Repository-Level Prompt Generation for Large Language Models of Code
Disha Shrivastava*, Hugo Larochelle, Daniel Tarlow

Introduction

Motivation
- Black-box access to LLMs: strongest models not publicly available, e.g., no access to model weights for Codex [1] that is deployed in GitHub Copilot[2].
- Incorporating the repository info: structure and context from other files.
- Example-specific discrete prompts: easy to plug-in human domain-knowledge, easy control.

Repo-Level Prompt Generator (RLPG)
- Learns to generate example-specific prompts without requiring access to the model weights.
- We propose a set of repo-level rules. A rule consists of (i) rule context location, (ii) rule context type, (iii) rule context ratio, e.g., get method names and bodies from first import file and fill 50% of the prompt space with this context (see below).

Methodology

Dataset: Java repositories from Google Code archives[3]
Preprocessing: Deduplication, Parsing the file level AST and collating repo-level meta-info
Methods:
1. Codex: default Codex context.
2. Oracle: use the ground-truth vector that indicates success for each rule per example.
3. Fixed Rule: using a fixed rule for all examples.
4. Rule Classifier: Use a learned model to select the next rule conditioned on the example. Modeled as a multi-label binary classification task.
   - RLPG-H: use the hole context
   - RLPG-R: use the similarity of the hole context with the rule context.

Prompt Generator: Concatenate the default Codex context with the selected rule’s context in the rule context ratio.

Experiments and Results

- Performance of the oracle
- Performance of different methods averaged across all holes (hole-wise) and individual repositories (repo-wise).

Conclusions

- An oracle constructed from our proposed rules gives 36% relative improvement over Codex.
- When we use our rule-classifier to select the best rule, we get 17% relative improvement over Codex. RLPG also better than fixed rule.
- Future Work: Composition of rules and human-in-the-loop prompt generation.

References:
[1] Chen, Mark, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards et al. "Evaluating large language models trained on code." arXiv preprint arXiv:2107.03374 (2021).
[2] https://github.com/features/copilot/
[3] https://code.google.com/archive/