Yueke Zhang

| Email | Website | LinkedIn |

Education

Vanderbilt University — Nashville, USA
PhD in Computer Science — *2023- 2027 (Expected)*
Southwest University — Chongqing, China
Bachelor in Management Information System — *2019 – 2023*

Publication

- Who' s Pushing the Code? An Exploration of GitHub Impersonation.
  Yueke Zhang, Anda Liang, Xiaohan Wang, Pamela Wisniewski, Fengwei Zhang, Kevin Leach, Yu Huang
  IEEE/ACM International Conference on Software Engineering (ICSE' 2025, Core-A*) [Paper] [Artifact]
- An Empirical Study of Modeling Fault Localization as a Decision Making Process with Uncertainty and Ambiguity
  Yueke Zhang, Kevin Leach, Yu Huang
  ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM' 2023, Core-A)
  [Paper] [Artifact]
- CodeACT-R: A Cognitive Simulation Framework for Code Reading
  Yueke Zhang, Zihan Fang, Greg Trafton, Danniel Levin, Yu Huang
  IEEE/ACM International Conference on Automated Software Engineering (ASE' 2025, Core-A), NIER Paper.*
  [Paper] [Artifact]
- Pre-Training Representations of Binary Code Using Contrastive Learning
  Yifan Zhang, Chen Huang, **Yueke Zhang**, Kevin Cao, Huajie Shao, Kevin Leach, Yu Huang
  Transactions on Machine Learning Research (TMLR) [Paper] [Artifact]

Projects

Lean as an Alignment Engine: Making NL Intent Checkable in Code

- Natural language to code alignment is hard to verify; we translate code generation tasks into Lean-checked subgoals. Lean' s tactic-driven decomposition makes proof obligations explicit, guiding iterative synthesis/repair so the code conforms to a formalized intent.
- Our approach achieves 0.32 pass@1, outperforming a Qwen3 Coder only baseline (0.21) in LiveCodeBench V6. Replacing the formal prover-7B with a stronger but general Qwen3-8B as the subgoal model reduces pass@1 to 0.29, indicating that formal specialization of the prover matters more than raw model size.

CodeGrad: Integrating Formal Methods into Gradient-Based LLM Code Refinement

- Proposed the **CodeGrad** framework: This approach converts formal constraints and structured feedback into *textual pseudo-gradients*, treating code as a differentiable variable within a "forward generation - backward evaluation - gradient propagation - formal optimization" loop.
- Surpasses baselines on HumanEval / HumanEval+ and LiveCodeBench V6: Achieves an improvement of 27% on HumanEval and an improvement of 41% on LiveCodeBench V6; particularly effective on medium-difficulty tasks and complex algorithms (e.g., DP/graphs/greedy).

Aligning LLMs for Code Repair via Reward-Augmented Preference Optimization

- Proposed the **Instruct-DPO** framework, a novel pipeline for generating aligned code repair feedback. This method enhances Direct Preference Optimization (DPO) with an auxiliary reward model to better capture preference strength. The process involves constructing a corpus of preference pairs from augmented code scripts and then fine-tuning the LLM to jointly produce natural language feedback and corrected code.
- Outperformed the baseline on code generation and feedback quality: increased Pass@5 by over 7.5 points (0.39 vs 0.32) and won **56.57%** of head-to-head comparisons (vs. 13.64% for baseline).

ScanCoder: Guiding LLMs for Code Understanding and Generation with Eye-movement Simulation

- Proposed and implemented **ScanCoder**: using ACT-R generated human-likevisual attention (scanpaths) as signals, to construct token-level weighted loss, performing cognitive-guided fine-tuning for LLMs, to enhance semantic focus and interpretability on code.
- The model achieved gains on the CodeXGLUE benchmark. This includes a maximum improvement of about 39% in CrystalBLEU and 22% in BERTScore, while increasing the attention on crucial semantic tokens by approximately 2.5 times.

Technical Skills

Languages: Mandarin (native), English (fluent)
Programming: PyTorch, TensorFlow, Java, Python, Shell