STEVENS
INSTITUTE OF TECHNOLOGY
1870

Ph.D. DISSERTATION DEFENSE

Candidate:	Guanqun Yang
Degree:	Doctor of Philosophy
School/Department:	Charles V. Schaefer, Jr. School of Engineering and Science / Department of Computer Science
Date:	Tuesday, July 7, 2026
Time/Location:	2:00 PM EST, GN 303
Title:	Scalable Software Testing and Analysis with NLP Methods
Chairperson:	Dr. Xueqing Liu (Dissertation Advisor), Charles V. Schaefer, Jr. School of Engineering and Science / Department of Computer Science, Stevens Institute of Technology
Committee Members:	Dr. Hui Wang, Charles V. Schaefer, Jr. School of Engineering and Science / Department of Computer Science, Stevens Institute of Technology Dr. William Eiers, Charles V. Schaefer, Jr. School of Engineering and Science / Department of Computer Science, Stevens Institute of Technology Dr. Wei Yang, Erik Jonsson School of Engineering and Computer Science / Department of Computer Science, The University of Texas at Dallas

ABSTRACT

Building reliable and secure software increasingly depends on work that grows faster than the people available to do it: sorting and prioritizing security vulnerability reports, testing machine learning models for hidden failures, and judging the AI assistants that now help write code. Each of these tasks is still done largely by hand. This dissertation demonstrates that natural language processing (NLP) methods can handle much of this manual work. It presents four studies that form a single line of progress, in which the language model is given more independence steadily, moving from reading, to writing, to acting, and finally to working with another agent; in each study, that added independence lets the model take over a kind of manual effort that does not scale.

The first study, FewVuln, reads security vulnerability reports and extracts structured facts, such as the names and versions of the affected software. By adapting a pre-trained language model to this narrow task, FewVuln achieves the accuracy of earlier systems while using only about one-tenth as many labeled examples, replacing most of the hand labeling that such extraction normally requires and cutting that effort by roughly 90%.

The second study turns the language model from a text reader into a test writer. TestAug uses a large language model to generate test cases that check specific behaviors of a text classifier, exposing failures that a single accuracy score hides, and it replaces the slow manual authoring of such tests, reducing that effort by more than 98%. HateModerate applies the same idea to a real setting, testing whether hate-speech classifiers adhere to the 41 written content-moderation policies of a platform, and shows that training on policy-aligned

examples makes a classifier adhere to the policies more closely without lowering its accuracy on ordinary inputs.

The third study, PatchHolmes, provides the language model with a small set of tools and lets it act on a real code repository. Its task is to find the exact commit that fixes a given security vulnerability, a link that most reports never record, and that today is recovered by hand or not at all. Acting in place of that manual search, PatchHolmes raises the rate at which the correct fix is ranked first by 25.34% over a strong existing method, and it shows that the structure built around the model, rather than the size of the model, drives this result: replacing the model with one about seven times smaller changes the outcome by less than one percent.

The fourth study, WatchPoint, allows the language model to work with another agent. Acting as a simulated software developer, WatchPoint runs diagnostic scripts against a live web application and reports what it observes back to an AI coding assistant, so the assistant can repair its own mistakes. In doing so, it replaces the human testers whose time and inconsistency make large-scale evaluation costly: WatchPoint recovers 57.6% of the failing tasks it examines, which a controlled study finds close to the human recovery rate of 54.5%.

Read together, the four studies trace one progression: as the language model is given more independence, from reading text, to writing tests, to acting through tools, to working alongside another agent, it takes over a steadily larger share of the work that software quality has long demanded of people. In every case, what makes these methods practical is how the model is applied, through careful adaptation, prompting, tool use, and feedback; in the study that tested this directly, that design mattered more than the size of the model. The dissertation establishes NLP methods as a scalable foundation for testing and analyzing modern software, and shows where the effort to build on them should go.

REFERENCES

1. Guanqun Yang, Shay Dineen, Zhipeng Lin, and Xueqing Liu. Few-Sample Named Entity Recognition for Security Vulnerability Reports by Fine-Tuning Pre-Trained Language Models. KDD MLHat Workshop, 2021. https://arxiv.org/abs/2108.06590

2. Guanqun Yang, Mirazul Haque, Qiaochu Song, Wei Yang, and Xueqing Liu. TestAug: A Framework for Augmenting Capability-based NLP Tests. International Conference on Computational Linguistics (COLING), 2022. https://aclanthology.org/2022.coling-1.307/

3. Jiangrui Zheng, Xueqing Liu, Mirazul Haque, Xing Qian, Guanqun Yang, and Wei Yang. HateModerate: Testing Hate Speech Detectors against Content Moderation Policies. Findings of the Association for Computational Linguistics: NAACL, 2024. https://aclanthology.org/2024.findings-naacl.172/

4. Xueqing Liu, Jiangrui Zheng, Guanqun Yang, Siyan Wen, Qiushi Liu, and Xiaoyin Wang. Fast and Accurate Silent Vulnerability Fix Retrieval. arXiv preprint arXiv:2503.22935, 2025. https://arxiv.org/abs/2503.22935

5. Guanqun Yang, Yingming Zhou, Jiangrui Zheng, Shudong Hao, and Xueqing Liu. PatchHolmes: Agentic Patch Retrieval via Listwise Selection. Under review at EMNLP, 2026.

6. Guanqun Yang, Wei Yang, and Xueqing Liu. WatchPoint: Executable User Feedback for Real-World Agentic Web Development. Under review at COLM, 2026.