One-size-fits-all Evaluation of LLMs for Safety Assurance Considered Harmful

Simon Diemert – Critical Systems Labs Inc.
Torin Viger – University of Toronto
Jeff Joyce – Critical Systems Labs Inc.
Marsha Chechik – University of Toronto

WWW.CRITICALSYSTEMSLABS.COM

One-size-fits-all Evaluation of LLMs for Safety Assurance Considered Harmful

Simon Diemert
Critical Systems Labs Inc.
Vancouver, Canada
simon.diemert@cslabs.com

Torin Viger
University of Toronto
Toronto, Canada
torin.viger@mail.utoronto.ca

Jeffrey Joyce
Critical Systems Labs Inc.
Vancouver, Canada
jeff.joyce@cslabs.com

Marsha Chechik
University of Toronto
Toronto, Canada
chechik@cs.toronto.edu

***Abstract*—Large Language Models (LLMs) have been proposed to support the development and maintenance of assurance cases (ACs), but their use comes with risk. This position paper calls on academic and industrial interest holders to establish methodological standards to evaluate the application of LLMs to ACs. Our position is that there is no "one size-fits-all" evaluation method, and that evaluation must be tailored to the specific objectives and risk profiles of different LLM use cases. As a first step, we outline a preliminary taxonomy of characteristics, risk, and evaluation methods for LLM use cases.**

I. Introduction

There is significant interest in using Large Language Models (LLMs) to support a range of human activities, including developing and maintaining Assurance Cases (ACs). As ACs are complex engineering artifacts, they often require significant effort to develop and maintain, and consequently, many applications of LLMs such as defeater generation, argument synthesis, and change impact assessment have been explored [1]. Since ACs are often prepared for safety-critical systems, LLM use introduces risk and must be carefully considered.

Various research questions and evaluation methods have been used to assess applications of LLMs in supporting AC development. As an example, in prior work evaluating LLM effectiveness in supporting *defeater identification* (i.e., identifying reasons to doubt an AC), some researchers assessed effectiveness in terms of the LLM's ability to reproduce defeaters identified by human experts [2], [4], while another group evaluated the LLM's ability to discover valid defeaters that humans had overlooked [5]. A question arises: "which evaluation methods are appropriate for a given use case?". In a recent report, Graydon and Lehman reviewed several results using LLMs to support AC development and concluded that "much research remains to be done before it can be shown that an LLM is fit for [supporting ACs] ..." [3].

We concur with Graydon and Lehman's assessment. Indeed, more evaluation is required before LLM use for AC development can be justified as effective. However, an important question remains: *what can we do about it?*

We distinguish between an LLM *application* and a *use case* of LLMs. We refer to an application as the specific task the LLM is intended to support, such as defeater generation, argument synthesis, or change impact assessment. In contrast, we refer to use cases as specific implementations of an application in a particular context, including factors such as the role of the LLM relative to a human practitioner (e.g., collaborator vs. reviewer) and the properties its output is intended to exhibit.

Our position is that LLMs have tremendous potential to support AC development and maintenance, but to realize these benefits, the community of AC interest holders needs to establish methodological standards to support the evaluation in this emerging field. Given the diversity of LLM applications and their varying objectives and risk profiles, **there is no one-size-fits-all approach** to evaluate their effectiveness and safety. The risks introduced by an LLM use case should be identified by analyzing its characteristics such as the LLM's role, objective, and the intended properties of its output, and the evaluation required (in terms of experiments, metrics, and stanards of evidence) must then be tailored to those risks.

We propose that LLM use cases should be viewed through a framework such as the one presented in Fig. 1. Importantly, the characteristics of each use case should be studied to identify relevant risks and select an appropriate combination of evaluation methods.

II. Towards Evaluation of LLM AC Applications

Fig. 1 presents a preliminary taxonomy of characteristics, risks and evaluation metrics/methods for LLM use cases. It presents a set of characteristics of an LLM use case (including the *LLM application*, *Role of the LLM*, *LLM objective* and *Intended Properties of LLM Output*) that inform its risk profile, two types of risks, and several different *evaluation methods* and *standards of evaluation* to address a use case's risks.To demonstrate how the characteristics and risk profile of an LLM use case can impact its evaluation needs, we explore an LLM application from the literature: supporting *Defeater Generation (A1)*. Defeater identification involves identifying potential reasons to doubt an AC so that these doubts can be mitigated or eliminated. We consider two possible use cases for this application and show how their risks, and consequently, their evaluation needs differ.

A. Use Case 1 - Independent Agent

In *Use Case 1* (inspired by the proposal of Shahandashti et al. [4]), the LLM is used as an *independent agent (R1)* to automate defeater identification, replacing the task of manual defeater generation. As manual defeater identification requires

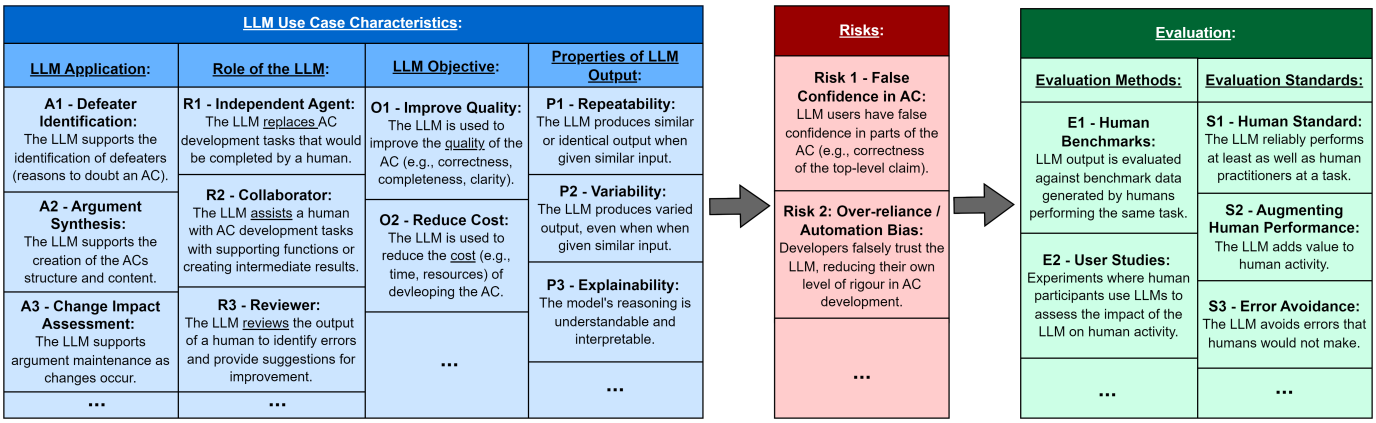

Fig. 1. Characteristics, risks and evaluation methods for LLM use cases in AC development.

time and resources, this use case prioritizes reducing the *cost (O2)* of defeater generation. However, as the LLM's role *replaces* human analysis, this use case has a high risk profile. If the LLM fails to identify critical defeaters a human expert would have included, developers and stakeholders may have *false confidence (Risk 1)* that all relevant challenges to their AC have been addressed. This is critical when false confidence could lead to the deployment of unsafe systems, and consequently, the LLM should meet a high evidentiary standard to address this risk. For this use case to be safe and effective, it may be required for the LLM to consistently *match or surpass human performance (S1)* in identifying critical defeaters. In practice, this may require *human benchmark (E1)* evaluations comparing LLM-generated defeaters to those identified by humans across a range of representative ACs.

B. Use Case 2 - Collaborator

In *Use Case 2*, the LLM is used as a *collaborator (R2)* to support manual defeater identification by helping practitioners brainstorm additional defeaters that they may have otherwise overlooked manually [5] [2]. Unlike *Use Case 1*, this application prioritizes improving the AC's *quality (O1)* by producing a more complete set of defeaters than a human might have identified on their own rather than reducing cost. Since AC developers remain responsible for identifying defeaters and determining which LLM suggestions to incorporate, the risks associated with the LLM overlooking defeaters are lower here than in *Use Case 1*. However, other risks arise when the LLM is used in a collaborative role — most notably, the potential for *automation bias (Risk 2)*, where users overrely on the LLM's suggestions even when they are low-quality or incorrect. Additionally, while the application aims to improve the completeness of identified defeaters, it may risk inadvertently increase AC development *cost* if substantial time is required to review and filter LLM defeaters.

Given these characteristics and risks, the evaluation of this use case should differ from Use Case 1. Instead of requiring the LLM to match or exceed human performance, evaluation should assess whether an LLM *augments human performance (S2)*. Suitable methods may include *user studies (E2)* comparing task outcomes with and without LLM support, such as improvements in defeater coverage or impact on time-on-task. Here, success is defined not by the LLM's output alone, but by the extent to which it adds value to human activities.

III. Call to Action

The above examples show that the appropriate methods for evaluating an LLM use case vary by risk profile. Even the same LLM application can have risks that depend on the use case. This preliminary work does not give a complete taxonomy of LLM use cases, characteristics, or risks. Rather, it offers a starting point by identifying key factors that influence a use case's risk profile and, in turn, its evaluation requirements.

We call on researchers and practitioners to extend the taxonomy the characteristics, risks, and evaluation methods for LLM use cases in AC development. Additionally, we call for collaboration between researchers and practitioners to develop structured methods for determining which evaluation methods are appropriate for addressing specific types of risk, and to identify which use case characteristics give rise to different risks. Progress in this direction will enable researchers and practitioners to systematically assess emerging LLM use cases.

References

[1] Diemert, S., Cyffka, E., Anwari, N., Foster, O., Viger, T., Millet, L., Joyce, J.: Balancing the risks and benefits of using large language models to support assurance case development. In: Computer Safety, Reliability, and Security. SAFECOMP 2025. Springer, Stockholm, Sweden (Sep 2025)

[2] Gohar, U., Hunter, M.C., Lutz, R.R., Cohen, M.B.: Codefeater: Using llms to find defeaters in assurance cases. In: ASE '24: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. pp. 2262–2267 (2024)

[3] Graydon, M.S., Lehman, S.M.: Examining Proposed Uses of LLMs to Produce or Assess Assurance Arguments. Tech. rep., National Aeronautics and Space Administration (NASA) (2025)

[4] Shahandashti, K.K., Sivakumar, M., Mohajer, M.M., Belle, A.B., Wang, S., Lethbridge, T.C.: Evaluating the Effectiveness of GPT-4 Turbo in Creating Defeaters for Assurance Cases. In: Proceedings of the 2024 IEEE/ACM First International Conference on AI Foundation Models and Software Engineering. pp. 52–56 (2024)

[5] Viger, T., Murphy, L., Diemert, S., Menghi, C., Joyce, J., Di Sandro, A., Chechik, M.: AI-Supported Eliminative Argumentation: Practical Experience Generating Defeaters to Increase Confidence in Assurance Cases. In: 2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE). pp. 284–294 (2024)