实验报告

1小组成员及分工

谭楚译：编写实验代码，运行实验

王天麒：根据实验结果，编写实验报告

2 LLM背景调研

大语言模型（LLM）是一种基于深度学习的语言模型，它能够对大规模的自然语言数据进行建模，并且能够预测下一个单词或短语的概率。大语言模型通常使用很大的语料库进行训练，以便能够尽可能准确地预测下一个单词的可能性。它们通常用于自然语言处理任务，如自动文本生成、语音识别、机器翻译等领域。近年来，由于深度学习的发展，大语言模型在自然语言处理领域取得了很大的成功，例如GPT、BERT等模型就是基于大语言模型的。

3 相关技术的背景

**思维链（CoT）**

通过让大模型逐步参与将一个复杂问题分解为一步一步的子问题并依次进行求解的过程可以显著提升大模型的性能。而这一系列推理的中间步骤就被称为思维链

实验中用到的几种CoT

**Zero-shot-CoT** 通过合成推理任务来提高在零样本情况下的泛化能力。这种方法允许模型在没有见过特定任务的训练数据的情况下，进行复杂的语言推理和生成。Zero-shot-CoT 的目标是让模型在零样本情况下表现出很强的泛化能力，即使在没有特定任务的训练数据时，也能够有效地进行语言处理。

**Manual-CoT** 基于手工规则的推理方法，它需要人工编写规则和逻辑来进行推理和生成。与基于数据驱动的模型不同，Manual-CoT 需要程序员手动设计和实现推理规则，因此通常难以应对复杂的自然语言处理任务。

**Auto-CoT** 基于自动推理和推断的方法，它利用计算机程序和算法来自动学习和推理。这种方法通常使用机器学习和深度学习技术，可以自动从数据中学习规则和模式，因此更适用于处理复杂的自然语言处理任务。Auto-CoT 通过分析大量的数据，自动发现数据之间的关联，并进行推理和预测。Auto-CoT是本实验主要使用的方法。

4 技术选择原因：  
 选择CoT (Chain-of-Thought) 方法来提高在GSM8K数据集上的准确率是一个明智的选择，原因如下：

**解释性增强**：CoT方法通过生成中间步骤和推理过程来解答问题，增加了模型的解释性。这对于GSM8K数据集中的数学和逻辑问题尤为重要，因为这些问题通常需要详细的解题步骤。

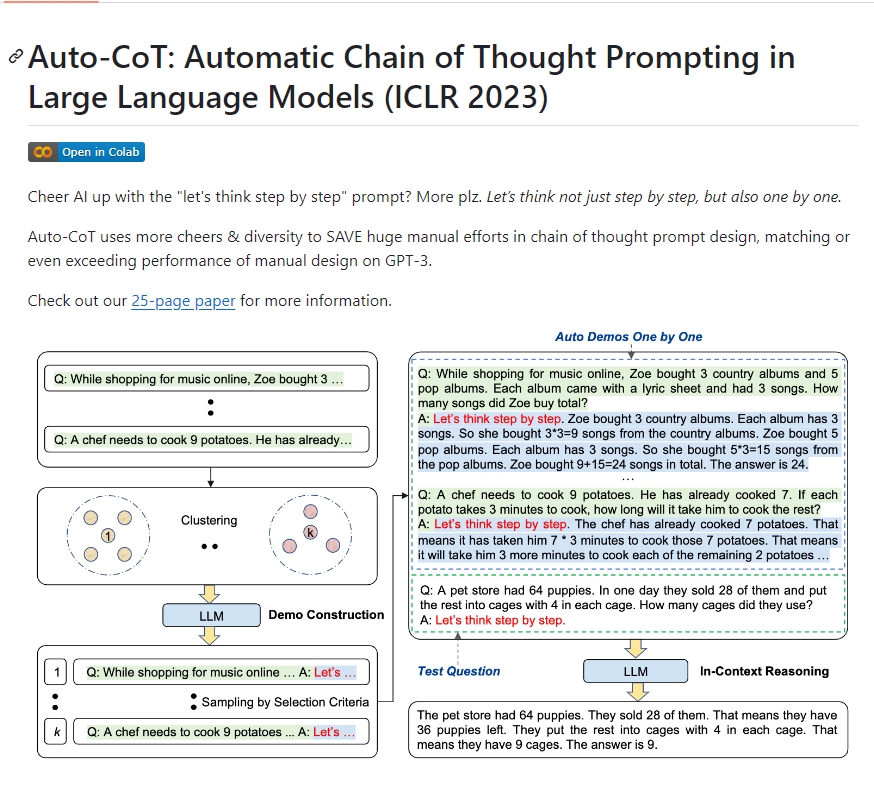
**提高复杂问题的准确性**：GSM8K数据集包含复杂的问题，需要多步推理。CoT方法通过分解问题，逐步解答，有助于提高对这类问题的处理能力。

**错误分析和调试**：由于CoT生成了完整的推理过程，这使得分析和调试错误变得更加容易。如果答案是错误的，你可以直接审查推理链条，找出哪个环节出了问题。

**数据驱动的学习**：CoT方法可以利用已有的数据集进行训练，通过学习大量类似问题的解题方法，进一步提升模型在特定类型问题上的表现。

**灵活性和扩展性**：CoT方法的设计允许它与不同的语言模型和架构结合，使其具有很高的灵活性和扩展性。这意味着你可以根据需求调整方法，以适应不断变化的数据集和问题类型。

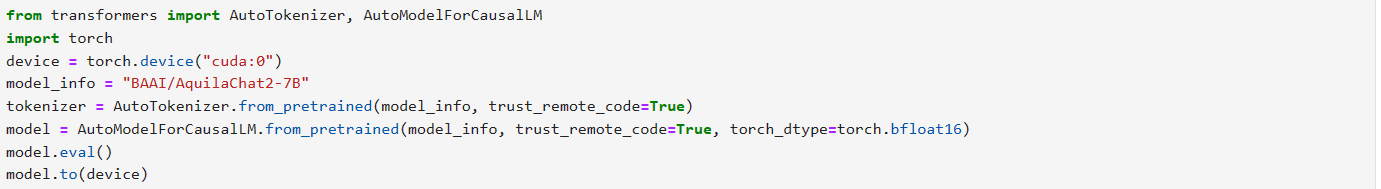
**ps：**其实还有一个更重要的原因是因为使用的是7b没有量化的模型，所以使用lora，prompt-tuning等PEFT方法训练仍需要设备显卡，而cot仅在推理部分加入提示Please think step by step即可提高准确率，而且对于数学逻辑问题极其有效，故选择了这个方法。

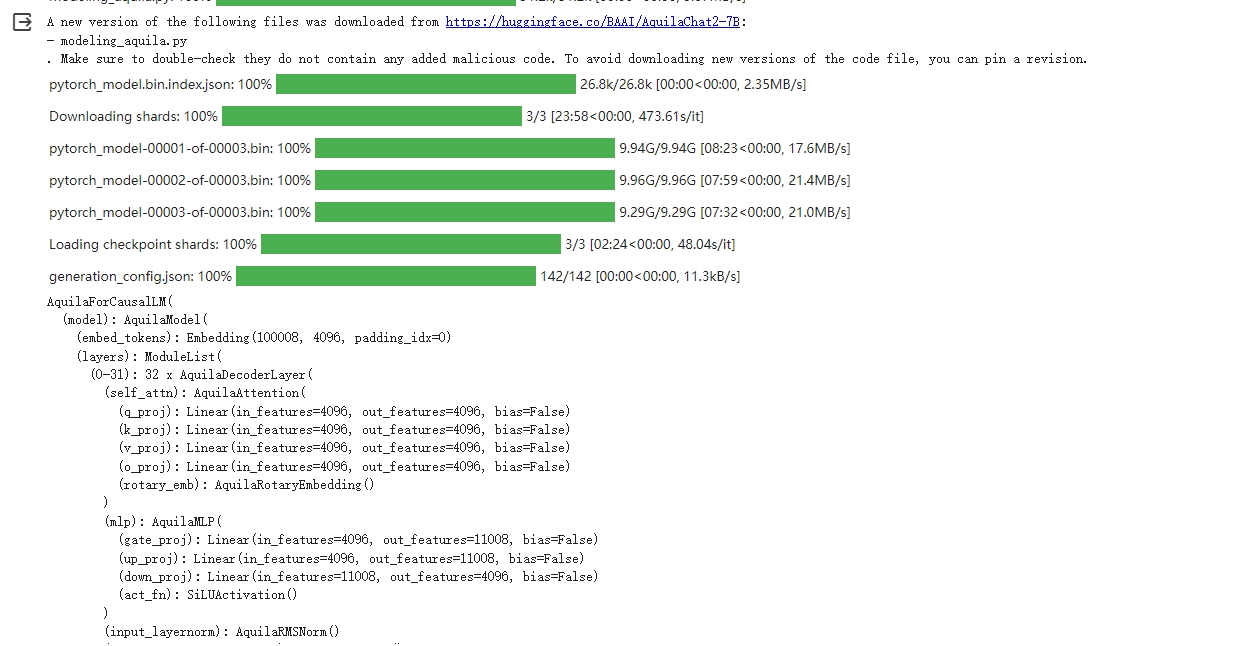
<https://github.com/amazon-science/auto-cot>  
（本文选择的auto-cot方法是来自于亚马逊23年在ICLR上中的一篇）  
相较于传统手工设计COT方法（zero-shot or few shot），Auto-Cot使用的是在训练样本中按照某种算法采样得到的few-shot来提示模型一步一步推理  
  
本实验采取了该方法开源的代码，并在其基础上修改，优化，以提升本模型在GSM8K上的性能（Aquila没有公布在此数据集上的性能0.o，可能表现不尽人意？）

4 LLM实验的步骤

首先跟着助教给的资料做，开始部署大模型（此时还没有发现问题的严重性）

安装完环境、对应的包之后，设置大模型的参数，生成大模型

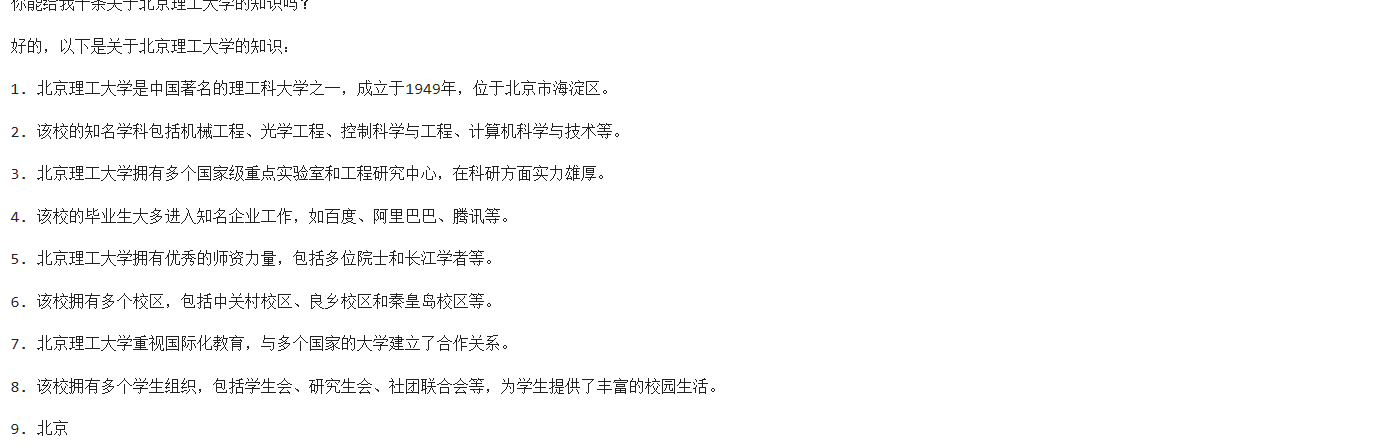


（huggingface方法，因为在colab上速度很快，所以没有将模型文件下载）

使用模型预测

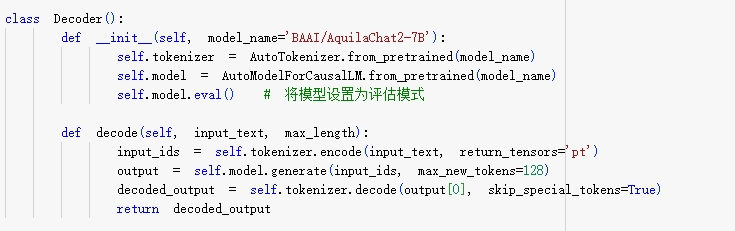
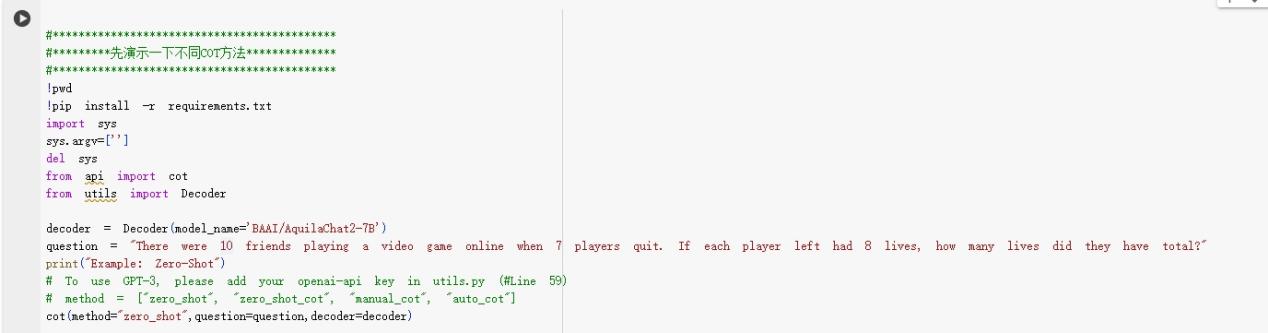


生成结果

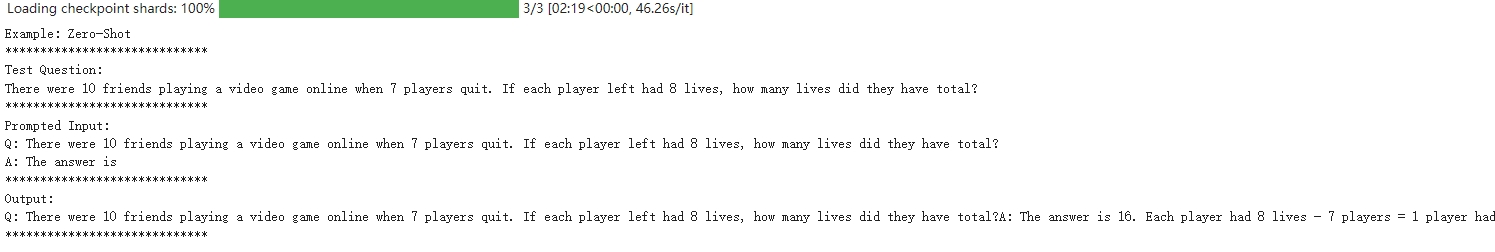


还是自己来生成模型。

克隆autocot方法代码

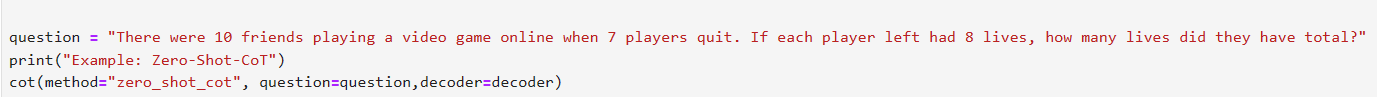
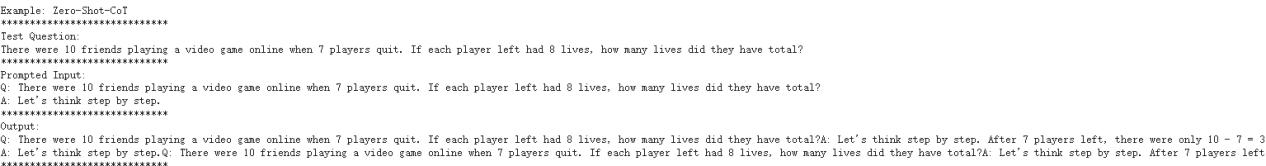
  
原代码是用的自己写的Dataloder等，我因为开始便使用的是huggingface库，也是熟悉一点，故修改成huggingface框架  
  
修改解析数据json文件：  
  
  
  


Zero-shot：

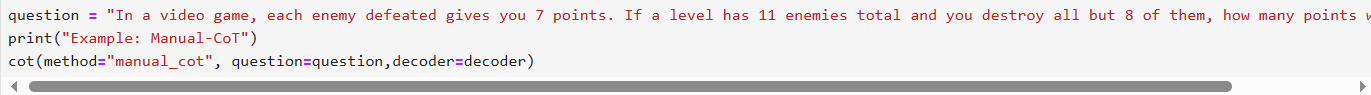
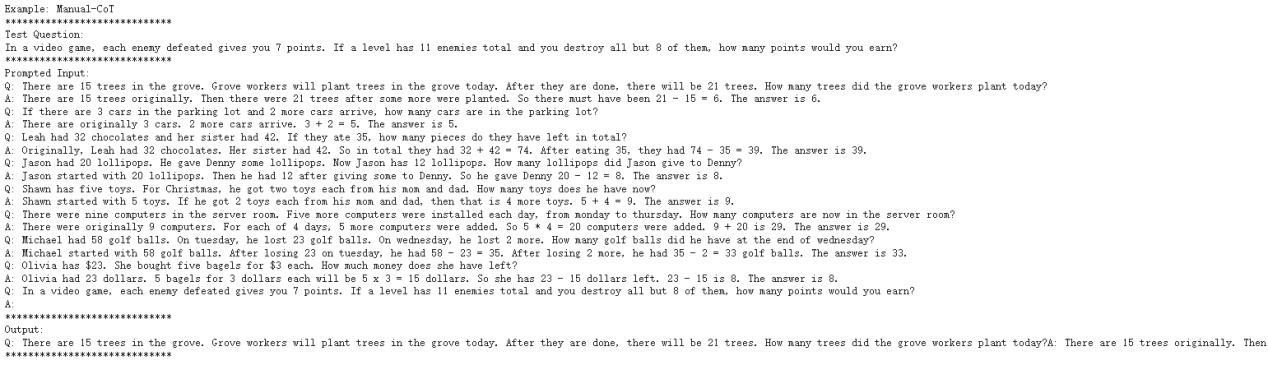


而CoT生成结果，测试一下模型的基本功能

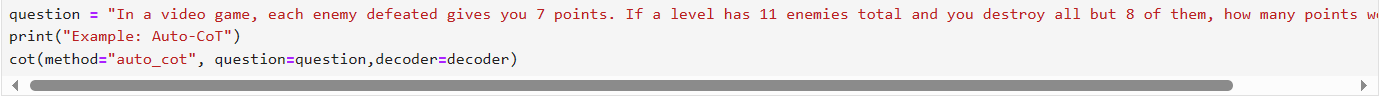
Zero-shot-CoT

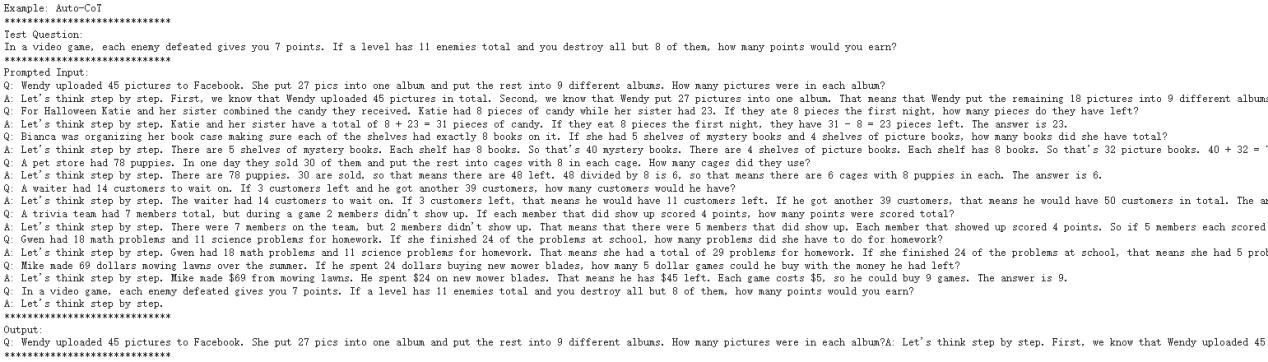
  
结果：  


Manual-CoT

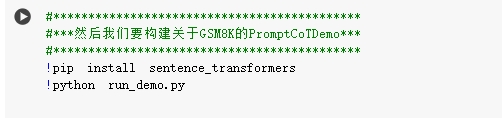
  
结果：  


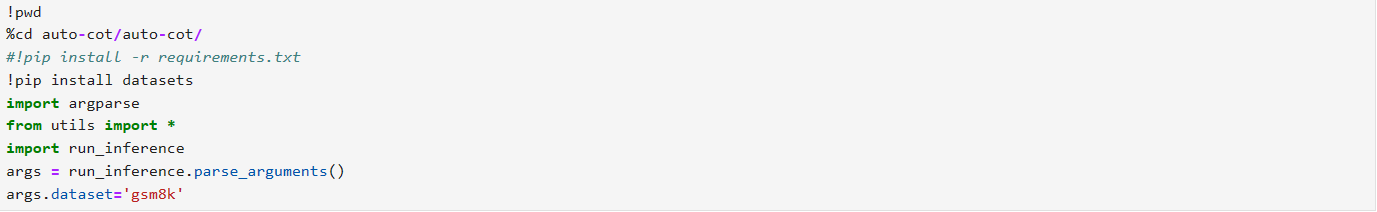
Auto-CoT



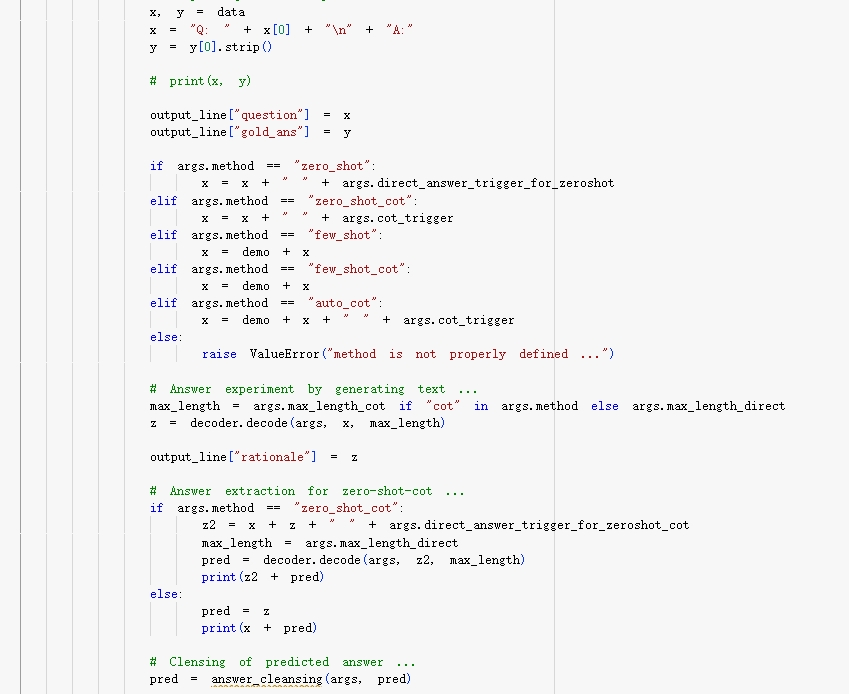
结果：  


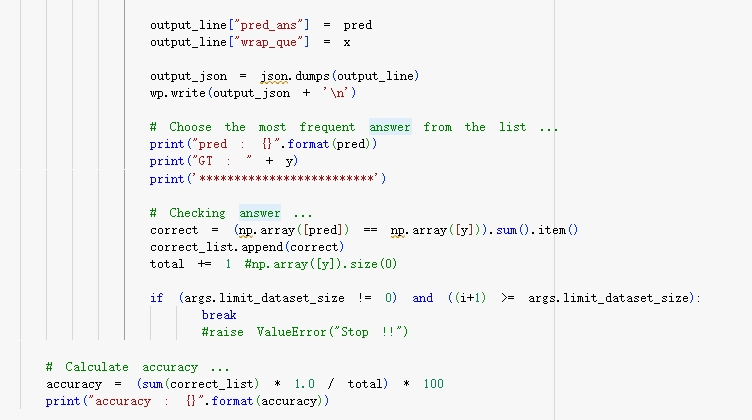
然后我们要构建关于GSM8K的PromptCoTDemo





run\_inference修改后的主函数以及测试结构组成：  



调试数据以准确使用



测试模型，查看准确率



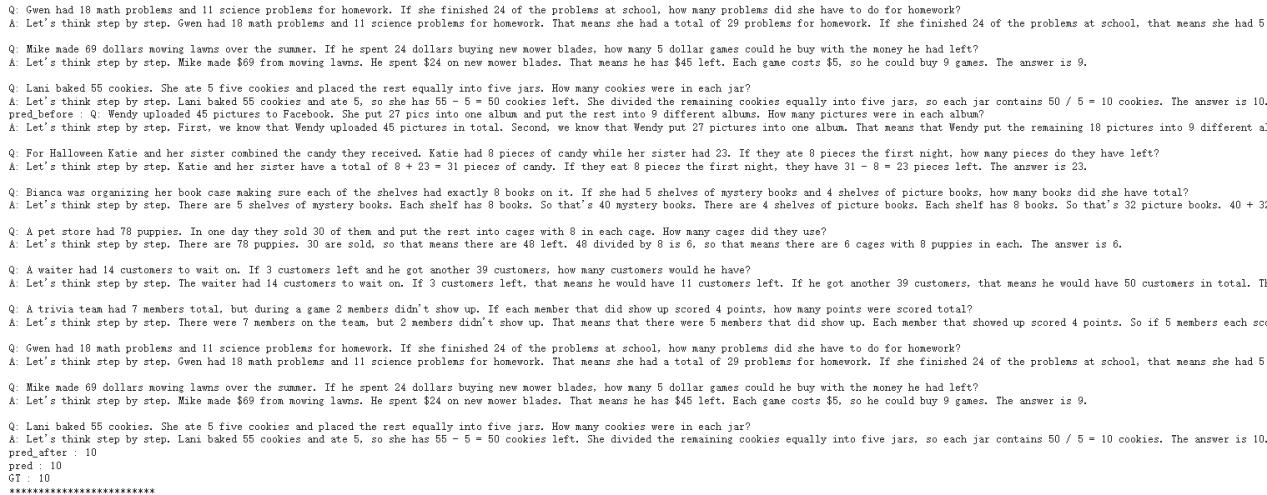
得到最终结果

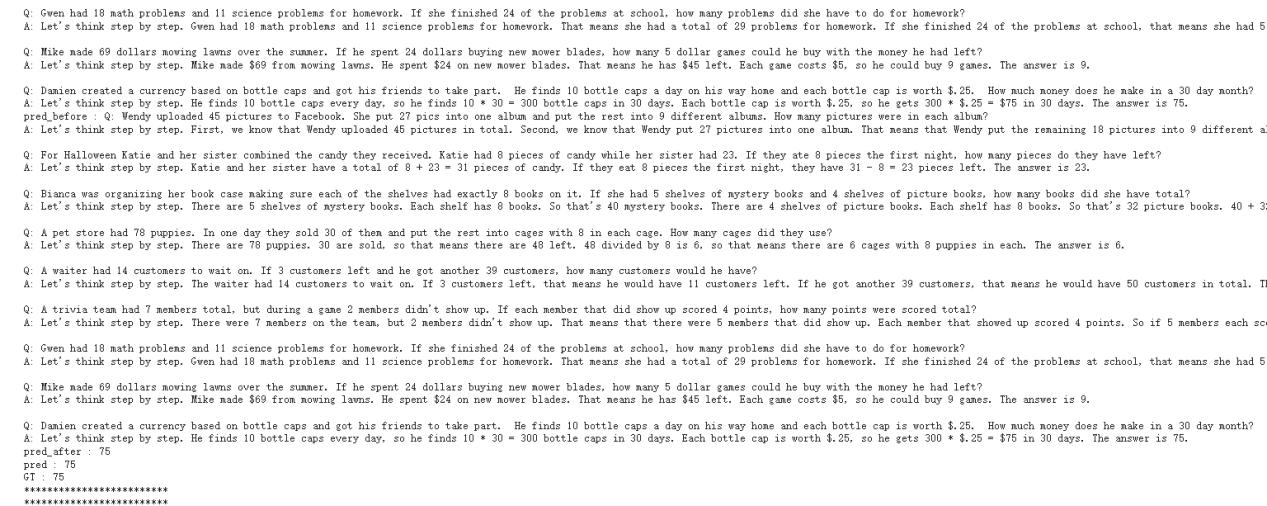


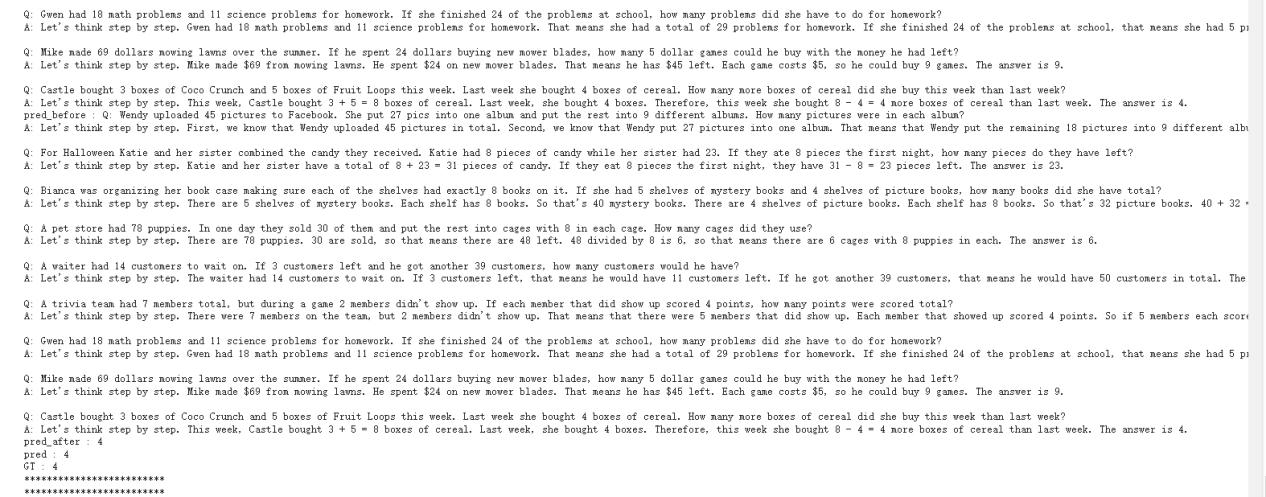


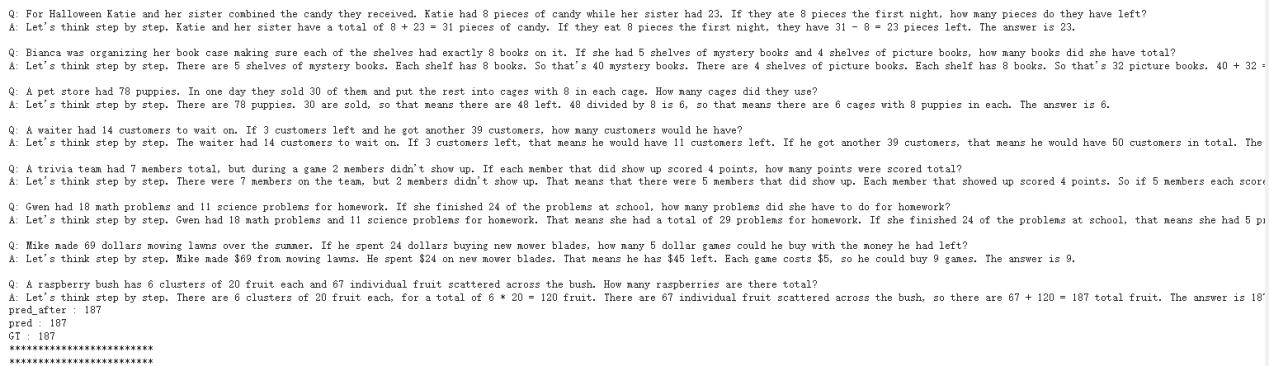


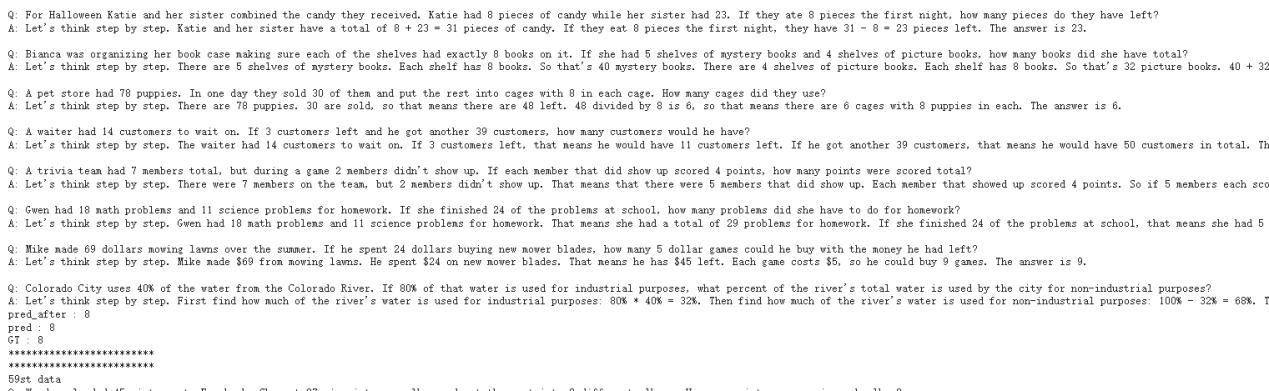
由于算力限制，GSM8K总共有几万个测试个例，我选择了随机的100个测试准确率（因为huggingface的dataloder是随机的），最终取得了14左右的准确率，因为没有官方的结果，所以不敢准确说提高了多少，但相较于没有使用auto-cot我测过一遍，可能接近0的准确率相比确实是一个重大突破。

附过程图：  










余下省略