Exploring LLM-Assisted Code Commenting On Class-Level Code

Team Name: TheCommentors
Pranav Hariharane pkh2120
Richard Cruz-Silva rsc2174

What I Want to Study

For my project, I want to evaluate how well modern code LLMs can generate comments on realistic software engineering code. This project builds upon my midterm paper where I explored the performance of LLMs in code generation and the potential for LLMs to achieve good performance in realistic software development scenarios. In my literature review and experiment, one of the main weaknesses I noticed for LLMs in code generation performance was the inability to properly reason about the more complex and difficult programming problems. That's why I chose to explore how well LLMs can generate comments for realistic software development code since I believe that may provide some insight into how well LLMs can reason about code and all the contextual dependencies involved in it. These contextual dependencies include class fields, other class methods, third party libraries, etc. In addition, most of the research papers that I reviewed for my midterm paper only evaluated the performance of LLMs on simple standalone functions that aren't representative of the code required for real-world software development. Thus, I wanted to see how well LLMs can generate comments for methods in class-level code which is more similar and representative of realistic software development.

Background

My experiment requires that I generate a dataset of class-level code with high quality comments in each method. To do this, I plan to make use of the existing ClassEval benchmark [1]. ClassEval contains 100 Python class-level code generation tasks that were originally created to evaluate the capability of LLMs in generating code for methods in a class with multiple interdependent methods and contextual dependencies. I plan to use the ground truth solutions for the class-level code generation tasks provided in this dataset since they also include good comments describing each crucial part in the code solutions for each method. I may also search for examples of class-level code annotated with good comments in other Github repositories if the number of sufficient examples provided in the ClassEval benchmark is not enough. To evaluate the comments produced by the LLMs on each class-level code example provided in the resultant dataset that I generate, I plan on using the BLEU and Human Evaluation metrics. The BLEU metric is basically a word and phrase similarity checking metric that follows the formula provided in the figure below.

$$\text{BLEU} = BP \times \exp \frac{1}{N} \sum_{n=1}^{N} \log p_n$$

Where:

$$p_n = \frac{\text{Number of ngram tokens in system and reference translations}}{\text{Number of ngram tokens in system translation}}$$

And:

The brevity penalty = $\exp(1 - r/c)$, where c is the length of the hypothesis translation (in tokens), and r is the length of the *closest* reference translation.

Figure 1. BLEU metric formula

The Human Evaluation metric will essentially just be me checking if the comments produced by the LLMs capture the same semantic meaning and intent as the comments provided for each class-level code example in the dataset that I generate.

Research Questions

1. How well can LLMs generate high-quality useful comments for methods in class-level code?
2. How well can LLMs capture the contextual dependencies present in the code through their comments?

Methodology

1. Gather a dataset of Python class-level code with high-quality useful comments from
    a. existing ClassEval benchmark [1]
        i. Review the comments for each class-level code example provided in the benchmark and add to or modify the existing comments if necessary
    b. other Github repositories if necessary
        i. Review the comments for each class-level code example obtained and add to or modify the existing comments if necessary
2. Develop a system that
    a. takes in the class-level code from the dataset
    b. Removes the comments
    c. Prompts the following LLMs for high quality comments
        i. GPT-4, CodeLlama, StarCoder, etc.
    d. Stores the outputs in some file for comparison later
3. Evaluate the comments generated by the LLMs for each method in each class against the comments provided for each method in each class in the original dataset with the following metrics
    a. BLEU(described in Background section)
    b. Human Evaluation(described in Background section)

Project Takeaways

I believe evaluating exactly how well LLMs are able to generate high-quality comments on the class-level code examples I provide can indicate how well LLMs are able to understand and reason about more complex and realistic code. Thus, I hope the results of my experiment and

research can provide insights into the future steps that can be taken to improve how LLMs perform in code generation for software development tasks.

References

[1] Du, X. *et al.* (2024) ‘Evaluating large language models in class-level code generation’, *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pp. 1–13. https://dl.acm.org/doi/10.1145/3597503.3639219