Background and Motivation

Software Analysis and Engineering with Large Language Models (LLMs) is a new course designed to equip graduate students with state-of-the-art tools and methodologies for using LLMs to enhance software development and reliability. Leveraging the power and popularity of LLMs, this course dives into the intersection of artificial intelligence, programming languages, and software engineering. The learning outcome of this course is to equip students to understand the motivation and fundamental concepts in software analysis and engineering, including automated program repair (APR), automated feedback generation, AI-assisted education, and the configuring and maintenance of modern large software systems. In particular, students will explore how advanced LLMs can redefine various aspects of software analysis and engineering in the leading research.

Prerequisite

- Programming: advanced (2+ years) programming experience, familiarity with Python-like languages
- Mathematical maturity: comfort with notation and proofs
- Moderate knowledge of algorithms and data structures
- Studied discrete math topics including sets, relations, and functions

Note

This is a newly designed course tailored for graduate students. For this first offering, enrollment is capped at 26 students to ensure enough support from both the instructor and teaching assistants. Specifically, to reserve slots for our research students in ECE, 10 slots have been allocated for MAScs and an additional 10 for PhDs. Undergraduate students who wish to pursue advanced study or research in software analysis and engineering must gain special permission from the instructor to enroll in this course.

Students do not need to have access to specific GPUs to take this course, but a laptop with at least 8GB of RAM (16GB or more highly recommended) is required.

Topics and schedule

This course plans to cover the following topics:

- Large Language Models
- Automated program repair (APR)
- Automated feedback generation
- AI-assisted CS education

1In this course outline, the terms “Large Language Models,” “LLMs,” and “LLM” are used interchangeably, following convention.
• Intermittent computing
• System misconfiguration

This table outlines a possible schedule for this course over a 12-week term. During every week of the course, students will learn the leading research on the selected topic. The instructor reserves the right to change the content and materials based on the student’s background and feedback.

<table>
<thead>
<tr>
<th>Week(s)</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Large Language Models [1-2]</td>
</tr>
<tr>
<td>2</td>
<td>Search-based APR [12-13]</td>
</tr>
<tr>
<td>3</td>
<td>Solver-based APR [27-28]</td>
</tr>
<tr>
<td>4</td>
<td>ML-based APR [3, 8, 9, 11]</td>
</tr>
<tr>
<td>5</td>
<td>LLM-based APR [5, 7]</td>
</tr>
<tr>
<td>6</td>
<td>Automated feedback generation [6, 17]</td>
</tr>
<tr>
<td>7</td>
<td>AI-assisted CS education [4-5]</td>
</tr>
<tr>
<td>8</td>
<td>White-box system misconfiguration [22, 24]</td>
</tr>
<tr>
<td>9</td>
<td>Black-box system misconfiguration [23, 28]</td>
</tr>
<tr>
<td>10</td>
<td>Intermittent computing [19-21]</td>
</tr>
<tr>
<td>11</td>
<td>Review session (1.5h) and in-class exam (1.5h)</td>
</tr>
<tr>
<td>12</td>
<td>Final project presentations</td>
</tr>
</tbody>
</table>

Grading Scheme

Grades will be based on weekly critiques, paper presentations, in-class discussions, exam, and a group research project including a final 10-page (IEEE-format) report detailing their techniques and findings. The selected project report targets top-tier programming languages and software engineering conferences for peer review.

<table>
<thead>
<tr>
<th>Items</th>
<th>% of final grade</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper critique</td>
<td>15%</td>
<td>Every week, students compose a one-page critique of the paper chosen by the instructor.</td>
</tr>
<tr>
<td>Paper presentation</td>
<td>10%</td>
<td>Students who are scheduled to present in a given week must submit slides for the pre-selected research papers. Each student will present once during the course.</td>
</tr>
<tr>
<td>In-class participation and discussion</td>
<td>5%</td>
<td>Student attendance is measured by five brief quizzes. The engagement is measured by the numbers of meaningful questions students asked in class.</td>
</tr>
<tr>
<td>In-class exam</td>
<td>25%</td>
<td>A 1.5-hour exam. The nature of this exam is to assess a student’s knowledge and understanding of the materials taught in this course.</td>
</tr>
<tr>
<td>Course project</td>
<td>45%</td>
<td>Each student group, consisting of two persons, must conduct an novel research project that digs into one or more topics from the course curriculum. The project report should be written within ten pages using the IEEE double-column format. Additionally, each group is required to submit a two-page project proposal in the same IEEE format. This proposal should include an outline, a detailed timeline, the methods to be used, and the expected results.</td>
</tr>
<tr>
<td>Total</td>
<td>100%</td>
<td></td>
</tr>
</tbody>
</table>
Textbooks and References

The course references would include (subject to be updated year by year):

5. Jialu Zhang, José Cambronero, Sumit Gulwani, Vu Le, Ruzica Piskac, Gustavo Soares, Gust Verbruggen: “PyDex: Repairing Bugs in Introductory Python Assignments using LLMs” (OOPSLA 2024)
12. Yalin Ke, Kathryn T Stolee, Claire Le Goues, and Yuriy Brun. Repairing programs with semantic code search. (ASE 2015)
15. Dongsun Kim, Jaechang Nam, Jaewoo Song, and Sunghun Kim. Automatic Patch Generation Learned from Human-Written Patches. (ICSE 2013)

22. Tianyin Xu, Jiaqi Zhang, Peng Huang, Jing Zheng, Tianwei Sheng, Ding Yuan, Yuanyuan Zhou, and Shankar Pasupathy. Do not blame users for misconfigurations. (SOSP 2013)


24. Jiaqi Zhang, Lakshminarayanan Renganarayana, Xiaolan Zhang, Niyu Ge, Vasanth Bala, Tianyin Xu, and Yuanyuan Zhou. EnCore: Exploiting system environment and correlation information for misconfiguration detection. (ASPLOS 2014)


26. Xinyu Lian, Yinfang Chen, Runxiang Cheng, Jie Huang, Parth Thakkar, and Tianyin Xu. Configuration Validation with Large Language Models. (CoRRabs/2310.09690)

