Carnegie Mellon University

Pragmatic Neural Code Generation

Raunak Sood, Saujas Vaduguru, Daniel Fried

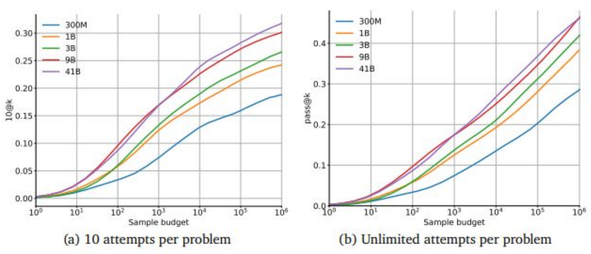
Introduction

Problem

- LLMs have shown remarkable success in text generation tasks
- They can also be used to generate programs from natural language descriptions
- Such applications have shown productivity gains among software developers

Challenges

- Selecting a single program from a set of possible model generated solutions is difficult



(a) 10 attempts per problem (b) Unlimited attempts per problem

Figure 1: Plot of 10@k and pass@k score against k using the AlphaCode family of models.

- It often takes several attempts for a model to generate the correct solution

Solution

- First, we sample programs and test cases via Listener and Speaker models (LLMs)
- Then, we use Bayesian inference to select the most informative programs based on the evaluation of the programs on the tests
- Finally, we use the CodeT algorithm to select a single program based on clustering programs with similar functionality
- This approach is promising because the two stacked approaches use different heuristics to filter programs

Methodology

CodeT

- CodeT scores generated programs based on the number of programs that pass the same test cases it does

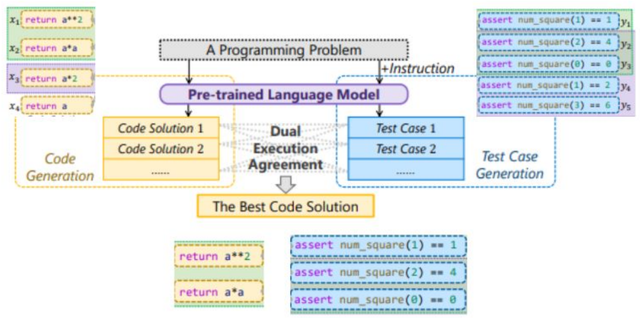


Figure 2: CodeT algorithm. An LLM is used to generate both programs and test cases, and the programs are re-ranked based on execution agreement

Pragmatic Inference

- Programs are selected by the Rational Speech Acts (RSA) procedure, which views pragmatic inference as a game between the Listener and Speaker models where they try to predict each other's intentions

$$S_1(\text{test} \mid \text{program}) \propto L_0(\text{program} \mid \text{test})\, P(\text{test})$$

$$L_1(\text{program} \mid \text{test}) \propto S_1(\text{test} \mid \text{program})\, P(\text{program})$$

Pipeline

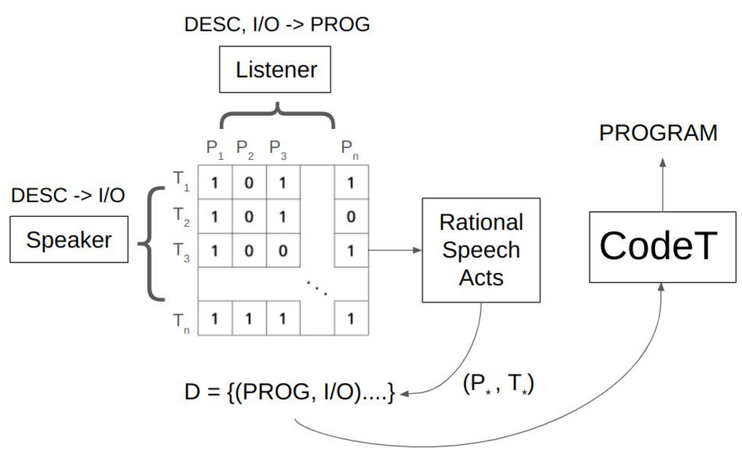


Figure 3: Pipeline for stacking our pragmatic inference procedure with CodeT.

Preliminary Results

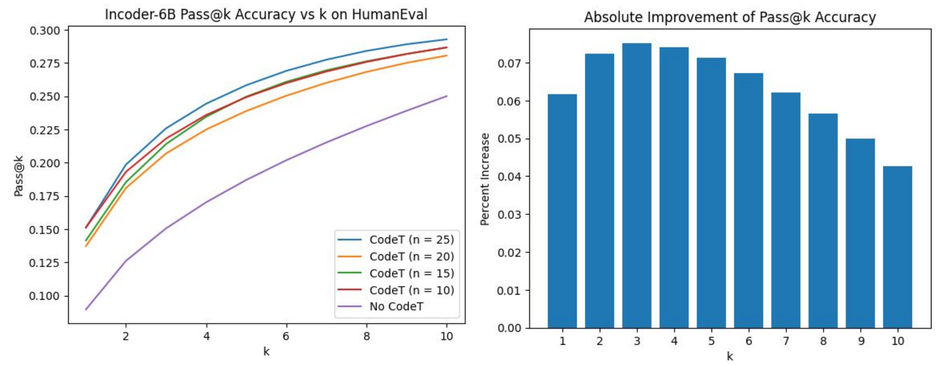


Figure 4: Our initial results comparing programs generated with and without CodeT using Incoder-6B. We evaluate the pass@k accuracy over several values of k and determine the absolute improvement over the baseline.

- We observe a 6% increase in the accuracy of the first attempt using CodeT
- As we increase the number of samples CodeT is allowed to use, we see a general improvement in the pass accuracy
- From previous research, the pragmatic inference procedure has a similar trend in improvement

Future Work

To Do

- Implement the RSA procedure and finish the pipeline
- Evaluate the pipeline on different benchmarks, comparing it to existing methods
- Try other execution based methods such as MBR-Exec instead of CodeT

Ideas for Future Work

- If we see improvements in pass accuracy, we might be able to generate an informative data set
- We can then use this data set to fine-tune the Speaker and Listener models and observe if they learn to generate more accurate programs

Contact
rrsood@andrew.cmu.edu