基于大语言模型的思维链推理

Group 20

贡献百分比
实验简介
Part1：Multiple Reasoning Paths & Self-Consistency
1.1 实现原理
1.2 实现过程
1.3 实验结果
1.4 方法总结
1.4.1 方法优越性
1.4.2 方法局限性
Part2: 针对不同推理路径构建verifier
2.1 实现原理
2.1.1 任务定义
2.1.2 核心思想
2.1.3 方法优越性
2.1.4 方法局限性
2.2 实验结果
2.2.1 单条路径
2.2.2 多条路径
Part3: 使用Retrieval方法为思维链提供先验知识
3.1 实现原理
3.1.1 实验动机
3.1.2 Retrieval方法概述
3.1.3 方法迁移及挑战分析
3.2 方法1——bruteforce
3.2.1 问题分析
3.3 方法2——精简database
3.3.1 问题分析
3.4 方法3——Human in the Loop（upper-bound）
3.4.1 局限性分析及改进展望
3.5 实验结果
3.5.1 方法1
3.5.2 方法2
3.5.3 方法3

贡献百分比

徐铭	梅昕宇	陈科睿	卢峰杰	嵇嘉宇

实验简介

在lab4的基础上，我们小组对“基于大语言模型的思维链推理”进行了相关进阶探索，我们的探索分为三个部分

- **Part1**：Multiple Reasoning Paths & Self-Consistency

  > 基于《*Self-Consistency Improves Chain of Thought Reasoning in Language Models*》

- **Part2**：针对不同推理路径构建verifier

  > 基于《*Making Large Language Models Better Reasoners with Step-Aware Verifier*》

- **Part3**：使用Retrieval方法为思维链提供先验知识
  - **方法1**: bruteforce
  - **方法2:** 精简database
  - **方法3:** Human in the Loop

  > 基于《*Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions*》

数据集：*AQuA* https://github.com/deepmind/AQuA

Part1： Multiple Reasoning Paths & Self-Consistency

1.1 实现原理

对于人类来说，每个人的解决问题的思路是不一样的，对于同一个问题，不同人也会想到不同的解法。自然，我们可以假设对于那些要求复杂思考的问题，LLM也可以得到多种路径去得到答案。一个模型可以生成多种合理的路径去得到正确的答案，但是不合理的路径得到相同错误答案的可能行相较前者来说显然较小。也就是说，我们假设正确的推理过程即使是多样化的，在最终答案上的一致性往往比错误的过程更高。

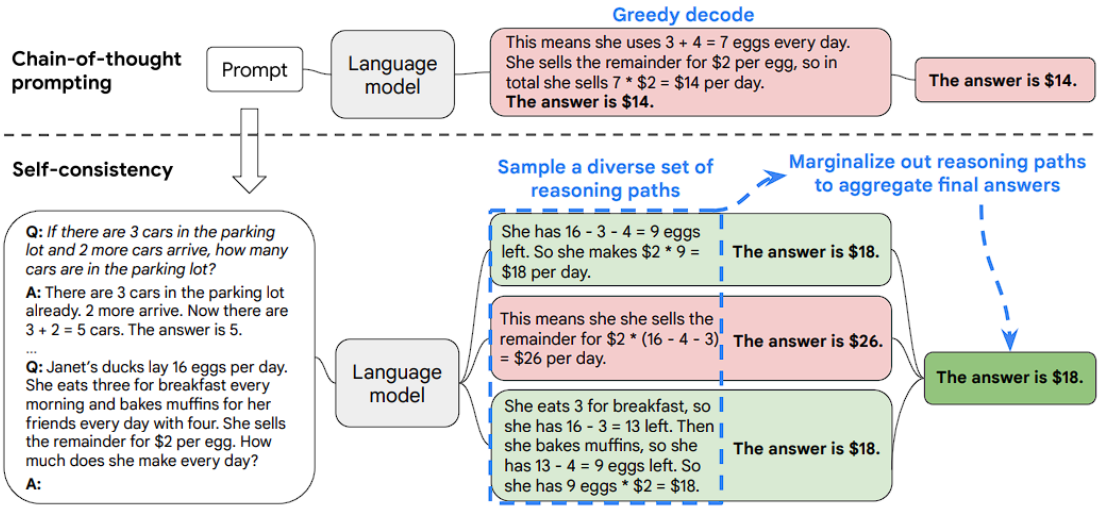


Figure 1: The self-consistency method contains three steps: (1) prompt a language model using chain-of-thought (CoT) prompting; (2) replace the "greedy decode" in CoT prompting by sampling from the language model's decoder to generate a diverse set of reasoning paths; and (3) marginalize out the reasoning paths and aggregate by choosing the most consistent answer in the final answer set.

1.2 实现过程

具体实验中，我们首先利用chain of thought生成多条推理路径和答案，最终选择出现最多次的答案作为最终答案输出。

```
num_paths = 5  # Number of reasoning paths to generate
```

```
for _ in range(num_paths):
    trigger_sentence = random.choice(trigger_sentences)
    X0 = f"Q: {question} A: {trigger_sentence}"
    Z = inference(X0)
    final_prompt = f"{X0} {Z} Therefore, among A through E,
    final_answer = inference(final_prompt)
    prediction = None
    for char in final_answer:
        if char.isupper():
            prediction = char
            break
    path_answers.append(prediction)
```

```
# Use self-consistency to select the most common answer
final_prediction = most_common(path_answers)
if final_prediction == true_label:
    correct_count += 1
total_count += 1
```

1.3 实验结果

图中可见，除了for循环使推理重复了num_paths次外，推理的具体过程和正常COT没有什么区别，在num_paths次训练完后，挑选出现次数最多的answer作为最终的预测。

```
E
['D', 'D', 'D', 'C', 'D']
C
['C', 'C', 'C', 'C', 'C']
B
['D', 'B', 'A', 'C', 'B']
A
['C', 'C', 'C', 'A', 'C']
E
['D', 'E', 'E', 'E', 'B']
A
['C', 'C', 'A', 'A', 'C']
A
['A', 'A', 'A', 'B', 'B']
E
['E', 'C', 'E', 'C', 'C']
B
['E', 'B', 'A', 'C', 'C']
C
['C', 'C', 'C', 'C', 'C']
Epoch 1/1 - Accuracy: 50.00%
Accuracies over epochs: [0.5]
```

由于数据集的数学问题比较复杂以及训练资源有限等原因，我们采取数据集中的前十条，每个问题生成5个COT，最终的准确率是50%。

```
Q: Question: Two friends plan to walk along a 43-km trail, starting at opposite ends of

To find out when they meet, we need to determine the time it takes for them to cover th

Since this distance equals 43 km, we can set up the equation:

\[ 2.15Qt = 43 \]

To find the time when they pass each other, we solve for \( t \):

\[ t = \frac{43}{2.15Q} \]

Now, since Friend P walks at a rate of \( 1.15Q \), the distance \( P \) travels in tha

\[ P = 1.15Q \times t = 1.15Q \times \frac{43}{2.15Q} \]

Simplifying, we get:

\[ P = 1.15 \times \frac{43}{2.15} \]
\[ Therefore, among A through E, the answer is(only one alphabet, no need to write the
Q: Question: In the coordinate plane, points (x, 1) and (5, y) are on line k. If line k

If we start at the point (0,0) and move 5 units along the x-axis (since the slope is 1/

Therefore, the value of x is 5 and the value of y is 1.
...
The only remaining option that results in an even last digit for 8XY is C) 12, where X

Therefore, the correct answer is C: 12. Therefore, among A through E, the answer is(onl
Accuracy: 40.00%
```

与不添加self-consistency的baseline对比，可见accuracy提高了10%，和原论文12%的提高率相近。

1.4 方法总结

1.4.1 方法优越性

1、提高准确性

2、提高鲁棒性，通过多条独立的推理路径，即使某些路径由于模型限制或数据噪音导致错误，整体结果仍然可以保持稳定。

3、可以非常简单地与其他优化方法融合

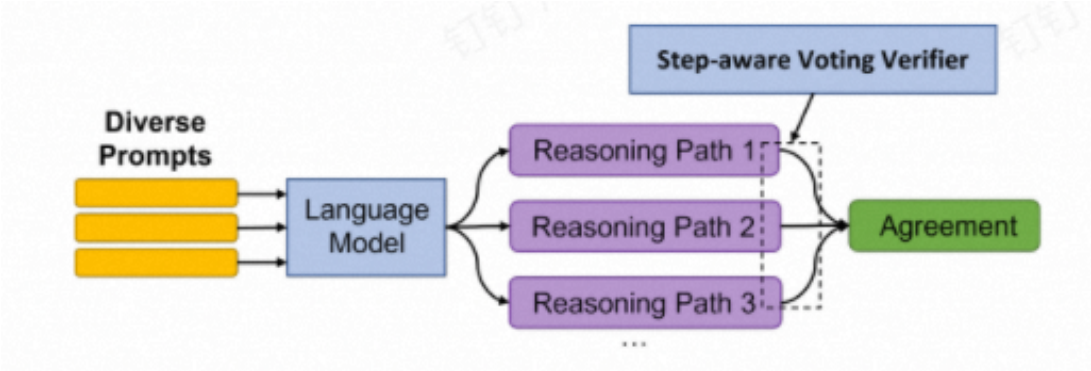
1.4.2 方法局限性

1、计算资源消耗大

2、复杂度增加

Part2: 针对不同推理路径构建verifier

Making Large Language Models Better Reasoners with Step-Aware Verifier



2.1 实现原理

2.1.1 任务定义

在Part1的多条思维链的基础上，增加了对结果的验证环节，即我们这里的Voting Verfier。我们希望通过这里的小模型来实现对推理路径的综合统一，从而得到一个更为准确和合理的答案。

考虑的调用方便的缘故，我们没有使用本地的方案，使用了通义千问的qwen-turbo模型作为verifier。

2.1.2 核心思想

1. **单条路径**：用第一个model（qwen-long）跑cot，然后生成结果，然后让verifier（qwen-turbo）来判断推理是否有误，如果有误，就用这个model重新算一遍，作为结果，然后计算准确率。

```
# 验证预测答案
if not verify_answer(question, final_prompt+prediction):
    # 如果验证失败，重新推理答案
    reinference_prompt = f"{X0}" + f" The answer is not {prediction}." + f" Therefore, the answer
    (chosen among A through E) is"
    corrected_answer = re_inference(reinference_prompt)
    for char in corrected_answer:
        if char.isupper():
            prediction = char
            break
```

2. **多条路径**：多条路径使用不同的cot得到若干结果进行投票，当发现投票结果不全一样时，则用verifier判断最高得分的选项是否正确，如果不正确，则重新推理一遍，并将答案作为最终推理结果。

```
first_prediction = most_common(path_answers)
second_prediction = sec_common(path_answers)
if second_prediction != None:
    # 如果存在第二种答案，用verifier检验
    print("verifying answer...")
    cot = f"{Z} Therefore, among A through E, the answer is {first_prediction}"
    if not verify_answer(question, cot):
        reinference_prompt = f"{X0}" + f" The answer is not {first_prediction}." + f" Therefore,
        the answer is(only one alphabet, no need to write the whole word, e.g., 'A' or 'B' or 'C'
        or 'D' or 'E') "
        print("verify failed, reinferencing...")
        corrected_answer = re_inference(reinference_prompt)
        prediction = ""
        for char in corrected_answer:
            if char.isupper():
                prediction = char
                break
        final_answer = prediction
    else:
        final_answer = first_prediction
else:
    final_answer = first_prediction
print(f"final_answer: {final_answer}")
```

2.1.3 方法优越性

1. 巧妙利用大小模型协同
2. 使用prompt让另一个LLM快速适应verify任务
3. 多重实验对比，得到性能提升

2.1.4 方法局限性

1. 没能自行训练verifier，与原论文有差异
2. fine-tuning model价格有点昂贵，没能尝试效果如何

2.2 实验结果

2.2.1 单条路径

通过单条路径验证结果，准确率60%.

```
[19]                                                                    Python
...  Inference outputs: To solve this problem, we don't need to calculate the exact time they will meet. Instead, we can fo

     Let's say Friend Q's speed is V km/h. Then Friend P's speed is 1.15V (15% faster).

     Since they start at opposite ends and walk towards each other, their combined speed is V + 1.15V = 2.15V.

     The distance they will cover before meeting is half the total trail length, which is 43 km / 2 = 21.5 km.

     The time it takes for them to meet (t) can be found using the formula:

     \[ \text{Distance} = \text{Speed} \times \text{Time} \]

     For Friend P, this would be:

     \[ 21.5 \text{ km} = 1.15V \times t \]

     But we don't need to find t; we want to know how far Friend P has walked. Since they both start at the same time, the

     \[ \text{Distance of P} = 1.15V \times t \]

     So, Friend P will have walked 21.5 km when they meet.

     The correct answer is B) 21.5.
     Inference outputs: B) 21.5.
     Yes.
     ...
     Inference outputs: C) 12.
     Yes.
     Epoch 1/1 - Accuracy: 60.00%
     Accuracies over epochs: [0.6]
     Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

2.2.2 多条路径

通过多条路径推理和验证，准确率70%，高于Part1中仅使用多条路径的准确率（50%）。

```
...  verify failed, reinferencing...
     Re-inference outputs: The question asks for the smallest number that is exactly divisible by 22, 33, 66, and 44. To find such a number, we need to find the least common multi

     First, we simplify the numbers:
     - 22 = 2 * 11
     - 33 = 3 * 11
     - 66 = 2 * 3 * 11
     - 44 = 2^2 * 11

     Now, we can see that all the prime factors are the same except for the power of 2 in 66 (which has an additional factor of 2). To make the number divisible by all, we should

     So, the LCM is \(2^2 \times 3 \times 11 \times 11 = 4 \times 3 \times 11^2 = 132^2\).

     Now, we need to convert this to a five-digit number. Since \(132^2 = 17424\) and we want the smallest five-digit number, we add the missing digits to make it 10,00
     final_answer: T
     selecting First,
     selecting The answer is after the proof.
     selecting Let's think step by step using common sense and knowledge.
     ['N', 'B', 'C']
     verifying answer...
     Yes.
     final_answer: N
     selecting Before we dive into the answer,
     selecting The answer is after the proof.
     selecting We should think about this step by step.
     ['C', 'C', 'C']
     final_answer: C
     Epoch 1/1 - Accuracy: 70.00%
     Accuracies over epochs: [0.7]
```

Part3: 使用Retrieval方法为思维链提供先验知识

3.1 实现原理

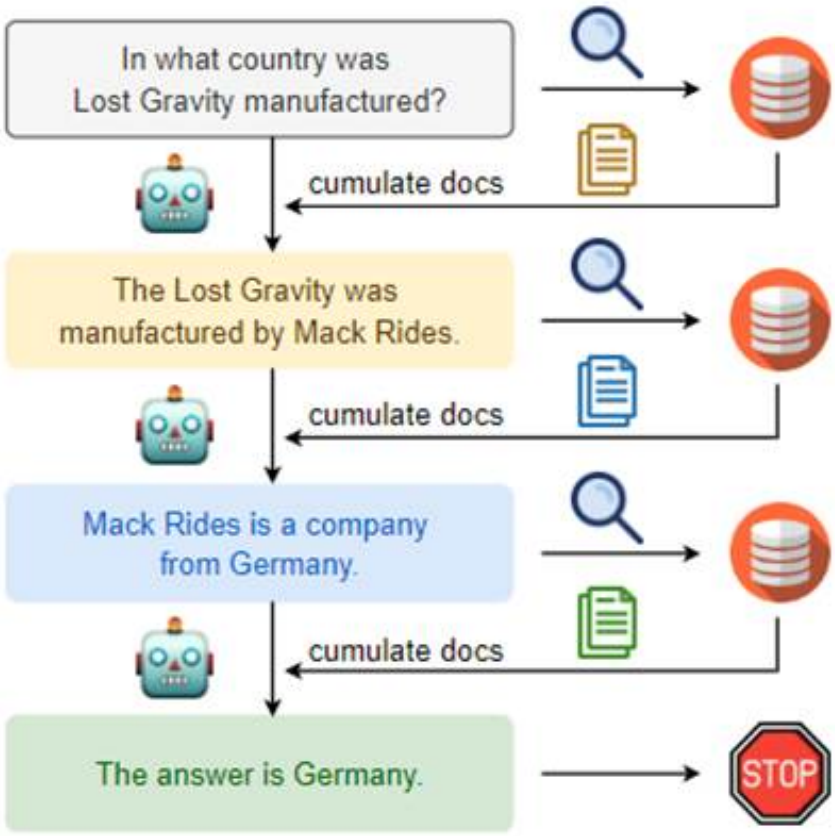
3.1.1 实验动机

在前面的Part1与Part2中，我们更多的是利用思维链的结构对生成过程进行优化。在这个过程中我们调用的大模型并没有实质上获得额外的知识。因此在Part3中，我们考虑在思维链中提供一些外部数据库中的先验知识，帮助大模型进行更好的思维推理。

3.1.2 Retrieval方法概述

Retrieval方法是大语言模型生成优化中一个非常热门的方法，在许多任务中都取得了非常出色的效果。它们大多用于解决常规的常识问题，例如"中国的首都是哪里"这类问题。就具体流程而言它的过程如下：1.将输入的问题传入检索器中。 2.利用检索器到外部数据库（如wiki等）中进行检索并将检索到的结果与问题以及生成器的输出（如果有）拼接为新的输入。 3.输入给生成器并产生对应的新的输出。 4.将输出的最后一个句子作为新一轮的检索，重复迭代执行1-4过程直至生成器的生成中出现"the answer is"等相关词或是达到最大值时终止该流程。

下图为该方法的一个形象性描述：



3.1.3 方法迁移及挑战分析

Retrieval的方法在常识问题中能够取得非常好的效果，且非常容易进行实现，这是因为常识类问题中往往**信息的逻辑性很低且有效信息（隐含知识）暴露非常直接**，通过简单的**文本匹配**即可得到较好的提示。而本次实验中，我们应对的是逻辑性要求极高的数学计算问题。这类问题它们的有效信息是**隐藏在深层逻辑当中**的，如一个数学题修改了数据后其有效信息其实没有变化，但是在文本表现上就会千差万别。这表明此类问题的有效信息可能需要经过若干轮的转换才能得到，也因此导致了普通的retrieval**检索难度极大**。如若仅使用普通的文本匹配可能未必能够

取得理想的效果，这给我们的方法迁移带来了极大的挑战。针对上述挑战我们进行了三个层次上的探索实验，**其中方法2、3相比于Part1、2中的方法取得了较大的提升**。

3.2 方法1——bruteforce

首先我们考虑使用最简单直接的生搬硬套，对于数学问题**Q1**我们利用搜索引擎，在外部网络数据库中进行检索，并选取相关度最高的k个（实验中k=5），作为问题**Q1**的先验知识。在与生成模型交互时，该先验知识会作为prompt进行输入。具体而言，在输入内容前添加prompt语句：`Here are some relevant information, may be you can first read them`。

3.2.1 问题分析

在方法1的实验过程中，我们对问题以及检索到的内容进行了输出，其中一个示例如下所示（由于篇幅原因我对result后面的一些内容进行了删减）：

```
问题：Q: Question: Two friends plan to walk along a 43-km trail, starting at opposite ends of the trail at the same time. If Friend P's rate is 15% faster than Friend Q's, how many kilometers will Friend P have walked when they pass each other? Choices: A: A)21, B: B)21.5, C: C)22, D: D)22.5, E: E)23
检索内容：Result 1: 【电大国开】国家开放大学23春人文英语1-8形考任务...
Result 2: 2023年浙江省高三名校协作体联考解析...
Result 3: 四六级听力真题练习Day15|备考|day|美文_网易订阅...
Result 4: 编英文对话:两个同学打算一起去旅行,从商量目的地到...
Result 5: 口国开《人文英语2》形考任务单元自测7答案...
.
```

我们可以看到该问题检索出来的结果非常不理想，可以说是**毫无关联**。我们认为其主要原因在于外部网站的数据信息**噪声过大**，有太多与数学问题毫无关联但是**在文本匹配角度相似度高**的信息。这会给使用基于文本检索的retrieval方法**引入众多无效知识**，带来极大的干扰。针对该问题，我们提出了方法2进行数据去噪的方法，即构建一个**精简版的database**，该数据库中的内容均为数学题，以此一定程度降低无关信息的干扰。同时，在retrieval过程中不再使用简单的问题匹配，而是**关注问题的相似度**，以此来衡量一个数学题是否作为先验提示。

3.3 方法2——精简database

在方法2中，我们首先构建了适合数学问题求解的数据集。数据集中包含多个数学问题的描述，以及其详细的解答（有详细的解答过程，以此为COT提供更好的知道）。保存格式为yaml文件，其具体格式如下：

```yaml
questions:
  - id: 1
    question: >
      Two friends plan to walk along a 50-km trail, starting at opposite ends of the trail at the same time. If Frien
    solution: |
      1. Let Friend Q's speed be v km/h. Then Friend P's speed is 1.2v km/h.
      2. Let d km be the distance Friend Q walks when they meet, so Friend P walks 50 - d km.
      3. Since they meet at the same time:
         d / v = (50 - d) / (1.2v)
      4. Cross-multiplying gives:
         1.2d = 50 - d implies 2.2d = 50 implies d = 50 / 2.2 ≈ 22.73
      5. Therefore, Friend P walks 50 - 22.73 ≈ 27.27 km.
```

完成数据库的构建后，我们考虑如何在新的数据库中进行高效的retrieval。注意到在现实的生活中数学题往往是有类型之分的，相同类型的题目它们的解题方法以及过程常常有异曲同工之妙。该相似性会为我们思维链的如何构建带来极好的引导。因此在检索时，我们考虑采用类型作为key来进行检查。考虑到我们资源的有限性（没法构造问题向量embedding进行向量相似度检查），我们借鉴了Part2中verify的思想，**利用另一个大模型作为我们的相似度评价器（verify）来进行类型相似度检查**。具体而言，对于问题s1，我们遍历数据库中的所有问题s2，构建输入：`Is it possible that the solution's method for problems {s1} and {s2} will be similar (regardless of whether they are both multiple-choice questions)? Just reply me with yes or no`，交给大模型LLM1来获得数据库中的问题s1与实际问题s2的相似度。对于回答为yes的问题，我们进行拼接，最后将拼接结果作为prompt交给生成模型LLM2。

该部分核心代码如下（其余部分几乎同baseline）：

```python
def similar(s1,s2):
    verification_prompt = f"Is it possible that the solution's method for problems {s1} and {s2} will be similar (regardless of whether they are both multiple-choice questions)? Just reply me with yes or no"
    verification_result = call_with_messages('qwen-14b-chat', verification_prompt)

    if verification_result:
        return "yes" in verification_result.lower()
    else:
        print("Error in getting verification output.")
        return False

import yaml

def search_similar_problem(s):
    with open('database.yaml', 'r') as file:
        data = yaml.safe_load(file)

    result=""
    for item in data['questions']:
        s1=item['question']
        solu=item['solution']
        if (similar(s,s1)==True):
            result+=f"Q:{s1}. A:{solu}. \n"
```

3.3.1 问题分析

在实验的过程中，我们发现虽然我们对外部数据库进行了有效的精简，但是在retrieval的返回中仍然会有部分与当前问题无关的信息被返回，这会导致**prompt过长**，大模型的生成能力一定程度上会受到这种长上下文的影响。导致该问题发生的主要原因还是在于**相似度评价器（verify）并不能够做到很准确**，如果存在一个专家模型可以高效地对要进行retrieval的所有内容进行筛选并返回，我们有理由相信性能会更好地提升。但是查阅了一些论文，我们目前没有搜到较好的对于数学问题的解决方法。考虑到对于中学数学问题而言，我们人本身其实就是一个训练非常完善的“专家模型”，故考虑使用**Human in the Loop**方法即将人作为retrieval系统中的一环，对数据库信息进行筛选，选择与当前问题相关度最高的k个作为返回。

3.4 方法3——Human in the Loop（upper-bound）

在方法3中，我们对retrieval过程进行更加进一步的优化。具体而言，将verify模块替换为Human，将人作为retrieval系统中的一环，对数据库信息进行筛选，选择与当前问题相关度最高的k个作为返回。在实验结果中，我们可以明显地感受到该方法带来的准确率的提高，我们也有理由相信在不提升模型性能的基础上，该方法可能是该问题的上界了。

代码改进部分如下：

```python
def test_useHumanInTheLoop():
    correct_count = 0
    total_count = 0

    for batch in aqua_dataloader:  # 修改为aqua_dataloader
        inputs, labels = batch

        for question, true_label in zip(inputs, labels):
            print(f"true_label:{true_label}")
            trigger_sentence = trigger_sentences[0]
            prompt=HumanInLoop(question)
            X0=f"I will give a sample: {prompt}. Please solve follow promblem. Q: {question} A: {trigger_sentence}"
            Z = inference(X0)
            final_prompt = f"{X0} {Z} Therefore, among A through E, the answer is(only one alphabet, no need to write the whole word, e.g., 'A' or 'B' or 'C' or 'D' or 'E')"
            print(final_prompt)
            final_answer = inference(final_prompt)
```

3.4.1 局限性分析及改进展望

虽然方法3取得了非常不错的效果，但是该系统中有了人的参与，这使得其与人工智能的初衷相违背。不过随着大模型的飞速发展，我们有理由相信上述系统中的Human部分未来可以用一个高效的专家模型来进行代替，该模型可以拥有与人一致甚至更强更准确的筛选能力。这也可以成为一个很不错的研究方向，鉴于我们大程的时间有限，便没有继续在这个方向上进行探索，而是给出这个我们认为的当下的理论上界。

3.5 实验结果

3.5.1 方法1

对于方法1，我们取得了60%的准确率

```
Friend P's distance = T * (1.15 * r)

Since they meet when the sum of their distances equals half the trail length:
Friend Q's distance + Friend P's distance = 21.5 km

Substituting their respective distance equations,
T * r + T * (1.15 * r) = 21.5 km

Combining the terms with 'T' and 'r':
T * (r + 1.15 * r) = 21.5 km

T * (2.15 * r) = 21.5 km
...

Therefore, the only possible answer is C) 12, with X being 1 and Y Therefore, among A through E, the answer is(only one alphabe
prediction:C
Accuracy: 60.00%
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

3.5.2 方法2

对于方法2，我们取得了80%的准确率（高于Part1、Part2中的任何方法）

```
3. Substituting point (6, y):
   y = 2/3 * 6 = 4
4. Therefore, x and y are 3 and 4 respectively.
Answer: D: 6 and 4
.
Q:For all numbers p and q, the operation @ is defined by p@q = p^2 + pq. If xy ≠ 0, then which of the following can be equal to
. A:1. x@y = x^2 + xy
   To be zero:
   x^2 + xy = 0 implies x(x + y) = 0
   Since xy ≠ 0, x ≠ 0 and x + y = 0 implies y = -x. So, I can be zero.
...

Therefore, a possible product of X and Y is 12, making answer C correct. Therefore, among A through E, the answer is(only one al
prediction:C
Accuracy: 80.00%
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

3.5.3 方法3

对于方法3，我们取得了90%的准确率（非常之高，我们有理由认为在不改变LLM模型性能的情况下这个可能会是一个理论上的upper-bound）

Friend Q's distance is:
Friend Q's distance = Q's speed * t = Q * t

When they meet, the sum of their distances is equal to the total trail length:

Friend P's distance + Friend Q's distance = 43 km
∴ 1.15Q * t + Q * t = 43 km

Combining like terms and solving for t:

(1.15+1)Q * t = 43
2.15Q * t = 43

Now, let's find t:
...

Therefore, the only possible product of X and Y among these options that would make 8XY a divisible-by-2 three-digit number is C)
prediction:C
Accuracy: 90.00%

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...