Project (gsm8k → gpt-4o 모델)

https://github.com/Mohammadjafari80/GSM8K-RLVR/tree/main

수학 문제 데이터셋 하나를 정해서 그 위에서 문제를 풀려보고, accuracy 및 틀린 sample을 파악→

모델은 GPT-4o(https://platform.openai.com/docs/guides/text?api-mode=responses) 로, **데이터셋은 GSM8K**(https://huggingface.co/datasets/openai/gsm8k)로

LLM 예제 튜토리얼: https://github.com/mov-z/LLM-Tutorials/tree/main?tab=readme-ov-file

langchain 설명 위키독스: https://wikidocs.net/book/14314, 블로그; https://pointer81.tistory.com/entry/about-langchain

```
from openai import OpenAI
from datasets import load_dataset
from tqdm import tqdm
import time
import re

# OpenAI API 키 설정
#api_key = "sk-proj-4PTUrPQ3lWTKlvjAd45GBCmMMnmSufiz_8b5Blv9l8kPV
client = OpenAI(api_key=api_key)

# 사용할 샘플 수 설정
max_samples = 100
dataset = load_dataset("gsm8k", "main", split=f"test[:{max_samples}]")

# GPT-4o 호출 함수 (프롬프트 없이)
def query_gpt(question: str, model="gpt-4o"):
    try:
        response = client.chat.completions.create(
            model=model,
            messages=[
                {"role": "system", "content": "You are a helpful math assistant."},
```

```
            {"role": "user", "content": question}
        ],
        temperature=0,
    )
    return response.choices[0].message.content
  except Exception as e:
    print(f"Error in query_gpt: {e}")
    return None

# 숫자 정답 추출 함수 -> 문장으로 결과를 보았을 땐 올바르게 추출했는데, 이 부분에서 오
def extract_final_answer(output: str):
  import re

  # 1. 쉼표 제거
  output_cleaned = output.replace(",", "")

  # 2. "boxed" 스타일 처리
  boxed = re.search(r"\\boxed{(\d+)}", output_cleaned)
  if boxed:
    return boxed.group(1)

  # 3. "Therefore, ... $X" or "The answer is X" 등의 문장에서 추출
  for line in reversed(output_cleaned.splitlines()):
    if any(kw in line.lower() for kw in ["therefore", "answer", "final", "so", "in c
      nums = re.findall(r"[-+]?\d*\.?\d+", line)
      if nums:
        return nums[0]  # 문장 내 마지막 숫자

  # 4. 전체에서 마지막 숫자 (백업용)
  matches = re.findall(r"[-+]?\d*\.?\d+", output_cleaned)
  return matches[0] if matches else None

'''def extract_final_answer(output: str):
  matches = re.findall(r"[-+]?\d*\.?\d+", output)
  return matches[-1] if matches else None
'''
# 평가 루프
correct = 0
```

```
results = []

for example in tqdm(dataset):
    question = example["question"]
    gt_answer = example["answer"].split("####")[-1].strip()

    output = query_gpt(question)
    pred_answer = extract_final_answer(output) if output else "None"

    is_correct = pred_answer == gt_answer
    correct += is_correct

    results.append({
        "question": question,
        "ground_truth": gt_answer,
        "prediction": pred_answer,
        "full_output": output,
        "is_correct": is_correct
    })

    time.sleep(1.0)  # OpenAI API rate limit 고려

# 결과 출력
accuracy = correct / max_samples
print(f"\nAccuracy: {accuracy:.2%}")

# 틀린 문제 일부 출력
wrong_samples = [r for r in results if not r["is_correct"]]
print(f"\nWrong samples ({len(wrong_samples)}):")
for sample in wrong_samples[:5]:  # 최대 5개
    print(f"\nQ: {sample['question']}")
    print(f"GT: {sample['ground_truth']} | Pred: {sample['prediction']}")
    print(f"Model Output:\n{sample['full_output']}")
```

출력값:

```
100%|██████████| 1000/1000 [1:33:11<00:00,  5.59s/it]
```

Accuracy: 78.50%

Wrong samples (215):

Q: Josh decides to try flipping a house. He buys a house for $80,000 and t
hen puts in $50,000 in repairs. This increased the value of the house by 15
0%. How much profit did he make?
GT: 70000 | Pred: 000
Model Output:
To determine Josh's profit, we need to calculate the final value of the hous
e after the repairs and then subtract his total expenses from this value.

1. **Calculate the total cost incurred by Josh:**
   - Purchase price of the house: $80,000
   - Cost of repairs: $50,000
   - Total cost = $80,000 + $50,000 = $130,000

2. **Determine the increased value of the house:**
   - The value of the house increased by 150% due to the repairs.
   - Original value of the house before repairs = $80,000
   - Increase in value = 150% of $80,000 = 1.5 × $80,000 = $120,000

3. **Calculate the new value of the house:**
   - New value = Original value + Increase in value
   - New value = $80,000 + $120,000 = $200,000

4. **Calculate the profit:**
   - Profit = New value of the house - Total cost
   - Profit = $200,000 - $130,000 = $70,000

Therefore, Josh made a profit of $70,000.

Q: Kylar went to the store to buy glasses for his new apartment. One glass
costs $5, but every second glass costs only 60% of the price. Kylar wants t
o buy 16 glasses. How much does he need to pay for them?
GT: 64 | Pred: 16
Model Output:
To determine the total cost for 16 glasses, we need to consider the pricing

structure: the first glass costs $5, and every second glass costs 60% of $5.

First, calculate the cost of every second glass:
\[60\% \text{ of } \$5 = 0.6 \times 5 = \$3. \]

Now, let's pair the glasses into sets of two:
- The first glass in each pair costs $5.
- The second glass in each pair costs $3.

Since Kylar wants to buy 16 glasses, he will have 8 pairs of glasses (since \(16 \div 2 = 8\)).

Calculate the cost for one pair of glasses:
\[\$5 + \$3 = \$8. \]

Now, calculate the total cost for 8 pairs:
\[8 \times \$8 = \$64. \]

Therefore, Kylar needs to pay \(\$64\) for 16 glasses.

Q: Carlos is planting a lemon tree. The tree will cost $90 to plant. Each year it will grow 7 lemons, which he can sell for $1.5 each. It costs $3 a year to water and feed the tree. How many years will it take before he starts earnin g money on the lemon tree?
GT: 13 | Pred: 12
Model Output:
To determine how many years it will take for Carlos to start earning money from the lemon tree, we need to calculate the net profit each year and see when it becomes positive.

1. **Initial Cost:**
   - Planting the tree costs $90.

2. **Annual Revenue:**
   - Each year, the tree produces 7 lemons.
   - Each lemon can be sold for $1.5.
   - Annual revenue from selling lemons = \(7 \times 1.5 = 10.5\) dollars.

3. **Annual Costs:**
   - It costs $3 per year to water and feed the tree.

4. **Annual Net Profit:**
   - Annual net profit = Annual revenue - Annual costs
   - Annual net profit = \(10.5 - 3 = 7.5\) dollars.

5. **Break-even Calculation:**
   - Carlos needs to recover the initial planting cost of $90.
   - Each year, he earns a net profit of $7.5.
   - To find out how many years it takes to recover the $90, we solve the eq
uation:
     \[
     7.5 \times \text{years} = 90
     \]
   - Solving for years:
     \[
     \text{years} = \frac{90}{7.5} = 12
     \]

Therefore, it will take Carlos 12 years to start earning money from the lemo
n tree.

Q: Jill gets paid $20 per hour to teach and $30 to be a cheerleading coach. If she works 50 weeks a year, 35 hours a week as a teacher and 15 hours a week as a coach, what's her annual salary?
GT: 57500 | Pred: 500
Model Output:
To calculate Jill's annual salary, we need to determine her earnings from bo
th teaching and coaching separately, and then sum them up.

First, let's calculate her annual earnings from teaching:
- Jill works 35 hours per week as a teacher.
- She gets paid $20 per hour for teaching.
- She works 50 weeks a year.

The annual earnings from teaching are calculated as follows:

\[
35 \text{ hours/week} \times 20 \text{ dollars/hour} \times 50 \text{ weeks/year} = 35 \times 20 \times 50 = 35,000 \text{ dollars}
\]

Next, let's calculate her annual earnings from coaching:
- Jill works 15 hours per week as a coach.
- She gets paid $30 per hour for coaching.
- She works 50 weeks a year.

The annual earnings from coaching are calculated as follows:
\[
15 \text{ hours/week} \times 30 \text{ dollars/hour} \times 50 \text{ weeks/year} = 15 \times 30 \times 50 = 22,500 \text{ dollars}
\]

Finally, we add her earnings from teaching and coaching to find her total annual salary:
\[
35,000 \text{ dollars} + 22,500 \text{ dollars} = 57,500 \text{ dollars}
\]

Therefore, Jill's annual salary is \(\boxed{57,500} \) dollars.

Q: Claire makes a 3 egg omelet every morning for breakfast. How many dozens of eggs will she eat in 4 weeks?
GT: 7 | Pred: 4
Model Output:
To find out how many dozens of eggs Claire will eat in 4 weeks, we can follow these steps:

1. Calculate the total number of eggs Claire uses in one day:
   - Claire uses 3 eggs per omelet.

2. Calculate the total number of eggs Claire uses in one week:
   - There are 7 days in a week, so she uses \(3 \text{ eggs/day} \times 7 \text{ days/week} = 21 \text{ eggs/week}\).

3. Calculate the total number of eggs Claire uses in 4 weeks:
   - \(21 \text{ eggs/week} \times 4 \text{ weeks} = 84 \text{ eggs}\).

4. Convert the total number of eggs to dozens:
   - There are 12 eggs in a dozen, so \(\frac{84 \text{ eggs}}{12 \text{ eggs/dozen}} = 7 \text{ dozens}\).

Therefore, Claire will eat 7 dozens of eggs in 4 weeks.

위 코드 문제점: gpt4 모델의 출력값은 자연어처리 문장 형태로 뱉어서, 수동으로 정답 추출 함수를 만들어서 (extract_final_answer(output: str)) 정답을 추출하도록 하였더니, 문장에 포함된 숫자의 형태도 다양하고 여러개의 숫자가 혼합되어 추출되어 특정한 순서로 고정시켜서 정답을 추출하게 하면 제대로 정답을 문장에선 말을 했어도, 정확도를 측정할 때는 실제 정확도보다 낮게 나오게 되는 케이스가 다수 발생함 (정확도: 78.5 %)
→ 프롬프트 추가하여 정답 뱉는 형식을 통일 시키는 방식 활용하여 코드 수정 (정확도 95%로 확 오름)

추가로 수정한 부분:
현재
`extract_final_answer` 함수에서 정답 숫자를 문자열로 추출할 때 소수점 아래 `.00`까지 포함해서 추출합니다. 그런데 정답(GT)에는 정수형(예: `"6"`)으로 되어 있어서 비교 시 `6.00` ≠ `6`가 되어 틀린 것으로 처리되는 거죠.

(
`GT`, 즉 `ground_truth`)은 `"114,200"`처럼 **쉼표가 포함된 문자열**인 반면, `pred_answer`는 `"114200"`처럼 쉼표 없이 출력되어 비교 시 **불일치**로 판단되는 문제

```
from openai import OpenAI
from datasets import load_dataset
from tqdm import tqdm
import time
import re

# API 클라이언트 설정
client = OpenAI(api_key=api_key)
```

```
# 사용할 샘플 수
max_samples = 1000
dataset = load_dataset("gsm8k", "main", split=f"test[:{max_samples}]")

# GPT-4o 호출 함수 (정답 형식 유도 프롬프트 포함)
def query_gpt(question: str, model="gpt-4o"):
    try:
        prompt = (
            f"Q: {question.strip()}\n\n"
            "A: Please answer the question and provide only the final answer at the
        )

        response = client.chat.completions.create(
            model=model,
            messages=[
                {"role": "system", "content": "You are a helpful math assistant."},
                {"role": "user", "content": prompt}
            ],
            temperature=0,
        )
        return response.choices[0].message.content
    except Exception as e:
        print(f"Error in query_gpt: {e}")
        return None

# 정답 숫자 추출 함수 (The answer is [number] 형식에서 추출)
def extract_final_answer(output: str):
    if not output:
        return None

    output_cleaned = output.replace(",", "")

    match = re.search(r"The answer is\s+([-+]?\d*\.?\d+)", output_cleaned, re.I
    if match:
        num_str = match.group(1)

    else:
        boxed = re.search(r"\\boxed{([-+]?\d*\.?\d+)}", output_cleaned)
```

```
        if boxed:
            num_str = boxed.group(1)
        else:
            num_str = None
            for line in reversed(output_cleaned.splitlines()):
                if any(kw in line.lower() for kw in ["therefore", "answer", "final", "so",
                    nums = re.findall(r"[-+]?\d*\.?\d+", line)
                    if nums:
                        num_str = nums[0]
                        break
            if num_str is None:
                matches = re.findall(r"[-+]?\d*\.?\d+", output_cleaned)
                num_str = matches[-1] if matches else None

    if num_str is None:
        return None

    # 숫자 문자열을 float으로 변환 후, 정수라면 int로 변환해 소수점 제거
    try:
        num_float = float(num_str)
        if num_float.is_integer():
            return str(int(num_float))
        else:
            return str(num_float)
    except:
        return num_str


# 평가 루프
correct = 0
results = []

# 쉼표 제거 + 소수점 정리 함수
def normalize_answer(ans: str):
    try:
        ans_clean = ans.replace(",", "")
        num = float(ans_clean)
        if num.is_integer():
```

```
            return str(int(num))
        else:
            return str(num)
    except:
        return ans.strip()

# 평가 루프 중 비교 부분
for example in tqdm(dataset):
    question = example["question"]
    gt_answer = normalize_answer(example["answer"].split("####")[-1].strip())

    output = query_gpt(question)
    pred_answer_raw = extract_final_answer(output) if output else "None"
    pred_answer = normalize_answer(pred_answer_raw)

    is_correct = pred_answer == gt_answer
    correct += is_correct

    results.append({
        "question": question,
        "ground_truth": gt_answer,
        "prediction": pred_answer,
        "full_output": output,
        "is_correct": is_correct
    })

    time.sleep(1.0)  # OpenAI API rate limit 고려


# 결과 출력
accuracy = correct / max_samples
print(f"\nAccuracy: {accuracy:.2%}")

# 틀린 문제 일부 출력
wrong_samples = [r for r in results if not r["is_correct"]]
print(f"\nWrong samples ({len(wrong_samples)}):")
for sample in wrong_samples[:]:  # 틀린 경우 모두 출력
    print(f"\nQ: {sample['question']}")
```

```
print(f"GT: {sample['ground_truth']} | Pred: {sample['prediction']}")
print(f"Model Output:\n{sample['full_output']}")
```

- Responses API 가 아닌 Chat Completion API 사용한 이유:
  → 현재(2025년 기준), responses API는 "chat-completions API"의 단순화 버전으로, 특정 용도에 최적화된 abstraction입니다.
  그러나 모든 기능을 지원하지 않고, GPT-4o처럼 정교한 제어가 필요한 경우엔 chat.completions.create() 같은 저수준 API를 사용하는 것이 더 낫습니다.

  Responses API: OpenAI가 2024년 말에 새롭게 도입한 API abstraction.
  목표: 간단한 작업(ex. 문서 요약, 질문 답변)을 더 쉽게 처리하도록 함.
  대표적인 특징:
  openai.responses.create() 사용
  chat/completion 구분 없이 "응답만 얻는다"는 관점에 집중
  prompt 한 줄이면 끝

  ⚠️ responses API의 한계
  한계 설명
  - 역할 설정 불가 multi-turn conversation 관리 어려움
  - 세밀한 응답 제어 어려움 step-by-step 출력 요구, 포맷 제한, temperature 설정 등 한계 있음
  - 고급 실험엔 부적합 chain-of-thought prompting, few-shot 예시 삽입 등 비효율적
  - 시스템 메시지 제어 불가 "You are a helpful math assistant" 같은 system prompt 사용 불가

  왜 chat.completions.create()를 사용하는가?
  현재 실험 목적이:
  GSM8K 같은 추론 기반 수학 문제 해결,
  정확한 프롬프트 조절 (CoT 유도 등),
  모델의 내부 reasoning 출력 관찰 이기 때문에, 더 강력하고 유연한 컨트롤이 가능한 chat API가 적합합니다.