Prompt Summarization for Program Synthesis

Kirby Kuznia

1 Project Organization

The GitHub repo contains 4 different directories: competitive, interview, introduction, and src. These are the three levels of difficulty defined in the APPS dataset, along with some scripts to help automate tasks.

The following scripts are provided. All scripts should be run from within the src directory.

- download.sh - This script will download the dataset, extract the files, then call the prepare.sh script.
- prepare.sh - This script will separate the dataset into the 3 difficulties.
- format.sh - This script will clean the four required files.
- view_solution.sh - This script will print a code solution for any given problem.
- solution.py - Used in conjunction with the bash script to print a code solution.
- test.sh - This script will test if all requried files exist and is run for every pull request.

The *download.sh* script will create a directory called *APPS* in the main directory. Any changes in this directory are ignored and not tracked by git. The *APPS* directory contains the original prompts and after the *prepare.sh* script is run, it will contain 3 directories introductory, interview, and competition.

2 How to Start

You should copy a problem from the *APPS* directory into the main directory under the corresponding difficulty. From there you need to create 3 files, *summary.txt*, *expert.txt*, and *instruction.txt*. The sections below provide a detailed description on how to create those files.

At the start each problem contains 4 files.

- question.txt - This is the prompt.
- input_output.json - These are the test cases.

- solutions.json - These are possible solutions.
- metadata.json - This is the difficulty.

You should read the *question.txt* file to understand the problem then follow the sections below to create the required files.

After you are done make sure each problem has the following 4 files: *question.txt*, *summary.txt*, *expert.txt*, and *instruction.txt*. Then you should run the $format.sh$ script. This will clean the text and create a one line file for each .txt file. So you can format your summarizations any way you want, only the cleaned versions will be passed to the model. Keep the uncleaned versions to help with readability for your pull requests.

3 Summarization

Create a file called *summary.txt* this will contain your summary of the prompt. It's recommended that you copy the *question.txt* file into the *summary.txt* file then starting from the top of the prompt follow the steps and remove words/lines as necessary.

These are the rough steps for making a summary. Following these steps will create the most consistency in our dataset. However, you should summarize as you see fit. First, read through the prompt and understand what it's asking, then follow these steps to help create a summary.

1. Directly state what is given in the problem.
   - Most problems start by setting the scene, to help humans understand.
   - Start the problems by explicitly telling the model what the input is.
   - *You are given ...*
2. Remove any *notes* given in the prompt.
   - They are usually reemphasizing points, which is redundant and not needed in the summary.
   - This includes the $-Notes-$ section at the bottom of the file.
   - If there is pertinent information given from a note, include it in the prompt without describing it as a note.
3. Remove any text in parenthesis.
   - Most of the text in parenthesis is repeating the information that precede them.
   - If the text in parenthesis provides more context or information, then remove the preceding text.
   - Keep any parenthesis if it is describing constraints, such as the minimum and maximum values for the input etc...

4. Remove any made up people, places, things, etc...
    - These abstractions are made to help humans understand but confuse the model.
    - The prompts often mention things like *Codefortia* or *Polycarp*, try to replace these with the word *you*.
    - Any text visualizing what the problem is asking, should be removed.
5. If the *Input* or *Output* section reference an abstraction they should be changed.
    - Overall, these sections are fine. However, if they mentioned something you removed in the previous steps, they should be changed to reflect that.
    - If these sections repeat themselves remove any redundancies.
    - In most cases these sections will be left alone.

4 Expert Summary

Create a file called *expert.txt* this will contain an expert summary of the prompt. It's recommended that you copy the *summary.txt* file into the *expert.txt* file then starting from the top of the prompt remove words/lines as necessary. You should aim for the expert prompt to be $2-4$ lines.

Imagine you are describing the prompt to a senior software engineer. What else could you trim out? The difference between the original and expert summary, is the original summary may include something obvious, whereas the expert solution should be the absolute bare minimum. To create *summary.txt* you want to remove superfluous details from the original prompt. To create *expert.txt* you want to remove details that an expert would find obvious, from the summary.

For example, in problem 2000 (which is competitive difficulty) the summary mentions '*It will be possible to travel between each pair of nodes ..., and the sum of times ... will be the minimum possible*'. This process is describing a minimum spanning tree so you can just say '*Find a minimum spanning tree*'.

Also, if the prompt included an example and subsequent explanation, that should remain in the summary but should be removed from the expert summary. An expert already understands the problem and does not need any extra explanation. You should still keep the $-$*Examples*$-$ section.

5 Instruction Summary

Create a file called *instruction.txt* this will contain a description of the actual implementation. You will be creating this prompt from scratch. You should still include the $-$*Input*$-$, $-$*Output*, and $-$*Exampels*$-$ sections. However, the prompt will be different.

First, you can use the *view_solution.sh* script to view a code solution to the problem. The script is used as follows *view_solution.sh problem_num example*. Where problem number is the number you want to view the solution for and example is the index of the solution you want to see (some problems provide multiple code solutions). You should look at this code, then describe what that code is doing.

The purpose of this summary is to see if the model is able to extract the logic of the problem based on the prompt. Whereas, the other summaries test to see if the model can summarize the significant points.

6 Contributing

When you are ready to commit your changes follow these steps:

1. Make sure the three different summary files have been added to the problem directory.
2. Run the *format.sh* script.
3. Submit your changes through a GitHub Pull Request.
4. Respond to any comments on the Pull Request if necessary.

Thank you for your contributions and happy summarizing!