Advancing Intelligent Software Development and Trustworthy Models Through the Synergy of Software Engineering and LLMs

Guanqun Yang
Stevens Institute of Technology
gyang16@stevens.edu

Introduction

In recent years, Large Language Models (LLMs) have demonstrated tremendous potential in addressing complex tasks across various domains, including software engineering. As they evolve, LLMs offer opportunities to automate and optimize many traditionally manual processes, especially in fields where adaptability is key. However, their integration into software engineering also raises reliability, safety, and scalability concerns. My research focuses on two aspects: first, developing LLM-powered solutions for software engineering problems traditionally solved by other methods, and second, using established software engineering principles to rigorously test and improve LLM reliability. By exploring these angles, my work aims to seamlessly integrate LLMs into software engineering while ensuring they meet high standards in critical applications.

At the core of this research are key questions: How can LLMs be applied to software engineering problems traditionally addressed through conventional tools? Additionally, how can software engineering techniques, such as systematic testing and adversarial assessment, improve LLM safety in high-stakes environments like cybersecurity and finance? These questions seek to bridge the gap between LLM capabilities and the rigorous demands of software engineering, creating systems that are reliable and performant in practice.

Another key focus of my work is how LLMs can replace or augment traditional software engineering tools. For example, in vulnerability detection, automated test generation, and red-teaming, conventional methods rely on specialized, labor-intensive tools requiring domain expertise. LLMs, on the other hand, can generalize across tasks and domains, offering a powerful alternative or supplement. However, challenges remain: ensuring reliable, verifiable, and safe outputs and maintaining control over generation. To address these, I apply established software engineering methodologies like automated and adversarial testing to ensure LLM performance is reliable and explainable.

LLM for Software Engineering

My work at the intersection of software engineering and large language models (LLMs) began with the development of a sample-efficient Named Entity Recognition (NER) system, addressing a critical problem in cybersecurity: extracting software metadata, including names and versions, from Common Vulnerabilities and Exposures (CVE) reports. Traditionally, this task has relied on shallow neural networks and classical machine learning models, which require large amounts of annotated data, making the process expensive and time-intensive. To overcome these challenges, I leveraged the fine-tuning of LLMs, significantly reducing data requirements while maintaining high accuracy. The resulting system achieved a 90% F1 score in vulnerability detection using only 10% of the annotated data needed by previous models. My primary contribution was the introduction of LLM fine-tuning to conventional software engineering tasks such as artifact management, demonstrating the potential of LLMs to provide highly efficient and scalable solutions, especially in data-constrained scenarios (Yang et al. 2021).

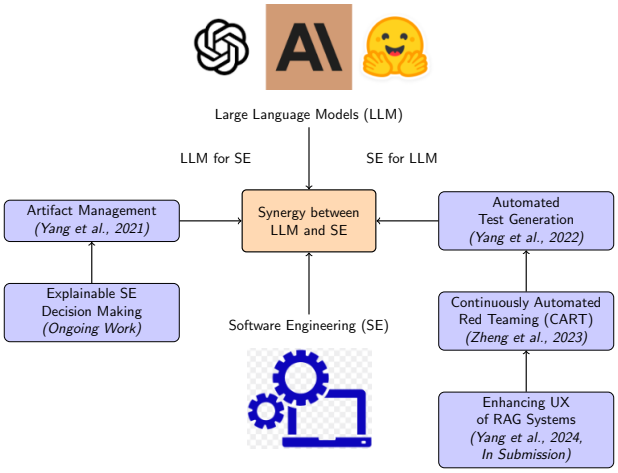

Figure 1: Synergy of LLM and Software Engineering

Currently, I am applying LLMs to another complex software engineering challenge: identifying vulnerabilities and weaknesses within code snippets based on trusted sources like the Common Weakness Enumeration (CWE). Traditional approaches often depend on large annotated datasets to train models that provide categorical decisions without

offering human-readable justifications, limiting the reliability and transparency of these systems in real-world applications. This issue is particularly acute when the distinctions between classes, such as those among CWE categories, are subtle. To address this, I am developing an LLM-driven solution that capitalizes on the reasoning abilities of multiple LLMs. Each LLM independently tackles the vulnerability identification task. At the same time, an additional LLM agent (1) verifies the step-by-step reasoning of each model and (2) reconciles their outputs to reach a final decision. This framework not only enhances explainability through a natural language interface but also improves accuracy by employing a consensus-based approach with iterative verification. This project is currently in progress, and I aim to complete it for submission to the AAAI DC in 2025.

Software Engineering for LLM

Equally important to my research is exploring how practitioners could leverage software engineering techniques and test and improve the LLMs. From August 2021 to August 2022, I developed an automated NLP system testing system, which applies behavioral testing principles to evaluate LLM outputs' reliability. Traditionally, testing frameworks in software engineering rely on test cases generated by humans or predefined rules, which are often limited in their diversity and scope. My system automated the generation of these test cases using LLMs, achieving a 98% reduction in manual effort while significantly increasing the diversity and coverage of test scenarios. This system uncovered critical vulnerabilities in top-performing NLP models on platforms like HuggingFace, demonstrating that LLMs can be both the subject and tool of rigorous testing frameworks. The success of this system highlights the potential of LLMs to drive innovation in software engineering and serve as a critical component in ensuring the robustness and safety of other LLM systems (Yang et al. 2022).

From August 2022 to December 2023, I developed a continuous automated red-teaming (CART) system designed to leverage large language models (LLMs) for automated vulnerability detection in LLMs. Traditional vulnerability detection methods rely heavily on manual intervention and domain expertise, making them labor-intensive and limited in scalability. In contrast, my system employs GPT-J-6B, utilizing instruction tuning to generate adversarial test cases that simulate complex real-world scenarios. These include producing harmful content under specific constraints, such as predefined hateful categories and targeted groups, mirroring the intricacies of real-world software systems. This automated and scalable solution reduces the need for manual assessments and broadens the depth and scope of testing. Additionally, it functions as a targeted data augmentation tool, enriching training data to improve classification performance in critical data slices, such as identifying hate speech directed at specific groups (e.g., "colored woman"). (Zheng et al. 2024).

During my internship at Capital One from June 2024 to August 2024, I applied software engineering principles to tackle the challenges of hallucination and verbosity in LLM-powered chatbots, significantly impacting user experience. My key focus was on building automated, scalable solutions to improve chatbot responses. I engineered a data synthesis pipeline that used LLMs to generate 260,000 concise training examples from unstructured data, automating a process that traditionally required manual intervention. I also developed a prompt-based filtering system with 99.4% accuracy to automatically identify relevant consumer banking examples, replacing more manual and less scalable approaches. This system directly improved the chatbot's performance by reducing verbosity by 90.55% and hallucinations by 142.85% through targeted fine-tuning with filtered synthetic data. By deliberately injecting irrelevant contexts into the training process, I enhanced the model's ability to focus on relevant information, showcasing how software engineering can solve complex chatbot development challenges, improving reliability and user experience (Yang et al. 2024).

Conclusion

In conclusion, my research is driven by a commitment to exploring how LLMs can revolutionize software engineering by providing scalable solutions to tasks that were once labor-intensive and domain-specific. Simultaneously, I am dedicated to applying established engineering methodologies to test and improve the LLMs themselves, ensuring that users can safely and reliably deploy these models in high-stakes environments like cybersecurity and finance. Through this work, I want to significantly contribute to advancing LLM applications in software engineering and developing more reliable LLM systems.

References

Yang, G.; Dashore, P.; Bathaee, N.; and Sorower, M. S. 2024. FocusRAG: Building Consumer Banking-Focused RAG Systems by Fine-Tuning LLMs to Focus. (Submitted to NeurIPS SoLaR 2024).

Yang, G.; Dineen, S.; Lin, Z.; and Liu, X. 2021. Few-Sample Named Entity Recognition for Security Vulnerability Reports by Fine-Tuning Pre-trained Language Models. In Wang, G.; Ciptadi, A.; and Ahmadzadeh, A., eds., *Deployable Machine Learning for Security Defense*, 55–78. Cham: Springer International Publishing. ISBN 978-3-030-87839-9.

Yang, G.; Haque, M.; Song, Q.; Yang, W.; and Liu, X. 2022. TestAug: A Framework for Augmenting Capability-based NLP Tests. In *Proceedings of the 29th International Conference on Computational Linguistics*, 3480–3495. Gyeongju, Republic of Korea: International Committee on Computational Linguistics.

Zheng, J.; Liu, X.; Haque, M.; Qian, X.; Yang, G.; and Yang, W. 2024. HateModerate: Testing Hate Speech Detectors against Content Moderation Policies. In Duh, K.; Gomez, H.; and Bethard, S., eds., *Findings of the Association for Computational Linguistics: NAACL 2024*, 2691–2710. Mexico City, Mexico: Association for Computational Linguistics.