Qeios

Peer Review

Review of: "Enhancing Code LLMs with Reinforcement Learning in Code Generation: A Survey"

Alexandru Tăbușcă[1]

1. Romanian-American University, Romania

The paper presents a comprehensive and well-structured survey of how reinforcement learning (RL) is being leveraged to improve code generation using large language models (Code LLMs). It thoroughly addresses the role of RL in compiler optimization, resource allocation, dataset design, model training, and benchmarking. The authors draw upon a broad range of literature and provide technical depth across multiple RL paradigms such as PPO, DPO, and GRPO.

There are several key points that I can mention on the positive side, strengths:

- Comprehensive Scope

Covers multiple aspects of RL-enhanced code generation: from compiler-level optimizations to end-to-end development workflows. Includes detailed discussion on pre-training, post-training, and RL fine-tuning strategies.

- Strong Technical Foundation

The paper provides in-depth explanations of reinforcement learning algorithms (e.g., PPO, Actor-Critic, DPO, GRPO), clearly articulating their role and applicability to code generation. A detailed presentation of training pipelines, including dataset construction, deduplication, and RLHF considerations, demonstrates real-world relevance.

- Relevant Use Cases and Frameworks

Highlights cutting-edge implementations like CodeRL, PPOCoder, or CORGI, linking theory to real-world applications. Provides valuable insights on tool integration (e.g., GitHub Copilot, Huawei's CodeArts Snap).

doi.org/10.32388/ACHELB

- Solid Evaluation and Benchmarking

It discusses standard and specialized metrics: BLEU, CodeBLEU, pass@k, and execution accuracy. Provides a rich comparison of model performances (see Table 1), distinguishing between base models, RL-based models, and hybrids.

- Future-Oriented Perspective

The “Prospects for Future Development” section is forward-looking, pinpointing challenges like dataset quality, support for low-level languages, and computational bottlenecks.

On the other hand, there are also several key points that I consider the authors should improve on, in order to increase the quality and realism of the paper. Among these items, I can list:

1. Lack of Critical Analysis - the paper catalogs many works but stops short of critiquing their relative effectiveness. A deeper analysis of trade-offs (e.g., between RLHF and DPO) would enhance its scholarly value.

2. Underdeveloped Limitations Section - although a section on limitations exists, it lacks depth and specificity. For example, it doesn't quantify how RL methods scale poorly or under which conditions PPO outperforms DPO and vice versa.

3. Redundancy and Length - the text is at times repetitive, especially in background sections (e.g., repeated explanations of unit test-based rewards). The paper could benefit from tighter editing to enhance readability without sacrificing depth.

4. Lack of Visual Summaries - while several figures are mentioned (e.g., Figures 1-5), the text would benefit from better integration of visuals and summary tables beyond Table 1 (perhaps summarizing key RL methods or datasets).

5. Missing Empirical Insight - the paper is a survey, but a brief experimental comparison or case study of a selected RL model (e.g., PPOCoder vs CodeRL) would greatly enrich the reader’s understanding.

In conclusion, I consider this paper to be a decently valuable resource for researchers and practitioners interested in the intersection of RL and LLMs for code generation. It is ambitious in scope, technically detailed, and timely. With several improvements in critical comparison, synthesis, and editorial precision, it has the potential to serve as a reference text in this rapidly evolving field.

More specifically, I think that the following opportunities for enhancement should be taken into account by the authors:

- Comparative analysis table for different RL methods in terms of computational cost, effectiveness, and usability.
- Deeper integration of ethical and security concerns, especially as RL may optimize code that introduces subtle vulnerabilities.
- Greater attention to interdisciplinary applicability, such as using RL-augmented code generation in domains like robotics or finance.

Declarations

Potential competing interests: No potential competing interests to declare.