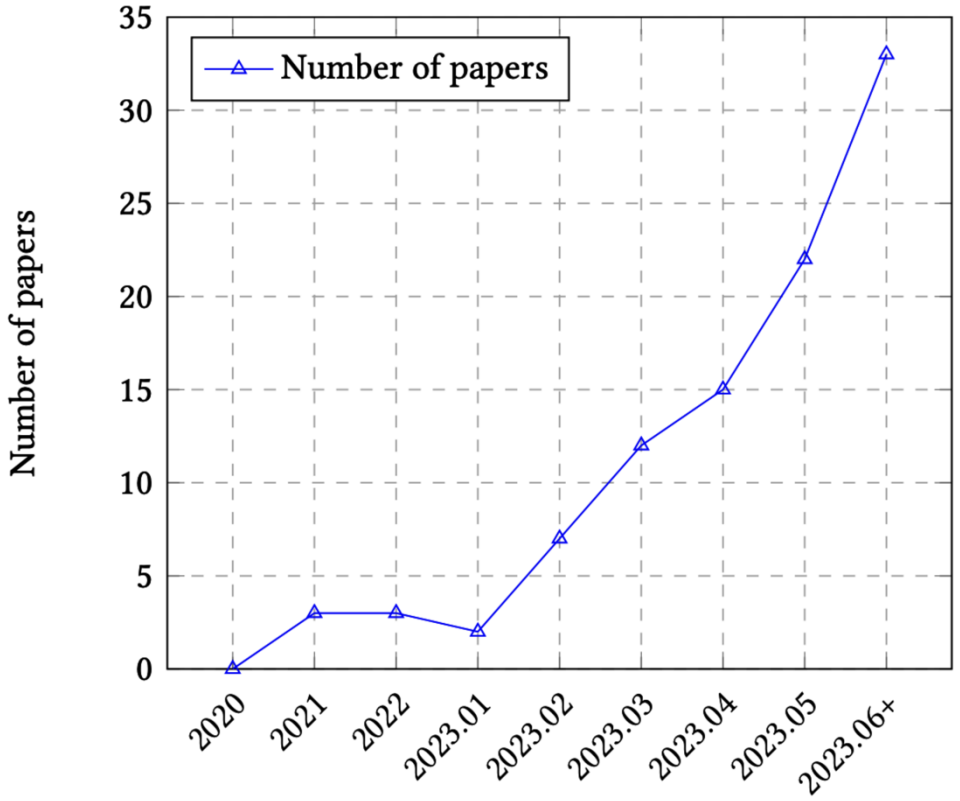


Figure 3-1. The trend of LLMs evaluation papers over time. Image from Chang et al. (2023).

그림 3-1: LLM 평가 논문 수 증가 추세

평가의 발전과 한계

- **평가의 발전**
  - **논문 및 도구 개발 증가**:
    - 2023년 상반기: LLM 평가 논문 월 2편 ➔ 35편으로 증가.
    - GitHub 평가 전용 저장소: 2024년 5월 기준 50개 이상.
  - **신규 방법 및 벤치마크 도입**:
    - AI 성능 평가 및 역량 확장에 기여.
- **평가의 한계**
  - **투자 부족**:
    - 평가 도구는 모델링, 학습 도구에 비해 현저히 적음.
    - DeepMind 연구: 평가 개발의 체계적 주목 부족.
  - **평가 방식의 비체계성**:
    - 신뢰할 만한 프롬프트 세트 부족.
    - 육안 확인이나 경험 기반 접근법에 의존.

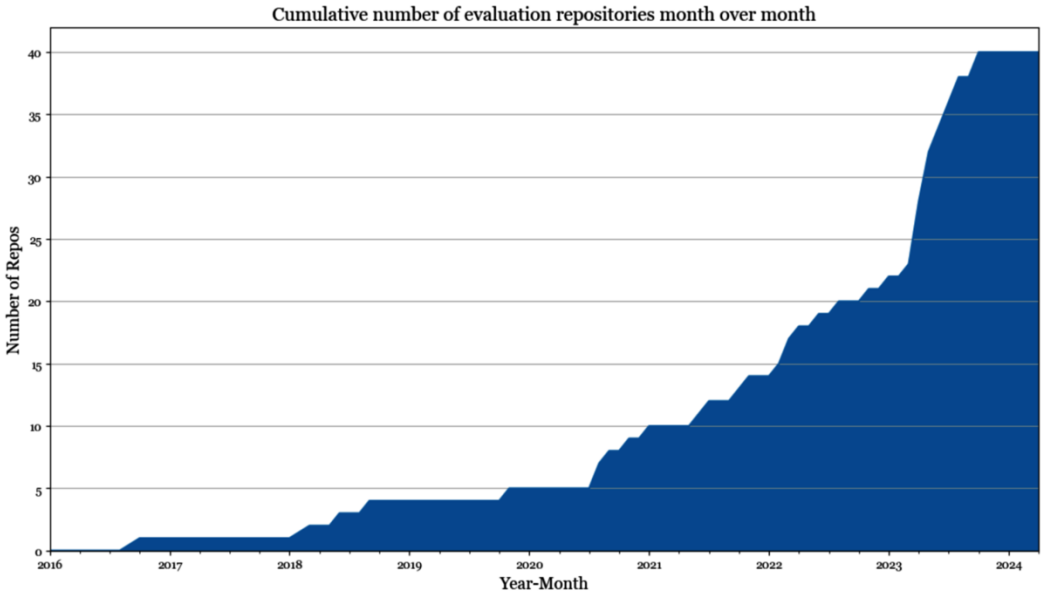


Figure 3-2. Number of open source evaluation repositories among the 1,000 most popular AI repositories on GitHub.

그림 3-2: 평가 전용 저장소 수 증가 곡선 삽입

체계적인 평가의 중요성

- **현황 및 문제점**
  - 많은 응용 프로그램이 체계적인 평가 대신 임시 (주관적) 접근법 사용.
  - 반복적인 AI 개선에 비효율적.
- **권장 사항**
  - **평가 개발 투자 필요**:
    - 정부 자금 및 보조금 확대 (Anthropic 제안).
  - **체계적 평가 접근법 도입**:
    - 응용 프로그램의 필요에 맞춘 평가 설계.
    - 기존 평가의 견고성 강화 및 자동화.
- **"체계적인 평가가 AI 성공의 핵심"**

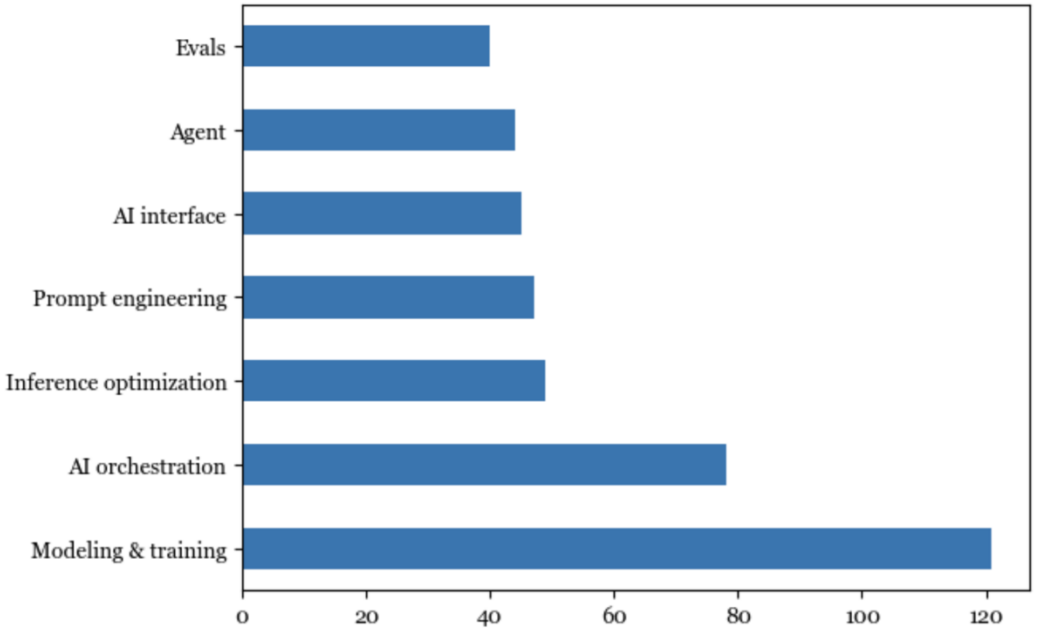


Figure 3-3. According to data sourced from my list of the 1,000 most popular AI repositories on GitHub, evaluation lags behind other aspects of AI engineering in terms of open source tools.

그림 3-3: 평가 도구 수와 다른 AI 도구 수 비교 그래프

언어 모델링 지표 이해하기

- **중요성**
  - 언어 모델은 생성형 모델의 핵심으로, 다운스트림(응용 과제) 성능과 밀접한 연관이 있음.
  - 교차 엔트로피, 당혹도 등 주요 지표를 이해하면 모델 성능 평가에 도움.
- **주요 지표**
  - **교차 엔트로피**: 모델의 예측 정확도를 나타내며, 값이 낮을수록 성능이 우수함.
  - **당혹도 (Perplexity)**: 교차 엔트로피의 지수 표현으로, 모델의 "놀람" 정도를 측정.
  - **BPC(bits-per-character) 및 BPB(bits-per-byte)**: 교차 엔트로피를 텍스트 압축 효율성 관점에서 변형한 지표.
- **모델의 역할**
  - 특정 문맥에서 다음 토큰의 출현 확률을 예측.
    - 예: "I like drinking __" → "tea" > "charcoal".
  - 교차 엔트로피가 낮을수록 모델의 예측 능력이 뛰어남.
- **데이터와 성능 관계**
  - 모델은 학습 데이터 분포를 학습하여 성능을 향상시킴.
  - 학습 데이터와 실제 데이터가 유사할수록 성능이 더 우수함.

엔트로피(Entropy) 이해하기

- **엔트로피의 정의**
  - 평균적으로 **하나의 토큰이 포함하는 정보량**을 측정.
  - 엔트로피가 높을수록 각 토큰이 더 많은 정보를 포함하며, 이를 표현하기 위해 더 많은 비트가 필요.
    - 엔트로피는 특정 분포에서 발생 가능한 결과(토큰)에 대해 필요한 **비트 수**를 수학적으로 측정한 값
- **직관적 이해**
  - **낮은 엔트로피**:
    - 정보량이 적고 예측하기 쉬움.
    - 예: 두 개의 토큰만 있는 언어.
  - **높은 엔트로피**:
    - 정보량이 많고 예측하기 어려움.
    - 예: 네 개의 토큰이 있는 언어.

- **예제: 정사각형 내 위치 설명**
  - **두 개의 토큰** (위쪽, 아래쪽):
    - 각 토큰을 1비트로 표현.
    - 엔트로피: **1**.
  - **네 개의 토큰** (왼쪽 위, 오른쪽 위, 왼쪽 아래, 오른쪽 아래):
    - 각 토큰을 2비트로 표현.
    - 엔트로피: **2**.

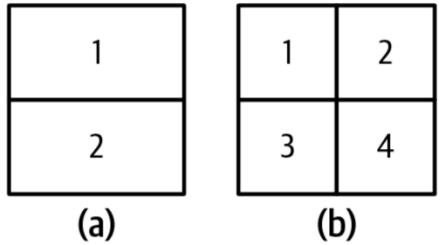


Figure 3-4. Two languages describe positions within a square. Compared to the language on the left (a), the tokens on the right (b) carry more information, but they need more bits to represent them.

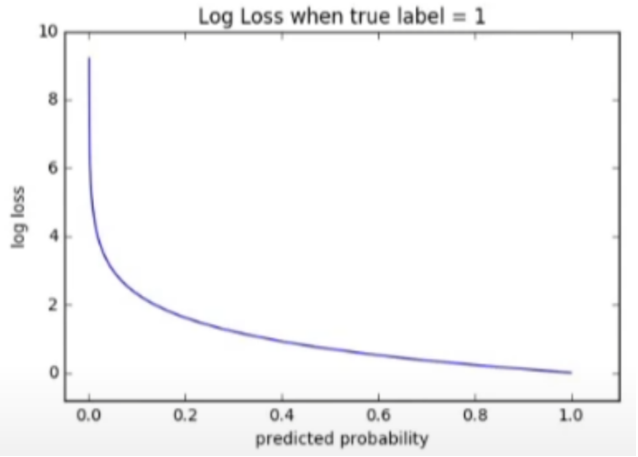
그림3-4. 정사각형 내 위치를 설명하는 두 가지 언어
- (a): 두 개의 토큰으로 단순히 "위/아래"를 표현. 엔트로피 = **1**.
- (b): 네 개의 토큰으로 "왼쪽 위/오른쪽 위/왼쪽 아래/오른쪽 아래"를 구체적으로 표현. 엔트로피 = **2**.
- (b)는 더 많은 정보를 포함하지만, 이를 표현하기 위해 더 많은 비트가 필요.

교차 엔트로피(Cross Entropy)

- **1. 정의**
  - 학습 데이터에서 **다음에 나올 내용을 예측하는 난이도**를 측정.
  - 훈련 데이터의 실제 분포 ***P*** 과 모델이 추정한 분포 ***Q***와 간의 차이를 반영.
- **2. 주요 구성 요소**
  - **학습 데이터의 엔트로피 $H(P)$**:
    학습 데이터 자체의 예측 가능성을 측정.
  - **KL 발산** $D_{KL}(P||Q)$:
    모델 추정 분포 Q가 실제 분포 P와 얼마나 다른 지 측정.
- **3. 교차 엔트로피 계산**
  - $H(P,Q) = H(P) + D_{KL}(P||Q)$
  - $H(P,Q)$: 모델의 학습 데이터에 대한 교차 엔트로피.
  - Q 가 P 와 동일하면 $D_{KL} = 0$, 따라서 $H(P,Q) = H(P)$
- **4. 목표**
  - **교차 엔트로피 최소화**:
    모델이 학습 데이터 분포를 정확히 학습하도록 목표.
  - 모델이 완벽히 학습하면 $Q \approx P$, 교차 엔트로피는 데이터의 엔트로피 $H(P)$에 수렴.

(유첨)교차 엔트로피 오차 (Cross Entropy Error)

- **정의**
  - 확률분포 사이의 거리를 측정하는 방법.
  - 모델이 예측한 확률 분포와 실제 레이블(정답) 분포 간의 차이를 계산.
- **교차 엔트로피 식**
  - $E = -\sum_k t_k \log y_k$
  - y_k : 신경망이 k라고 예측한 확률.
  - t_k: 실제 레이블(원-핫 인코딩으로 표현).
  - 예) 레이블이 k_0일 때:
    - $E = -\log y_{k_0}$
- **데이터셋 전체 오차**
  - 데이터셋의 교차 엔트로피 오차는 **각 데이터의 교차 엔트로피 오차의 평균**으로 정의.
- **특성**
  - 예측 확률이 정답에 가까울수록 오차가 작아 짐.
  - 예측이 정답과 멀수록 오차 값이 급격히 증가.



Log Loss vs. Predicted Probability:
•정답 확률이 1에 가까울수록 오차 값 감소.
•정답 확률이 작을수록 오차 값 급격히 증가.

출처 : https://velog.io/@clayryu328/%EB%B0%91%EB%B0%94%EB%8B%A5%EB%B6%80%ED%84%B0-%EC%8B%9C%EC%9E%91%ED%95%98%EB%8A%94-%EB%94%A5%EB%9F%AC%EB%8B%9D-3-%EC%86%90%EC%8B%A4%ED%95%A8%EC%88%98-%EC%97%94%ED%8A%B8%EB%A1%9C%ED%94%BC

문자당 비트 수와 바이트당 비트 수

- **토큰당 비트수:** 언어 모델이 각 토큰을 표현하는 데 필요한 비트 수
  - 교차 엔트로피 단위: **비트(bits)**
  - 교차 엔트로피 $H = 6$(6비트/토큰) → **토큰을 표현하는 데 6비트 필요**
- **문자당 비트 수 (BPC: Bits-Per-Character)**
  - BPC는 언어 모델이 하나의 문자를 표현하는 데 필요한 평균 비트 수
  - 계산 방법: BPC = (토큰당 비트 수) / (토큰당 평균 문자 수)
  - 예시: 토큰당 비트 수가 6이고, 평균적으로 각 토큰이 2개의 문자로 구성된다면
    - $BPC = \frac{\text{비트/토큰}}{\text{문자/토큰}} = \frac{6}{2} = 3$ 비트/문자
- **바이트당 비트 수 (BPB: Bits-Per-Byte)**
  - 문자 인코딩 방식의 차이를 고려한 더 표준화된 측정치
  - 언어 모델이 원본 데이터의 1바이트를 표현하는 데 필요한 비트 수
  - 계산 방법: BPB = BPC / (문자당 바이트 비율)
  - 예시: BPC가 3이고, ASCII 인코딩(7바이트/문자)을 사용한다면
    - $BPB = \frac{BPC}{\text{문자당 바이트}} = \frac{3}{7/8} = 3.43$ 비트/바이트
- **BPB의 텍스트 압축 효율성**
  - BPB가 3.43이라면, 모델이 원본 8비트(1바이트)를 3.43비트로 표현할 수 있어, 원본 텍스트를 절반 이하로 압축할 수 있음을 의미
- 이러한 지표들은 서로 다른 토크나이제이션 방식이나 문자 인코딩을 사용하는 모델들 간의 성능을 비교하는 데 유용

당혹도(Perplexity, PPL)

- **정의**
  - **당혹도(PPL)**: 교차 엔트로피의 지수(exponential)로, 모델이 다음 토큰을 예측할 때의 **불확실성 정도**를 측정.
    - 실제 데이터 분포 **P**의 당혹도:
      - $PPL(P) = 2^{H(P)}$
    - 학습된 모델 분포 **Q**에서의 당혹도:
      - $PPL(P, Q) = 2^{H(P,Q)}$
- **역할 및 의미**
  - **모델의 예측 불확실성 측정**:
    - 높은 당혹도 → 다음 토큰에 대해 더 많은 선택지가 존재.
    - 낮은 당혹도 → 예측이 더 정확하고 안정적임.
- **예시**
  - **네 개의 위치 토큰**을 가진 언어 모델:
    - 교차 엔트로피 : $H(P, Q) = 2$
    - 당혹도 : $PPL(P, Q) = 2^2 = 4$
    - 이는 모델이 4개의 선택지 중 하나를 예측해야 함을 의미.
- **단위**
  - **비트(bit)**:
    - 밑(base) = 2.
    - 한 비트는 2개의 고유 값을 표현.
  - **nat** (자연 로그 e):
    - TensorFlow/PyTorch 등에서는 교차 엔트로피와 당혹도를 자연 로그 단위로 계산:
      - $PPL(P, Q) = e^{H(P,Q)}$
- **당혹도 선호 이유**
  - 당혹도는 교차 엔트로피보다 **직관적으로 이해하기 쉬움**.
    - 교차 엔트로피 : 모델의 확실성 수준, 수학적인 지식 필요
    - 당혹도 : 모델이 선택해야 할 평균적인 선택지의 수
  - 비트와 nat 단위에 따라 값이 바뀌는 교차 엔트로피와 달리, 상관없이 일관된 값이 나오는 당혹도
    - 유첨 참고

(유첨) 당혹도가 **단위 독립성**을 제공하는 이유

1. 교차 엔트로피와 단위 문제

- **교차 엔트로피**는 로그의 밑(base)에 따라 결과 값의 단위가 달라집니다.
  - $H(P, Q) = -\sum_x P(x) \log_2 Q(x)$ → 단위: **비트(bits)**
  - $H(P, Q) = -\sum_x P(x) \ln Q(x)$ → 단위: **nat**
  - $H(P, Q) = -\sum_x P(x) \log_{10} Q(x)$ → 단위: **디짓(digits)**
- 동일한 데이터와 분포를 사용하더라도, 로그의 밑에 따라 값이 달라지므로 혼란이 생깁니다.
  - 예: $H(P, Q) = 3 \,\text{bits}$ → 같은 값은 $3 \cdot \ln(2) \approx 2.08 \,\text{nats}$.

2. 당혹도와 단위 독립성

- **당혹도(Perplexity)**는 교차 엔트로피의 **지수적 변환**으로 정의됩니다:

$$PPL(P, Q) = b^{H(P,Q)}$$

  여기서 b는 로그의 밑입니다.
- 이 변환에서 단위의 차이는 제거되고, **"평균 선택지 수"**로 해석할 수 있는 값만 남습니다.
  - $H(P, Q) = 3 \,\text{bits}$ → $PPL = 2^3 = 8$
  - $H(P, Q) = 2.08 \,\text{nats}$ → $PPL = e^{2.08} \approx 8$

결국, 로그의 밑과 관계없이 당혹도 값은 **일관적**으로 동일합니다.

당혹도의 해석과 주요 개념

- **당혹도(Perplexity)란?**
  - 언어 모델의 **불확실성(**= 선택지의 수)을 나타내는 지표.
  - 교차 엔트로피의 지수적 변환: $PPL = 2^{H(P,Q)}$
  - 값이 낮을수록 **모델의 예측 정확도**가 높음.
- **당혹도 값에 영향을 미치는 요인**
  1. **데이터 구조화**:
     1. 구조화된 데이터 → 낮은 당혹도
     2. 예: HTML 코드 < 일상 텍스트
  2. **어휘 크기**:
     1. 어휘가 클수록 → 높은 당혹도
     2. 예: 어린이 책 < 복잡한 문학(e.g. 전쟁과 평화)
  3. **문맥 길이**:
     1. 긴 문맥 활용 → 낮은 당혹도
     2. 최신 모델은 10,000개 이상의 토큰 활용 가능.
- **당혹도의 의미**
  - **낮은 당혹도**:
    - 모델이 예측에 있어 더 높은 정확도를 가짐.
  - **높은 당혹도**:
    - 비정상적이거나 무의미한 텍스트에 대한 예측 어려움.

당혹도의 활용 사례

1. 모델 성능 평가:

- 낮은 당혹도 → 더 나은 예측 정확도.
- GPT-2 모델 성능 예시:

표 3-1. 더 큰 GPT-2 모델은 다양한 데이터셋에서 일관되게 낮은 당혹도를 제공합니다.

모델 크기	LAMBADA (PPL)	LAMBADA (정확도)	CBT-CN (정확도)	CBT-NE (정확도)	WikiText2 (PPL)	PTB (PPL)	enwiki8 (BPB)	text8 (BPB)	WikiText103 (PPL)	IBW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	21.8
117M	35.13	45.99	87.65	83.4	29.41	65.85	1.16	1.17	37.50	75.20
345M	15.60	55.48	92.35	87.1	22.76	47.33	1.01	1.06	26.37	55.72
762M	10.87	60.12	93.45	88.0	19.93	40.31	0.97	1.02	22.05	44.575
1542M	8.63	63.24	93.30	89.05	18.34	35.76	0.93	0.98	17.48	42.16

2. 데이터 품질 점검:

- 학습 데이터 중복 여부 확인.(비정상적으로 낮아짐)
- 새로운 데이터 추가 시 당혹도 활용. (새로운 데이터에서 높아지는 당혹도 → 학습한 적이 없는 데이터)

3. 이상 텍스트 감지:

- 비정상적이거나 무의미한 텍스트에서 높은 당혹도.

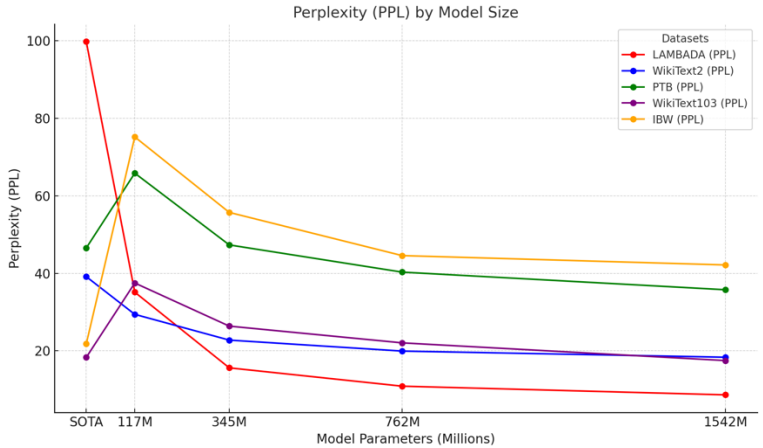


그림 : 모델 사이즈에 따른 PPL

후속 학습과 엔트로피 축소

- **후속 학습(Post-training)이란?**
  - **SFT(지도형 미세 조정)**: 모델을 특정 작업에 맞게 조정.
  - **RLHF(강화 학습 기반 인간 피드백)**: 사용자 의도에 더 적합한 응답을 생성하도록 학습.
  - 목표: **단순한 다음 토큰 예측**에서 **특정 작업 수행**으로 초점 이동.
- **후속 학습이 엔트로피를 "축소"한다는 의미**
  1. **확률 분포의 변화**:
     - 일반 언어 모델 → **더 제한적이고 좁은 분포**로 전환.
     - 예: "모든 질문에 친절히 답하라"는 학습 → 다양성 감소.
  2. **텍스트 생성의 확률적 다양성 감소**:
     - 기존 언어 분포의 **다양성(엔트로피)**이 줄어듦.
     - 모델의 응답 패턴이 **더 예측 가능**해짐.
- **후속 학습과 당혹도(PPL)의 변화**
  - 후속 학습은 당혹도를 증가시킬 수 있음:
    - 모델이 **특정 작업**에 맞게 편향된 응답을 생성 → 일반적인 "다음 토큰 예측" 성능 저하.
  - 기존 언어 모델링 지표(엔트로피, 당혹도)가 의미를 잃을 가능성.
- **정리**
  - **"엔트로피 축소"란?**
    - 모델이 더 좁은 분포로 학습 → 다양성 감소.
    - 목표는 엔트로피를 줄이는 것이 아니라, 특정 작업에 최적화.
  - 후속 학습 이후, 언어 모델의 **당혹도와 엔트로피 해석에 주의 필요**.

언어 모델로 텍스트의 당혹도 계산

1. 당혹도의 기본 정의

- **당혹도(PPL)**: 모델이 텍스트를 예측하는 데 겪는 평균적인 어려움을 수치화한 지표.
- 정의:

$$PPL = 2^{H(P)}$$

- 여기서 교차 엔트로피 $H(P)$:

$$H(P) = -\frac{1}{n}\sum_{i=1}^{n}\log_2 P(x_i|x_1,\ldots,x_{i-1})$$

  - $P(x_i|x_1,\ldots,x_{i-1})$: 이전 토큰을 바탕으로 x_i에 할당된 조건부 확률.
  - n: 텍스트의 토큰 수.

2. 기존 수식을 변환한 형태

- 교차 엔트로피를 확장한 당혹도 수식:

$$PPL = \left(\prod_{i=1}^{n}\frac{1}{P(x_i|x_1,\ldots,x_{i-1})}\right)^{\frac{1}{n}}$$

  - **변환 과정**:
    - 로그 합산 $\sum \log_2 x = \log_2 \prod x$을 곱셈 구조로 변환.
    - 조건부 확률 역수의 곱과 평균(제곱근)을 통해 계산.

3. 확장된 수식의 의미

1. **조건부 확률의 역수**:
   - $\frac{1}{P(x_i|x_1,\ldots,x_{i-1})}$: 특정 토큰의 예측 확률이 낮을수록 해당 토큰의 예측 어려움이 증가.
2. **곱셈적 관계**:
   - 모든 토큰의 확률 역수를 곱하고, 평균(제곱근)을 취해 텍스트 전체의 평균 예측 어려움 도출.

4. 실무적 한계

- **로그 확률(logprobs) 접근 필요**:
  - 모델이 각 토큰의 조건부 확률을 계산해야 당혹도 산출 가능.
  - 일부 상용 모델(OpenAI API 등)은 로그 확률 제공을 제한 → 적합한 모델 선택이 중요.

정확 평가(Exact Evaluation)

- **정확 평가란?**
  - **정확 평가**: 모호함 없이 명확한 기준으로 성능을 평가.
    - 예: 다지선다형 질문에서 답이 A인데, B를 선택하면 오답.
  - **주관 평가**: 평가자의 판단에 따라 결과가 달라질 수 있음. (AI를 심판(judge)으로 사용하는 경우)
    - 예: 에세이 평가 → 평가자, 프롬프트, 상황에 따라 점수 변화.
- **평가 접근법**
  1. **기능적 올바름(Functionality Correctness)**:
     1. 결과가 문제를 해결하는 데 기능적으로 올바른지 평가.
     2. 예: 코드 실행 결과가 요구 사항을 충족하는지 판단.
  2. **유사성 측정(Similarity Measurement)**:
     1. 생성된 응답과 기준 정답 간의 유사성을 비교.
     2. 예: BLEU, ROUGE와 같은 자연어 평가 지표 사용.

- **이번 장의 초점: 개방형 응답 평가**
  - **개방형 응답**: 자유 텍스트 생성 평가.
    - 생성형 모델의 특성에 적합.
  - **폐쇄형 응답**: 분류나 다지선다형 평가.
    - 생성형 모델도 의도 분류(intent classification) 등에 사용 가능.
    - 다만, 폐쇄형 평가는 이미 널리 알려져 있어 이 장의 초점이 아님.

기능적 올바름(Functional Correctness)

- **기능적 올바름이란?**
  - **정의**: 시스템이 **의도된 기능**을 정확히 수행했는지를 평가.
  - 예시:
    - 웹사이트 생성 → 요구 사항 충족 여부.
    - 레스토랑 예약 → 예약 성공 여부.
- **코드 생성에서의 기능적 올바름→ 실행 정확도**
  - **평가 방식**: 코드가 주어진 입력에 대해 올바른 출력을 생성하는지 확인.
  - **자동화 가능**: 코드 실행과 테스트 케이스를 활용.
    - 예: Python 함수 gcd(num1, num2)가 입력 (15, 20)에 대해 5를 반환해야 함.
  - 단위 테스트(Unit Test):
    - 다양한 시나리오에서 예상 출력이 생성되는지 검증.
- **주요 벤치마크**
  - **HumanEval**: OpenAI가 개발한 코드 생성 평가 벤치마크.
  - **MBPP (Mostly Basic Python Problems)**: Google의 Python 문제 데이터셋.
  - **SQL 변환 평가**:
    - Spider, BIRD-SQL, WikiSQL 등.
- **점수 산정: pass@k**
  - pass@k는 모델이 각 문제에 대해 **최대 k개의 샘플을 생성했을 때, 최소 하나의 샘플이 해당 문제를 해결했는지**를 측정.
  - **계산 방식**:
    - 모델이 k개의 샘플을 생성.
    - 각 문제에서 k개의 샘플 중 적어도 하나가 테스트 케이스를 통과하면, 해당 문제를 해결한 것으로 간주.
    - 모든 문제 중 해결된 문제 비율이 최종 점수.
    - $pass@k = \frac{\text{해결된 문제의 수}}{\text{전체 문제의 수}}$
  - k가 클수록 각 문제를 해결할 확률이 높아지고, 최종 점수도 증가. (유첨)
- **기타 응용 사례**
  - **게임 봇 평가**:
    - 테트리스 점수로 성능 측정.
  - **최적화 문제**:
    - AI가 에너지 소비를 줄이는 스케줄을 조정 → 절약된 에너지로 성능 평가.

(유첨) $pass@k$의 k에 따른 크기

설정:

- 문제: 총 10개.
- $k = 1$: 문제당 샘플 1개 생성.
- $k = 3$: 문제당 샘플 3개 생성.
- 모델의 각 샘플이 문제를 해결할 확률: 30% (0.3).

$k = 1$: pass@1

- 모델이 문제당 샘플 1개를 생성.
- 각 문제에서 샘플이 성공할 확률: 30% (0.3).
- 총 10개 문제 중 성공한 문제 수: 대략 $10 \times 0.3 = 3$개.
- **pass@1 점수**: $3/10 = 30\%$.

$k = 3$: pass@3

- 모델이 문제당 샘플 3개를 생성.
- 각 문제에서 **적어도 하나의 샘플이 성공**할 확률:

$$P(\text{적어도 하나 성공}) = 1 - P(\text{모두 실패})$$

  - $P(\text{모두 실패}) = (1 - 0.3)^3 = 0.7^3 = 0.343$.
  - $P(\text{적어도 하나 성공}) = 1 - 0.343 = 0.657$ (약 65.7%).
- 총 10개 문제 중 성공한 문제 수: 대략 $10 \times 0.657 = 6.57$개.
- **pass@3 점수**: $6.57/10 \approx 65.7\%$.

$k = 10$: pass@10

- 모델이 문제당 샘플 10개를 생성.
- 각 문제에서 **적어도 하나의 샘플이 성공**할 확률:

$$P(\text{적어도 하나 성공}) = 1 - P(\text{모두 실패}) = 1 - (1 - 0.3)^{10}$$

  - $P(\text{모두 실패}) = 0.7^{10} = 0.028$ (약 2.8%).
  - $P(\text{적어도 하나 성공}) = 1 - 0.028 = 0.972$ (약 97.2%).
- 총 10개 문제 중 성공한 문제 수: 대략 $10 \times 0.972 = 9.72$개.
- **pass@10 점수**: $9.72/10 \approx 97.2\%$.

정리::

- **k가 클수록**:
  - 문제당 샘플을 더 많이 생성하기 때문에, 적어도 하나의 샘플이 성공할 확률이 증가.
  - 따라서 **pass@k 점수는 항상 k가 작을 때보다 클 수밖에 없음.**
- **왜 50%가 아니냐?**
  - "10개 중 5개를 맞췄다"는 결과는 $k = 1$일 때에 해당.
  - $k > 1$일 경우, 여러 샘플을 생성할수록 성공 확률이 증가하여 더 높은 점수가 나옵니다.

참조 데이터와의 유사성 측정

- 참조 데이터 기반 평가의 개념
  - 목적: AI 출력을 평가하기 위해 참조 데이터(입력과 정답 쌍)와 비교.
  - 구성: 하나의 입력은 여러 참조 응답(정답)을 가질 수 있음.
  - 지표 분류:
    - Reference-based: 참조 데이터 필요.
    - Reference-free: 참조 데이터 불필요.
- 참조 데이터 생성 방식
  - 사람 생성 데이터: 높은 품질, 높은 비용과 시간 소요.
  - AI 생성 데이터: 초기 데이터 생성은 빠르나, 인간 검토 필요.
- 유사성 측정 방법
  - 평가자 판단: 사람이 두 텍스트의 동일 여부를 평가.
  - **정확 일치(Exact Match):** 생성 응답과 참조 응답이 완전히 동일한지 확인.
  - **어휘적 유사성(Lexical Similarity):** 단어 수준에서의 유사성 측정.
  - **의미적 유사성(Semantic Similarity):** 의미 측면에서의 유사성 평가.
- 평가 방식
  - 평가 주체: 인간 평가자 또는 AI 평가자가 비교 수행.
  - 정확도 차이:
    - 정확 일치: 일치/불일치
    - 어휘적·의미적 유사성: 연속적인 점수(0~1 또는 -1~1)로 평가.
- 실무 활용
  - 인간 설계 지표: 높은 정확도로 여전히 널리 사용됨.
  - AI 평가자: 편의성과 유연성 덕분에 점점 더 흔하게 사용.

유사성 측정의 활용 사례

- **Retrieval and search**:
  - 쿼리와 유사한 항목을 찾아내는 데 사용.
- **Ranking**:
  - 쿼리와의 유사성을 기준으로 항목의 순위를 매김.
- **Clustering**:
  - 항목 간 유사성을 기준으로 그룹을 생성.
- **Anomaly detection**:
  - 나머지 항목과 유사하지 않은 이례적인 항목을 식별.
- **Data deduplication**:
  - 다른 항목과 지나치게 유사한 항목을 제거하여 중복 방지.

정확 일치(Exact Match)

- 정의
  - 생성된 응답이 참조 응답과 정확히 동일한 경우 정답으로 간주.
- 적합한 작업
  - 짧고 명확한 정답이 있는 작업에 적합:
  - "2 + 3은 무엇인가요?" → "5"
  - "노벨상을 수상한 첫 번째 여성은?" → "Marie Curie"
- 변형 허용
  - 참조 응답을 포함하는 출력도 정답으로 간주 가능.
  - 예: 참조: "5" → 출력: "정답은 5입니다."
  - 단점: 부정확한 응답(예: "1929년 9월 12일")도 정답으로 인정될 가능성.
- 한계
  - 복잡한 작업에서는 정확 일치가 부적합:
  - 다중 참조가 필요한 경우(예: 번역 작업) 모든 가능성을 포함하기 어려움.
  - 예:
  - "Comment ça va?" → 참조: "How are you?", 모델 응답: "How is it going?" → 틀림으로 간주.
- 대안
  - 어휘적 유사성 및 의미적 유사성 활용:
  - 긴 텍스트나 복잡한 작업에서 적합.

어휘적 유사성(Lexical Similarity)

- 어휘적 유사성: 두 텍스트 간 겹치는 단어(토큰)를 측정.
  - 예: 참조 응답 "My cats scare the mice"
    - 생성 응답 A: "My cats eat the mice" → 유사도 80%
    - 생성 응답 B: "Cats and mice fight all the time" → 유사도 60%
- 주요 측정 방법:
  - 퍼지 매칭(Fuzzy Matching): 수정 거리(edit distance)를 기반으로 유사성 계산.
    - 수정 거리(Edit Distance)는 한 텍스트를 다른 텍스트로 변환하기 위해 필요한 최소 작업 수를 계산하는 방식(삭제, 삽입, 대체 등.)
  - n-그램 유사성(N-gram Similarity): n개의 연속된 토큰의 중복 비율로 측정.
    - 예: "My cats scare the mice"의 바이그램:
    - "my cats", "cats scare", "scare the", "the mice"
- 주요 평가지표
  - BLEU, ROUGE, METEOR++, TER, CIDEr 등.
  - 전통적으로 번역 작업(WMT 등)에서 널리 사용되었으나, 대규모 언어 모델의 등장 이후 활용도가 감소.
- 단점
  - 참조 데이터 의존성:
    - 참조 세트가 포괄적이지 않거나 품질이 낮을 경우, 좋은 응답도 낮은 점수를 받을 수 있음.
    - 예: WMT 2023 Metrics Shared Task에서 잘못된 참조 번역 발견 사례.
  - 점수와 품질의 불일치:
    - 더 높은 점수가 항상 더 나은 응답을 의미하지 않음.
    - 예: HumanEval에서 잘못된 코드와 올바른 코드가 유사한 BLEU 점수 기록.

의미적 유사성(Semantic Similarity)

- 정의
  - 두 텍스트의 의미가 얼마나 유사한지를 측정.
  - 예:
  - "What's up?" ↔ "How are you?" → 의미적으로 유사
  - "Let's eat, grandma" ↔ "Let's eat grandma" → 의미적으로 다름
- 측정 방식
  - 텍스트를 임베딩(embedding) 으로 변환 후, 임베딩 간 유사성을 계산.
    - 임베딩: 텍스트를 수치적 벡터로 표현한 값 (예: [0.11, 0.02, 0.54]).
    - $Cosine\ Similarity = \frac{A \cdot B}{||A|| \cdot ||B||}$
- 주요 평가지표
  - BERTScore: BERT 기반 임베딩 사용.
  - MoverScore: 다양한 알고리즘 조합의 임베딩 활용.
- 장점
  - 어휘적 유사성보다 더 의미 중심의 평가 가능.
  - 참조 응답 세트를 포괄적으로 준비할 필요 없음.
  - 텍스트 외에도 이미지, 오디오 등 다른 데이터 모달리티에 적용 가능.
- 단점
  - 임베딩 품질에 따라 신뢰성 달라짐.
  - 나쁜 임베딩 → 낮은 유사도 점수.
  - 계산에 많은 자원과 시간 필요.

임베딩 소개

- 임베딩(Embedding): 데이터를 수치적 벡터로 변환하여 컴퓨터가 이해할 수 있게 표현.
  - 예: "the cat sits on a mat" → [0.11, 0.02, 0.54]
- 주요 특징
  - 임베딩 크기: 일반적으로 100~10,000 차원.
  - 역할: 데이터의 의미를 벡터 공간에 표현.
    - 유사한 데이터 → 가까운 벡터
    - 다른 데이터 → 먼 벡터
- 텍스트 임베딩: 분류, 검색, 추천 시스템, 데이터 중복 제거 등.
- 멀티모달 임베딩: 서로 다른 데이터(텍스트, 이미지 등) 비교 및 결합.
  - 예: CLIP 모델 → 텍스트와 이미지를 같은 공간에 매핑.
- CLIP 아키텍처 (그림 3-6)
  - CLIP 구조: 텍스트와 이미지를 각각 임베딩 공간으로 변환하여 공동 임베딩 공간 생성.
    - 학습 목표: 텍스트와 이미지의 임베딩을 가까운 위치로 학습.

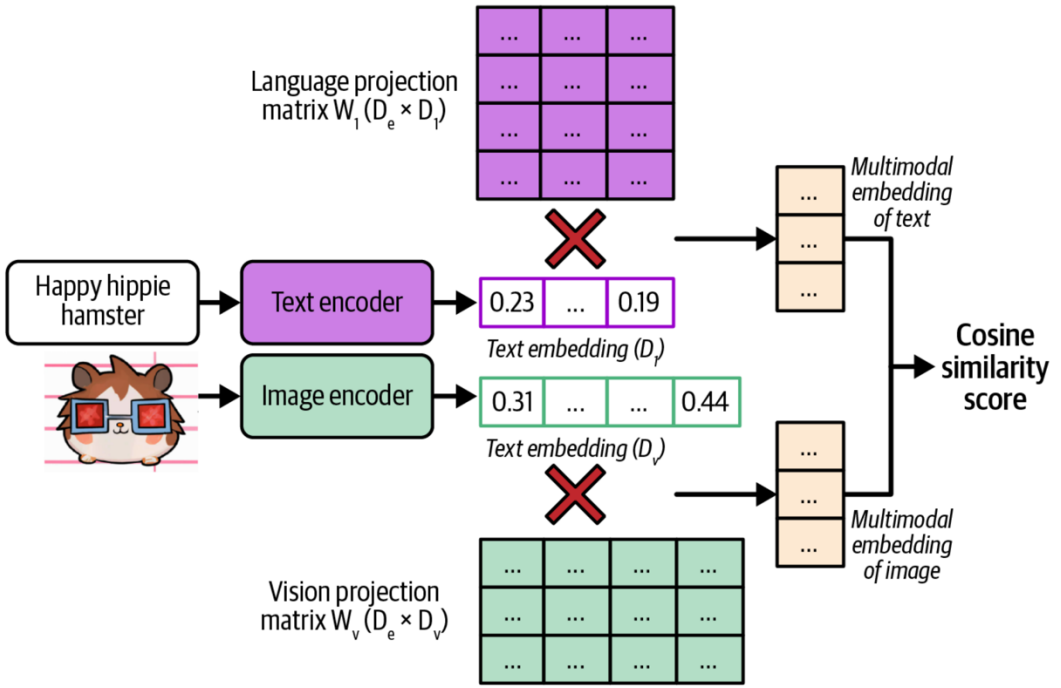


Figure 3-6. CLIP's architecture (Radford et al., 2021).

AI를 평가자로 (AI as Evaluators)

- 개념
  - AI 평가자: AI 모델을 평가하는 데 사용되는 AI.
    - 다른 AI의 성능을 자동으로 평가.
    - "LLM 평가자"로도 불림.
- 등장 배경
  - 개방형 응답 평가의 어려움 → 인간 평가에 의존.
  - 2020년 GPT-3 출시 이후 AI 평가가 실용화됨.
  - AI 모델의 능력이 평가 자동화에 충분히 적합해짐.
- 현황
  - 2023~2024: AI 평가자는 실제 운영 환경에서 가장 일반적인 평가 방법 중 하나.
  - LangChain 보고서(2023):
    - 58%의 평가가 AI 평가자에 의해 수행됨.
  - AI 평가자는 평가 자동화 스타트업과 연구에서 핵심 기술로 활용.
- 의의
  - AI 평가자는 비용 절감과 효율성 증대를 가능하게 함.
  - 활발히 연구 및 발전 중인 중요한 AI 활용 사례.

왜 AI를 판사로 사용해야 하는가?

- 장점
  - 빠름: 평가 시간이 짧음.
  - 저렴: 인간 평가에 비해 비용 절감.
  - 사용 용이: 참조 데이터 없이도 작동 가능.
  - 유연성: 정확성, 반복성, 유해성, 건전성, 환각 여부 등 다양한 기준으로 평가 가능.
- 신뢰성
  - AI 모델은 대중의 의견을 집약한 결과를 반영.
  - 적절한 프롬프트로 합리적 판단 가능.
  - 연구 사례:
    - GPT-4와 인간 평가자 간 일치율: 85% (Zheng et al., 2023).
    - AI 평가자와 인간 리더보드 간 상관관계: 0.98 (Dubois et al., 2023).
- 추가 기능
  - 평가 결과에 대한 설명 제공 가능.
    - Figure 3-7: GPT-4가 평가와 판단 근거를 설명한 사례.
- 의의
  - 다양한 애플리케이션에서 활용 가능.
  - 인간 평가 수준에 도달하지 못하더라도, 프로젝트 개발과 초기 방향 설정에 충분히 신뢰할 수 있는 도구로 활용.

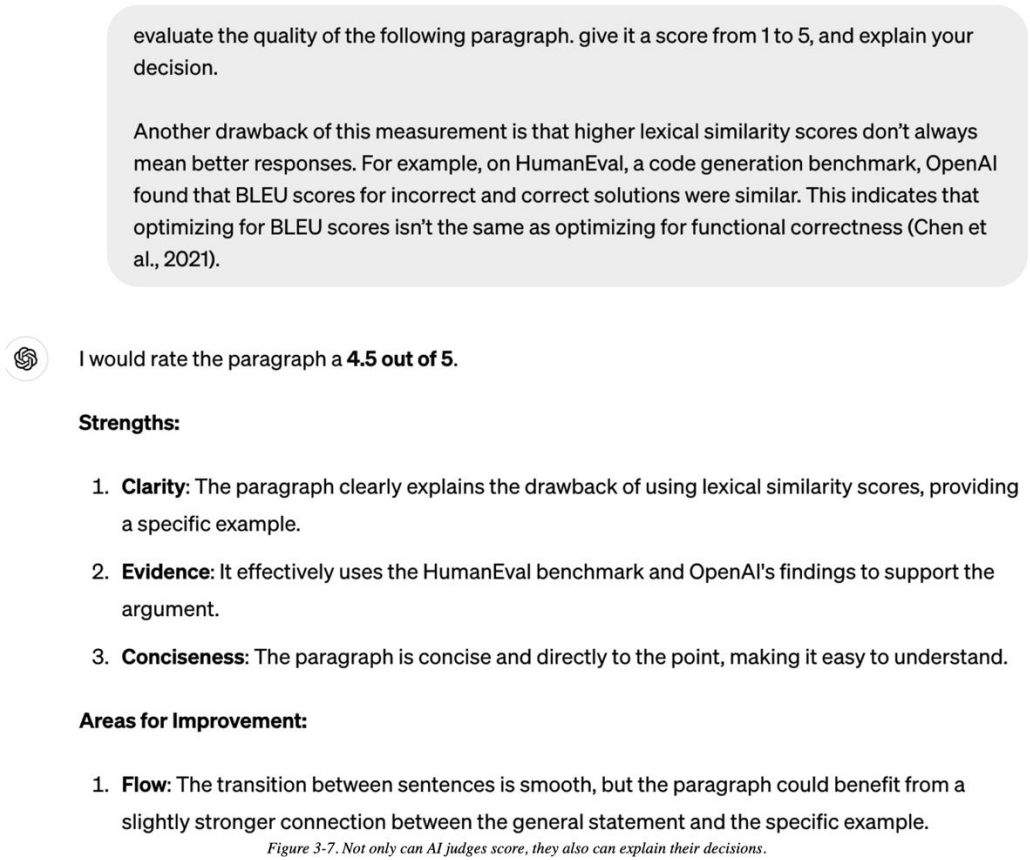


Figure 3-7. Not only can AI judges score, they also can explain their decisions.

AI를 판사로 사용하는 방법

- AI 판사의 주요 역할
  - 응답 평가 방식:
    - 응답 자체 평가: 품질을 1~5점으로 평가.
    - 참조 데이터 비교: 생성된 응답이 참조와 동일한지 평가.
    - 응답 간 비교: 두 응답 중 더 나은 것을 선택.
- AI 평가 기준
  - 기준 예시 (Table 3-3):
    - Azure AI Studio: 근거, 유창성, 유사성 등.
    - LangChain: 정확성, 유해성, 도움 여부 등.
    - Ragas: 신뢰도, 응답 관련성 등.
  - 기준은 도구와 프롬프트 설정에 따라 다름.

Table 3-3. 2024년 9월 기준 일부 AI 도구에서 제공하는 판사 기준의 예

AI 도구	내장 기준
Azure AI Studio	근거, 관련성, 일관성, 유창성, 유사성
MLflow.metrics	신뢰도, 관련성
LangChain Criteria Evaluation	간결성, 관련성, 정확성, 일관성, 유해성, 악의성, 도움 여부, 논란성, 여성혐오, 민감성 부족, 범죄성
Ragas	신뢰도, 응답 관련성

- 효과적 프롬프트 작성
  - 명확한 지침: 작업, 평가 기준, 점수 체계 명확히 설명.
  - 예제 포함: 점수별 응답 사례와 이유 제공 → 품질 향상.
  - 평가 결과 : 분류, 이산적인 숫자, 연속적인 숫자
  - 예시: "질문과 응답 간 관련성을 1~5점으로 평가하고 이유를 설명하세요."
  - 팁: 숫자보다 텍스트 기반 평가에, 연속보다 이산적 숫자에서 더 잘 작동.
- 시각적 예시
  - Figure 3-8: AI 판사가 질문과 응답 간의 품질을 평가하고 점수를 제공하는 과정.
  - AI는 결과와 이유를 명확히 설명하여 평가를 투명하게 만듦.

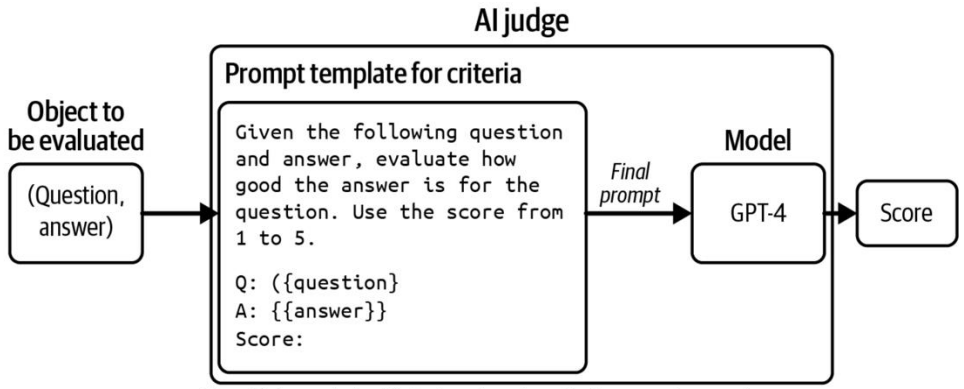


Figure 3-8. An example of an AI judge that evaluates the quality of an answer given a question.

AI 판정자로서의 한계

- 신뢰성 부족
  - AI로 AI를 평가하는 것은 동어반복적으로 보일 수 있음.
  - AI의 확률적 특성은 평가 결과의 신뢰성을 떨어뜨릴 가능성.
- 비용과 성능 문제
  - AI 판정자는 비용 증가와 평가 지연을 초래할 수 있음.
  - 특히 대규모 프로덕션 환경에서는 부담이 될 수 있음.
- 제한적 활용
  - 다른 평가 방법이 없는 상황(참조 데이터 부족 등)에서 대안적 선택지로 사용됨.
  - AI 판정자의 한계를 극복하려면, 신중한 적용과 추가적인 보완 필요.

AI 판정자의 불일치 문제

- 문제점: 평가 결과의 불일치
  - 확률적 특성으로 인해 동일한 입력이라도 다른 결과를 출력할 수 있음.
    - 동일한 판정자가 동일한 입력을 두 번 평가할 경우에도 결과가 다를 수 있음.
  - 불일치는 재현성과 신뢰성을 떨어뜨림.
- 일관성 향상 방법
  - 프롬프트에 평가 예제 포함:
    - Zheng et al. (2023):
      - 예제 추가 시 일관성: 65% → 77.5% 증가.
      - 그러나 정확성 증가로 이어지지 않을 수 있음.
      - 일관된 잘못된 판단 가능성 존재.
      - 긴 프롬프트는 추론 비용 증가 초래.
      - Zheng et al.: 예제 추가로 GPT-4 비용 4배 증가.

기준 모호성

- 문제점: 표준화 부족
  - AI 판정 기준은 표준화되어 있지 않아, 모호성과 오용 가능성 존재.
  - 동일한 기준(Faithfulness)이라도 도구마다 지침과 점수 체계가 다름.

Table 3-4: Faithfulness 기준 비교

도구	점수 체계	특징
MLflow	1–5	출력이 문맥과 얼마나 일치하는지 평가.
Ragas	0과 1	문맥에 근거한 진술의 검증 가능 여부를 이진적으로 평가.
LlamaIndex	YES/NO	문맥의 일부라도 정보가 지지되면 YES를 반환.

- 판정 일관성 문제
  - AI 판정자의 기준과 프롬프트가 변하면 동일한 애플리케이션에서도 평가 결과가 달라질 수 있음.
  - 판정자의 변경이 평가 결과의 변화를 초래했는지, 실제 애플리케이션의 변화 때문인지 구분하기 어려움.
- 신뢰성 확보 팁
  - 판정에 사용된 모델과 프롬프트를 기록 및 확인해야 함.
  - 판정자와 애플리케이션 관리 팀 간 긴밀한 소통 필요.

비용 증가와 지연

- AI 판정자의 비용 문제
  - 프로덕션에서의 활용:
    - AI 판정자를 안전 장치로 사용하여 양호한 응답만 표시.
  - 비용 증가 요인:
    - 강력한 모델 사용 시 API 호출 수 증가.
    - 예: GPT-4로 응답 생성 + 평가 → 비용 2배 증가.
    - 3가지 기준 평가 시 → 비용 4배 증가.
- 비용 절감 방법
  - 약한 모델 활용:
    - 비용을 줄이기 위해 강력한 모델 대신 더 간단한 모델을 판정자로 사용.
  - Spot-checking:
    - 일부 응답만 평가 → 비용 절감 가능.
    - 샘플 비율이 클수록 신뢰도 증가, 하지만 비용도 높아짐.
- 지연 문제
  - AI 판정자의 사용은 응답 딜레이 증가를 초래.
  - 위험 감소와 응답 지연 간의 트레이드오프 발생.
    - 엄격한 지연 요구사항이 있는 애플리케이션에서는 사용 어려울 수 있음.

AI 판정자로서의 편향

- 주요 편향 유형
  - 자기 편향(Self-bias)
    - 모델이 자신이 생성한 응답을 더 선호.
    - Zheng et al. (2023):
      - GPT-4: 자기 응답에 10% 더 높은 승률.
      - Claude-v1: 자기 응답에 25% 더 높은 승률.
  - 첫 번째 위치 편향(First-position bias)
    - 쌍 비교에서 첫 번째 옵션을 선호.
    - 해결 방법: 순서 랜덤화, 반복 테스트, 신중한 프롬프트 설계.
    - 인간 편향과 반대: 인간은 최신성 편향(Recency bias) 을 가짐.
  - 장문 편향(Verbosity bias)
    - 응답 품질과 무관하게 긴 응답 선호.
    - Wu와 Aji (2023): GPT-4와 Claude-1이 긴 응답(100단어)을 짧은 응답(50단어)보다 선호.
    - Zheng et al. (2023): 강력한 모델(GPT-4)은 이 편향이 덜함.
- 기타 한계
  - 데이터 보안: 독점 모델 사용 시 개인 정보 및 지적 재산(IP) 보호 우려.
  - 상업적 안전성: 모델 훈련 데이터가 공개되지 않을 경우, 상업적 리스크 존재.

어떤 모델이 판정자 역할을 할 수 있을까

- 판정자의 유형
  - 강력한 판정자:
    - 장점: 더 정확한 판단, 약한 모델 개선 유도.
    - 단점: 비용 및 지연 증가.
  - 약한 판정자:
    - 장점: 비용과 지연 감소.
    - 단점: 판단 정확성 저하 가능.
  - 자체 평가(Self-evaluation):
    - 모델이 자신의 응답을 평가 → 간단한 오류 확인 및 개선에 유용.

```
질문: 10+3은 무엇입니까?
응답: 30
자체 비판: 이 답변은 잘못되었습니다. 정답은 13입니다.
```

전문화된 판정자

- 약한 판정자도 가능할까?
  - 평가가 생성보다 쉬운 작업일 수 있음.
  - 예: 누군가 노래를 평가할 수는 있지만, 모든 사람이 노래를 작곡할 수는 없음.
  - 약한 모델도 강력한 모델의 출력을 평가할 수 있음.
- 강력한 모델의 장점과 한계
  - 장점:
    - 인간 선호도와 높은 상관관계(Zheng et al., 2023).
    - 약한 모델을 개선하도록 유도 가능.
  - 한계:
    - 강력한 모델 평가 시 적합한 판정자를 찾기 어려움.
    - 평가를 위해 대체 기준 필요.
- 전문화된 판정자 유형
  - 보상 모델 (Reward model)
    - (프롬프트, 응답) 쌍을 입력으로 받아 점수(0~1)를 생성.
    - 예: Cappy(Google, 2023) → 경량 채점기로 정확성 평가.
  - 참조 기반 판정자 (Reference-based judge)
    - 참조 응답과 생성된 응답의 유사성을 점수화.
    - BLEURT: 유사성 점수 출력.
    - Prometheus: 품질 점수(1~5) 제공.
  - 선호도 모델 (Preference model)
    - 두 응답 중 더 나은 것을 선택 및 이유 설명.
    - 예: PandaLM → 선호도와 이유를 함께 출력(Figure 3-9 참고).

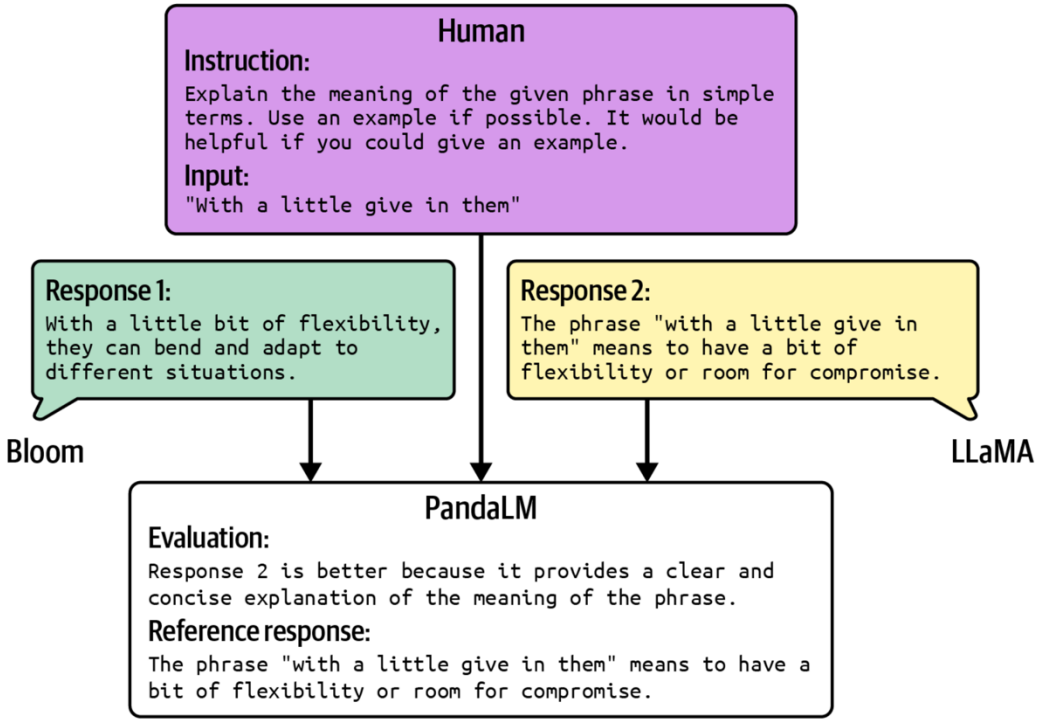


Figure 3-9. An example output of PandaLM, given a human prompt and two generated responses. Picture from Wang et al. (2023), modified slightly for readability. The original image is available under the Apache License 2.0.

비교 평가를 통한 모델 순위 매기기

- 평가 방법
  - 점수 기반 평가(Pointwise Evaluation):
    - 모델별 점수를 독립적으로 평가 후 순위 매김.
    - 예: 무용수를 개별적으로 평가해 점수 부여.
  - 비교 평가(Comparative Evaluation):
    - 모델들을 쌍으로 비교하여 순위를 계산.
    - 예: 댄스 경연에서 판정자가 가장 선호하는 무용수를 선택.
- 비교 평가의 장점
  - 주관적인 평가(예: 선호도)에서 더 쉬움.
  - AI 사례: LMSYS의 Chatbot Arena에서 쌍 비교(pairwise comparison) 기반 리더 보드
  - Figure 3-10: 두 모델의 출력을 사용자에게 비교하게 하는 예.

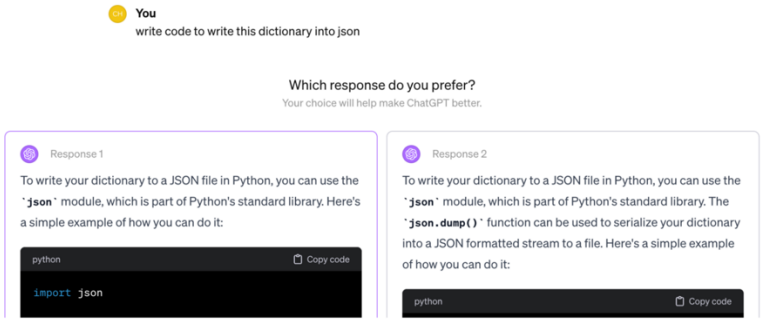


Figure 3-10. ChatGPT occasionally asks users to compare two outputs side by side.

- 한계 및 주의점
  - 선호도 vs 정답:
    - 모든 질문이 선호도로 평가될 수 없음.
    - 예: "휴대폰 방사선이 뇌종양과 관련이 있습니까?" → 정확성 평가 필요.
  - 사용자 좌절 가능성:
    - 사용자가 정답을 모르는데 선택을 요구 받을 경우 혼란.
- 비교 평가의 구성
  - 매치(match): 두 모델을 비교한 결과.
  - Table 3-5: 매치 결과의 예.

매치 #	모델 A	모델 B	승자
1	Model 1	Model 2	Model 1

  - 승률(win rate): A가 B를 이기는 비율.
    - 승률 기반으로 모델 간 순위를 매김.
- 순위 계산 알고리즘
  - 알고리즘 예시:
    - Elo, Bradley–Terry, TrueSkill.
    - LMSYS Chatbot Arena: 초기 Elo → Bradley–Terry로 전환.
  - 순위 예측:
    - 순위는 과거 매치 데이터를 바탕으로 미래 결과를 예측.

비교 평가의 도전 과제

- 점수 기반 평가의 특징
  - 벤치마크와 지표 설계가 핵심.
  - 점수 계산 후 순위를 매기는 과정은 비교적 간단.
- 비교 평가의 도전 과제
  - 신호 수집:
    - 쌍 비교를 통해 신뢰할 수 있는 데이터를 수집하는 것이 어려움.
  - 모델 순위 매기기:
    - 쌍 비교 데이터를 기반으로 정확한 순위를 산출하는 과정이 복잡.
  - 평가 설계:
    - 평가 방식과 조건을 신중히 설정하지 않으면 결과가 왜곡될 가능성.

Table 3-6. 다섯 개 모델의 승률 예제. A >> B 열은 A가 B보다 선호되는 사건을 나타냅니다.

Model pair #	Model A	Model B	# matches	A >> B
1	Model 1	Model 2	1000	90%
2	Model 1	Model 3	1000	40%
3	Model 1	Model 4	1000	15%
4	Model 1	Model 5	1000	10%
5	Model 2	Model 3	1000	60%
6	Model 2	Model 4	1000	80%
7	Model 2	Model 5	1000	80%
8	Model 3	Model 4	1000	70%
9	Model 3	Model 5	1000	10%
10	Model 4	Model 5	1000	20%

확장성의 병목 현상

- 문제: 모델 수 증가에 따른 비교 부담
  - 비교 평가는 데이터 집약적으로, 모델 쌍 수가 모델 수에 따라 제곱 비율로 증가.
    - 예: 57개 모델 → 1,596개 쌍 → 평균 153회 비교(총 244,000회).
  - 다양한 작업을 평가하기에는 데이터가 여전히 부족.
- 전이성(transitivity) 가정
  - 전이성 활용:
    - 예: A > B, B > C → A > C로 가정.
    - 전이성을 통해 직접 비교 수를 줄일 수 있음.
  - 한계:
    - 인간 선호와 평가 조건은 비전이성(non-transitivity)을 나타낼 수 있음.
    - 서로 다른 평가자나 프롬프트로 인해 결과가 불일치 가능.
- 새로운 모델 평가의 도전
  - 비교 평가는 기존 모델과의 추가 비교 필요.
    - 새로운 모델 도입 시 기존 순위도 재조정 가능.
  - 비공개 모델 평가:
    - 자체 데이터로 모델 개발 시, 공개 리더보드 활용 또는 자체 비교 신호 생성 필요.
- 해결책: 효율적인 매칭 알고리즘
  - 모든 모델 쌍을 동일하게 비교할 필요 없음.
    - 특정 모델 쌍에 대한 결과가 명확하면 추가 비교를 생략.
  - 효율적 샘플링:
    - 불확실성이 큰 매치에 우선순위를 두어 비교 효율성 개선.

표준화 및 품질 관리 부족

- 크라우드소싱 평가의 장단점
  - 장점:
    - 다양한 신호를 캡처하고 조작이 어렵다.
  - 단점:
    - 표준화와 품질 관리가 어렵다.
    - 사용자 선호도가 잘못된 방향으로 모델 순위를 왜곡할 수 있음.
    - 예: 독성이 높은 응답 선호 또는 부적절한 요청 거부에 대한 다운보트.
- 평가 환경의 한계
  - 현실과의 연계 부족:
    - 단순하고 정교하지 않은 프롬프트 사용(예: "hello", "hi").
    - 복잡한 실제 작업을 충분히 반영하지 못함.
  - 단순 프롬프트 문제:
    - 단순한 질문은 모델 성능 차이를 구별하기 어려움.
    - 랭킹 왜곡 가능.
- 개선 방안
  - 어려운 프롬프트 활용:
    - LMSYS는 **어려운 프롬프트(hard prompts)**를 필터링해 평가에 사용.
  - 신뢰할 수 있는 평가자 도입:
    - 평가자 교육 및 훈련으로 정교한 기준 제공.
    - 예: Scale의 비공개 비교 리더보드.
    - 단점: 비용 상승 및 비교 횟수 감소.
  - 제품 내 비교 통합:
    - 사용자 작업 흐름에 비교 평가 포함(예: 코드 생성기에서 두 코드 조각 비교).
    - 위험: 무작위 클릭이나 전문성 부족으로 인한 잡음 발생.
  - AI 평가자 사용:
    - 무작위 사용자보다 신뢰도 높음.
    - 전문 평가자만큼 정교하지는 않을 수 있음.

비교 성능에서 절대 성능으로

- 비교 평가의 한계
  - 비교 평가는 모델 간의 상대적 우위를 제공하지만, 모델의 절대 성능이나 특정 애플리케이션에 적합한지 여부는 알 수 없음.
  - 예: 모델 B가 모델 A보다 나은 경우에도, 두 모델 모두 좋지 않을 수 있음.
- 절대 성능 평가의 필요성
  - 특정 시나리오를 파악하려면 비교 평가 외에 추가적인 평가가 필요:
    - 모델 B는 좋고, 모델 A는 나쁘다.
    - 모델 A와 B 모두 나쁘다.
    - 모델 A와 B 모두 좋다.
  - 예: 모델 A가 고객 지원 요청의 70%를 해결한다고 가정.
    - 모델 B의 51% 승률이 실제로 얼마나 더 많은 요청을 해결할지 모호.
- 비용-편익 분석
  - 성능 향상 vs 비용 고려:
    - 모델 B가 더 나은 성능을 제공하더라도, 비용이 모델 A의 2배라면, 성능 향상이 추가 비용을 정당화할 수 있을지 불확실.
  - 성능 개선의 규모와 애플리케이션의 특성에 따라 비용 대비 효과가 다름.

비교 평가의 미래

- 비교 평가의 이점
  - 평가 용이성:
    - 구체적인 점수를 주는 것보다 두 응답을 비교하는 것이 더 쉬움.
    - 모델이 인간 성능을 넘어서는 경우에도 비교를 통해 차이를 감지 가능.
    - 예: Llama 2 논문에서 인간 주석자의 도움으로 비교 피드백 제공(Touvron 외, 2023).
  - 인간 선호를 포착:
    - 인간이 중요하게 생각하는 품질을 평가.
    - 강력한 모델 도입 시 벤치마크가 쓸모없어지는 문제를 완화.
- 신뢰성과 유용성
  - 속이기 어려움:
    - 참조 데이터 기반 훈련보다 조작 가능성 낮음.
    - 공개 비교 리더보드의 결과가 더 신뢰받음.
  - 다양한 평가 신호 제공:
    - 오프라인: 기존 벤치마크에 추가적인 평가 신호 제공.
    - 온라인: A/B 테스트를 보완하여 더 나은 평가 가능.

요약

- 평가의 중요성
  - 강력한 AI 모델은 치명적 실패 가능성이 높아 평가의 중요성이 증가.
  - 개방형 및 강력한 모델 평가에는 많은 도전 과제가 존재.
  - 인간 평가가 여전히 유용하며, 많은 경우 필수적이지만, 자동 평가 방식을 중점적으로 논의.
- 평가의 도전과 투자 부족
  - 기본 모델 평가의 어려움: 전통적 기계 학습 모델보다 더 복잡.
  - 평가에 대한 투자가 여전히 모델 및 애플리케이션 개발에 비해 부족.
- 언어 모델링 지표
  - Perplexity, Cross-Entropy 등 언어 모델링 지표에 초점.
  - 지표의 의미와 데이터 처리 및 평가 활용법에 대한 명확한 해석 제공.
- 개방형 응답 평가 접근 방식
  - 기능적 정확성:
    - 명확한 기준으로 응답 평가.
  - 유사성 점수:
    - 정답과의 유사성을 기반으로 평가.
  - AI 심판 기반 평가:
    - 주관적 평가로, 심판의 맥락에 따라 점수 해석 필요.
    - AI 심판은 보완적 평가(정확한 평가와 인간 평가)로 활용 권장.
- 비교 평가
  - 모델 순위 매기기:
    - 독립적 점수 기반 순위 또는 비교 평가 활용.
  - 비교 평가는 스포츠와 같은 분야에서 일반적이며 AI 평가에서도 증가 추세.
    - 선호 신호(preference signals) 제공 → 선호 모델 개발 촉진.
- 평가 기술의 발전
  - 전통적 지표(Perplexity, 유사성 측정)에서 시작해 AI 심판과 비교 평가 채택.
  - 많은 팀들이 이를 평가 파이프라인에 통합 중.
- 결론 및 다음 장 예고
  - 개방형 응답 애플리케이션 평가를 위한 신뢰할 수 있는 평가 파이프라인 구축이 다음 주제.