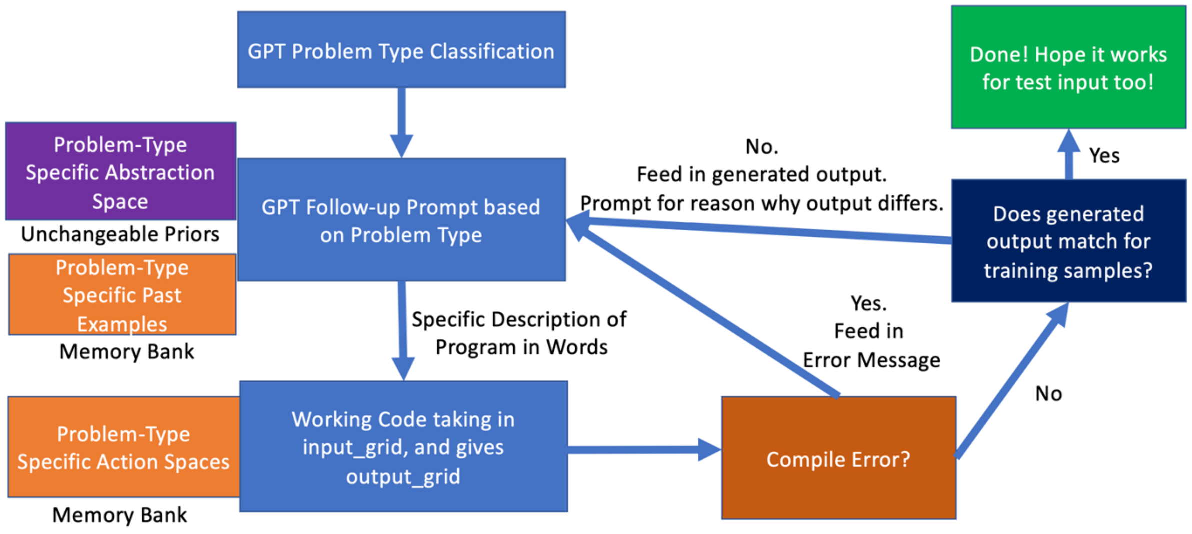
LLMs as a System to solve the Abstraction and Reasoning Corpus (ARC) Challenge

26 Jun 2023

John Tan Chong Min

Summary:
Using the flexibility of Large Language Models (LLM) to be prompted to do various novel tasks using zero-shot, few-shot, context-grounded prompting, we implement LLM (specifically GPT4) as a system with memory and recursive environment feedback loop, as an attempt to solve the ARC challenge. GPT4 has been shown to be able to perform an association between the input space to the action space relatively well (e.g. Voyager/Ghost in the Minecraft), and so we can frame the ARC challenge as a problem to find the actions given the input/output associations and utilize the generalizability of GPT4 to solve it!


GPT Problem Type Classification
Problem-Type Specific Abstraction Space
Unchangeable Priors
Problem-Type Specific Past Examples
Memory Bank
GPT Follow-up Prompt based on Problem Type
Specific Description of Program in Words
Problem-Type Specific Action Spaces
Memory Bank
Working Code taking in input_grid, and gives output_grid
Compile Error?
Yes. Feed in Error Message
No
Does generated output match for training samples?
No. Feed in generated output. Prompt for reason why output differs.
Yes
Done! Hope it works for test input too!

Details:

Useful Abstraction Spaces: One thing which GPT4 does not do well is to identify objects accurately from text alone. Objects are defined as continuous sections of the grid with the same non-zero value. Providing such an object view as an abstraction space greatly helps with GPT4 to form associations with the input-output pair and is better able to find a solution. Providing various abstraction spaces such as object view and section view (multiple sections of the grid to overlay) to GPT4 can greatly reduce the complexity of the problem.

Encoding Human Biases via Primitive Functions: An initial implementation of using GPT4 to solve ARC was done with just prompting the human biases and action spaces via text. This did not do so well due to lack of grounding using words alone. A key innovation was the usage of primitive functions as action spaces, as a way to encode human priors. If we could use functions for grounding, and express the semantic meaning of the function in words, GPT4 could use the function to provide the code needed for the solution. Hence, the problem now becomes finding out what are the primitive functions we need to encode in order for GPT4 to solve any generic ARC problem.

Recursive Feedback Loop from Environment: Another key idea is that a learning system would need to utilize feedback from the environment, and so a recursive loop feeding in feedback from the environment (whether the code matches the intended output) can help a lot in getting the right answer.

Using Memory for Additional Context: New problems might mix and match aspects of previous solutions, so having a memory bank to provide some examples of the code for similar solved problems in the past can help to ground GPT4 to better generate the answer.

LLMs as a System: Humans do not operate with only one system. We have various systems to call for various tasks. Similarly, we can have multiple expert agents for each task, like object-level, pixel-level, section-level and call on them to come up with their view of the task, and select the most promising agent (*GPT Problem Type Classification*). This greatly helps narrow the search space for the solution. Then, we utilize the specialized functions this agent has and solve the problem (*GPT Follow-up Prompt*). Interfacing this agent with environment feedback, the problem-type specific abstraction space, past examples and action spaces can greatly help filter and ground GPT4 to generate a plausible solution. I believe that with better grounding via expert agents, and with better abstraction space representations and better primitive function grounding, we will eventually be able to solve most of the ARC tasks using this system.

<u>Extra Details:</u>
Conditional Code: Rather than letting GPT free-form generate its own code, we ask it to use the primitive functions, but additionally, let it use one or more conditions to create a conditional flow. Such a conditional flow is needed, as for the ARC challenge, some problems require logic that only apply if a particular condition is met (e.g. turn the shape red if it has exactly 6 cells). Without this conditional flow, the program would need many more steps before it can solve the problem. Example:
If {condition}: {Primitive Function}

Please see https://www.youtube.com/watch?v=plVRxP8hQHY for more information.