Sample for ACM Hypertext Paper

Ben Trovato*
G.K.M. Tobin*
trovato@corporation.com
webmaster@marysville-ohio.com
Institute for Clarity in Documentation
Dublin, Ohio, USA

Lars Thørväld
The Thørväld Group
Hekla, Iceland
larst@affiliation.org

Valerie Béranger
Inria Paris-Rocquencourt
Rocquencourt, France

Aparna Patel
Rajiv Gandhi University
Doimukh, Arunachal Pradesh, India

Huifen Chan
Tsinghua University
Haidian Qu, Beijing Shi, China

Charles Palmer
Palmer Research Laboratories
San Antonio, Texas, USA
cpalmer@prl.com

John Smith
The Thørväld Group
Hekla, Iceland
jsmith@affiliation.org

Julius P. Kumquat
The Kumquat Consortium
New York, USA
jpkumquat@consortium.net

Abstract

Large language models (LLMs) have become an increasingly popular tool for developers to develop and debug programs. However, for complex programs it is hard for LLMs to identify failures or areas for improvement solely using source code as input. Adding runtime information to prompts can improve the quality of LLMs' responses, but this requires significant developer time to parse logs. To increase LLM effectiveness and improve ease of use for developers, we present the framework Ghost in the Shell (GinS). GinS runs in a thread along an application and collects runtime information, which is then used to automatically construct effective prompts for an LLM. LLM analysis can be provided upon program termination, or can be triggered automatically using our novel extension of the existing "assert" debugging tool, the smart assert. A developer can insert smart asserts into their code to automatically trigger LLM analysis upon assert failure of how the program's runtime behavior caused that assert to fail. [insert more about evaluation]

CCS Concepts

• **Do Not Use This Code → Generate the Correct Terms for Your Paper**; *Generate the Correct Terms for Your Paper*; Generate the Correct Terms for Your Paper; Generate the Correct Terms for Your Paper.

*Both authors contributed equally to this research.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

Keywords

Do, Not, Us, This, Code, Put, the, Correct, Terms, for, Your, Paper

ACM Reference Format:
Ben Trovato, G.K.M. Tobin, Lars Thørväld, Valerie Béranger, Aparna Patel, Huifen Chan, Charles Palmer, John Smith, and Julius P. Kumquat. 2024. Sample for ACM Hypertext Paper. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX).* ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Large Language Models (LLMs) have rapidly become a powerful aid for developers seeking assistance with coding, debugging, and software analysis. Recent advancements have shown that LLMs are highly capable of understanding source code and suggesting improvements. However, for complex or dynamic software systems, source code alone often lacks sufficient context for effective diagnosis or optimization. Developers must typically invest significant time parsing runtime logs and program outputs to supply an LLM with the additional information needed for high-quality feedback. This reliance on manual log interpretation presents a major bottleneck: while LLMs offer potential to accelerate debugging and program comprehension, the overhead of preparing detailed runtime information can outweigh their benefits. Bridging this gap between static code and dynamic behavior remains an open challenge in improving LLM effectiveness for real-world software development tasks. To address this, we present Ghost in the Shell (GinS), a lightweight framework designed to streamline and automate the integration of runtime data into LLM-assisted workflows. GinS operates alongside a target application, continuously collecting relevant runtime information in a non-intrusive thread. Upon program termination—or triggered dynamically during execution via our novel smart assert extension—GinS compiles the collected data into rich, targeted prompts for LLMs. These prompts enable significantly more accurate and actionable analysis without requiring manual effort from the developer. Our smart assert mechanism extends traditional assertion tools by not only detecting program

failures, but also automatically capturing the runtime state that contributed to the failure. This allows LLMs to provide immediate, context-sensitive feedback on the causes of runtime errors, making debugging faster and more intuitive. Through evaluation across [insert a short preview of your evaluation here — e.g., benchmarks, case studies, examples], we demonstrate that GinS improves the quality of LLM responses, reduces developer workload, and integrates naturally into existing development practices.

1.1 Do Not Remove Boilerplate Code

The TEX document includes code to generate sections such as the the copyright block. Do not remove these. Authors of accepted papers will receive sections of code to replace these and customise the final paper accordingly.

Pay attention to the code comments about author information and add/remove authors as necessary; there must be at least one author. It is desirable that all/most authors have an ORCID ID as this is replacing the need for explicit emails (that my change as researcher move around. If an ORCID is supplied that author's name is made the anchor text of a web link to their ORCID page.

Macro codes are offered (`e.g. authornote`) which may be used as well as indicating the corresponding author(s) for communication during publication.

1.2 General points on ACM papers

See the file 'sample-sigconf.pdf' provided with this template set. It explains the basics of a number of structural features.

Newer users of LaTeX should take care to understand LaTeX's special characters that need explicit escaping. Also be aware that typographic quotes and en-dash/em-dash hyphens are not used literally in TEX documents but are indicated with macros.

Overleaf has extensive documentation covering how to indicate typographic elements in LaTeXcode.

1.2.1 A Third-level Heading. Note that level-three headings are inset into the beginning in the first paragraph of that section, regardless of line breaks in the source document.

Sections at all three levels are auto-numbered, so these do not need to be numbered in your text.

Acknowledgments

Acknowledgements go here. Delete enclosing begin/end markers if there are no acknowledgements.

References

Received 20 February 2024; revised 12 March 2024; accepted 5 June 2024