RAG Evaluation: From Model-Centric Benchmarks to System-Level Metrics

Alta de Waal[1], Daniel van Niekerk[1], Florian Donhauser[2], Salmaan Suliman[1], and Dehan Lamprecht[3]

[1] BMW IT Hub, South Africa
[2] BMW Group, Munich, Germany
[3] Department of Applied Mathematics, University of Stellenbosch, South Africa

Abstract. As Large Language Models (LLMs) increasingly support and automate business-critical workflows, the need for robust evaluation frameworks becomes paramount. This paper proposes a system-level testing approach designed to assess the performance and reliability of LLM-based applications integrated into enterprise processes. Moving beyond model-centric benchmarks, the framework adopts principles from software engineering, including black-box and end-to-end testing, to evaluate real-world outcomes in retrieval-augmented generation (RAG) systems. It features modular performance indicators such as context precision, hallucination detection, business tonality alignment, and answer correctness, many harnessing the LLM-as-a-Judge methodology. Answer correctness is used as a case study for the design of interpretable performance indicators grounded in concepts from information retrieval while considering the objectives of business and technical stakeholders. Empirical evaluations in four use cases demonstrate how this approach enables organisations to validate not only the accuracy of their systems but also business relevance.

Keywords: retrieval augmented generation · evaluation · answer correctness.

1 Introduction

Large Language Models (LLMs) have become a core technology powering modern AI systems, enabling everything from conversational assistants to code-generation tools. Robust evaluation is essential to ensure that these models perform reliably, safely, and ethically in real-world applications, where failures can result in misinformation, biased output, or significant business and reputational risks [5,7]. Traditionally, evaluation efforts have focused on benchmarking LLMs as standalone entities, using model-centric metrics such as perplexity, BLEU [21], ROUGE [15], and METEOR [2]. These metrics assess intrinsic linguistic capabilities, such as text generation, comprehension, and summarisation, and provide valuable information to improve foundational models [24,27]. However, as

LLMs increasingly become integral components within larger and more complex systems, such as AI copilots, customer service platforms, or domain-specific assistants, their evaluation must extend beyond isolated performance. In these integrated systems, users interact not with the raw model outputs but with a broader ecosystem that includes user interfaces, databases, APIs, and business logic. Consequently, evaluation shifts toward extrinsic, system-level metrics that reflect how effectively the LLM contributes to the overall goals and workflows. Key performance indicators include task completion rates, precision and recall in context-specific tasks, response latency, robustness to diverse inputs, and user satisfaction [18]. Moreover, given the inherent nondeterministic nature of LLM outputs, ongoing monitoring and uncertainty quantification are crucial to maintaining consistent and reliable system behaviour over time. End-to-end evaluation treats the entire LLM-powered application as a black box, simulating real user interactions by providing inputs and assessing outputs against criteria like relevance, correctness, and factual accuracy. This approach draws parallels from traditional software testing, unit and integration testing, but adapts to the dynamic, probabilistic nature of LLMs [10]. Instead of isolated code components, the evaluators focus on the dynamic behaviour of the system and task-oriented outcomes, ensuring that it meets user needs effectively and predictably [11].

Retrieval augmented generation (RAG) is an important type of LLM-based system which combines information retrieval with generative models to enable many useful applications. For this reason, ours and many existing evaluation frameworks have provided useful system-level performance indicators that focus on different information sources associated with RAG workflows (discussed in Section 4). Despite this, RAG evaluation remains an unsolved problem [4]. In this paper we discuss system-level metrics and how they fit into existing software testing paradigms (in Section 3), but focus our analysis on end-to-end evaluation, in particular, the comparison of an expected (golden) answer and the response generated by a RAG system for a given query. This comparison, often referred to as *answer correctness*, provides a single indicator which is representative of the system performance as a whole, implicitly evaluating components such as information retrieval and text generation. Our analysis highlights the importance of addressing two broad objectives for system-level testing: (1) identifying defects and opportunities for improvement, and (2) determining the overall quality of the system and tracking broad improvements. This translates directly to two questions to be answered by each individual performance indicator and by extension the evaluation framework as a whole:

1. *Is the system working as expected?* Where the *interpretability* of test results may be important to developers to identify system issues or areas for enhancement.
2. *Is the system improving or regressing?* Where the *alignment* between test scores and product owners' (business) expectations regarding system quality is important to evaluate the system as a whole.

Given these objectives, the contributions of this paper are summarised as follows:

1. The above goals are motivated by contextualising system-level testing within the domains of LLM evaluation and software engineering, leading to an evaluation framework which addresses interpretability and alignment.
2. The implementation of RAG performance indicators with interpretability grounded in concepts from information retrieval while aligning with human evaluation is explored by considering answer correctness as a case study.
3. An empirical analysis of these implementations is presented across four different RAG applications, to validate their correlation with business-driven quality expectations and the degree to which intermediate representations such as factual statements agree with manual evaluation.

We begin with the contextualisation of this work in Sections 2 and 3. This is followed by a more focused description of our evaluation framework and a general perspective on the application of system-level metrics such as answer correctness in Section 4. Lastly, we present our empirical work with results in Section 5, followed by conclusions and recommendations in Section 6.

2 System-level LLM evaluation

In the previous section, we highlighted the distinction between model-level and system-level evaluation and motivated our focus on system-level evaluation. In this section, we thus briefly describe key approaches to LLM system-level evaluation.

2.1 LLM as a judge

The LLM-as-a-Judge approach to evaluation was formally introduced and empirically validated in early works such as [35], which proposed using strong LLMs like GPT-4 to evaluate chatbot responses via pairwise comparisons in benchmarks like MT-Bench and Chatbot Arena. Their study demonstrated over 80% agreement between LLM judgments and human preferences, establishing LLMs as viable proxies for subjective evaluations[35].
Subsequent research expanded this paradigm, with surveys [13,8] compiling foundational methodologies and applications. For instance, comprehensive overviews classify LLM-as-a-Judge into pointwise (single-output scoring), pairwise (comparative ranking), and listwise (multi-output ordering) evaluation modes, highlighting its flexibility across tasks like question answering and summarisation[13,8].

Recent advancements have focused on enhancing the robustness of LLM-as-a-Judge through fine-tuning and ensemble methods[13]. For example, models like Themis have been developed as specialised LLM judges, fine-tuned to provide context-aware evaluations with improved interpretability[9]. Other innovations include reference-guided scoring[1], where judges use ground-truth examples for calibration, and multi-agent panels[31] (e.g., ensembles of smaller LLMs) to reduce intra-model bias and costs.
Empirical studies in 2024-2025 have also explored domain-specific adaptations,

such as in code generation[22] or query parsing[3], where LLM judges correlate highly with expert assessments.

Empirical analyses underscore the paradigm's efficacy across domains. In question-answering tasks, LLM judges achieve Spearman correlations of 0.6-0.7 with human annotations, outperforming traditional metrics like ROUGE or BLEU for open-ended responses [16]. Applications extend to enterprise settings, such as evaluating generative AI in chatbots[8] or judicial decision-making, where platforms like Amazon Nova leverage LLM-as-a-Judge for automated quality checks. However, domain-specific benchmarks reveal varying performance, with higher alignment in general tasks but challenges in expert domains like mental health[12] or mathematics.

Despite its strengths, the LLM-as-a-Judge paradigm is not without limitations. Inherent biases, such as position bias [32] (favouring outputs in specific prompt orders), verbosity bias (preferring longer responses), and self-bias (preferring responses from similar models), can undermine reliability [34]. Studies highlight non-transitivity in preferences and sensitivity to prompt variations, leading to inconsistent judgments. In expert knowledge tasks, LLM judges often fail to align with subject matter experts, achieving only 60-70% agreement in domains like dietetics or law. Resource intensity and adversarial vulnerabilities further complicate deployment.

Looking ahead, meta-evaluation frameworks like RobustJudge [14] aim to benchmark LLM judges' reliability across scenarios, emphasising the need for standardized prompts and bias mitigation. Hybrid approaches combining LLM judges with human oversight or programmatic metrics could further enhance trustworthiness, paving the way for broader adoption in high-stakes applications.

2.2 Validating the validators

Despite the appeal of using LLMs as evaluators, several open questions remain regarding the strengths and weaknesses of this paradigm, including their alignment with human expert-driven evaluations [30]. Prior studies have explored this alignment using metrics such as Cohen's Kappa [30] and Spearman's rank correlation [17]. When evaluations are task-specific, an empirical analysis can be valuable in determining the suitability of LLMs as judges, as previously described for software engineering tasks such as code translation, code generation, and code summarization [33]. Beyond measuring alignment, the design of evaluator functions and prompts is also a topic of active research. For instance, EvalGen is an interface designed to assist in evaluating LLM outputs by automating the generation of evaluation criteria and implementing them as Python functions or LLM grader prompts [29]. It incorporates human feedback on a subset of outputs, refining and selecting implementations that align better with user preferences. Using a collaborative, mixed-initiative approach, EvalGen allows humans to iteratively align LLM-generated evaluation functions with their requirements, effectively "validating the validators". Furthermore, it addresses the phenomenon of "criteria drift" by supporting users in adapting evaluation crite-

ria as they grade outputs, acknowledging that criteria may evolve and sometimes depend on specific LLM outputs observed.

3 Evaluation through software engineering principles

Evaluating LLM-based systems presents challenges that require the adaptation of traditional software engineering testing methodologies. Unlike conventional software components with deterministic behaviour, LLM-based systems exhibit nondeterministic outputs, opaque internal processes, and complex interactions between retrieval, generation, and domain-specific logic.

Before introducing our evaluation framework, we situate it within the established test pyramid framework to assess the reliability and performance of LLM-based applications. Our approach integrates well-recognised testing methodologies, specifically black-box testing, that are adapted to address the unique characteristics of AI systems, such as nondeterministic outputs and evolving models.

Together, these methodologies underpin our end-to-end evaluation framework. Finally, we relate these approaches to the broader concepts of verification and validation, highlighting their roles in systematically assuring the correctness and quality of LLM applications.

3.1 Software testing pyramid

The software testing pyramid is a model used in software engineering to organise and prioritise various types of automated tests [6]. The primary objective of this model is to motivate developers and testers to consider various levels of testing and to distribute their efforts proportionately among these levels. This means that emphasis is placed on a large foundation of low-level tests and progressively fewer tests at higher levels (see Figure 1).

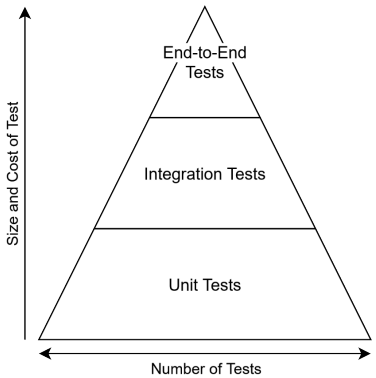

Fig. 1. The software testing pyramid [6]

Unit Tests form the base of the pyramid and represent the largest number of tests. These tests focus on verifying the smallest individual pieces of code, such as functions or methods, in isolation. They are inexpensive to develop and run quickly, thereby providing developers with timely feedback on the integrity of individual components. However, these tests are deterministic and only test the low-level functionality of the system.

The middle layer consists of integration tests. These tests ensure that various software components or modules function correctly when integrated. Because these tests usually involve multiple components of the system, they tend to be more complex and take longer to execute compared to unit tests. Although they assess the interactions between components, they do not evaluate the entire system as a whole. As a result, it is not possible to fully measure business-relevant metrics.

At the top of the pyramid are end-to-end (E2E) tests. These tests validate the entire application from the user's perspective, simulating real user scenarios to ensure the software meets business requirements and functions as expected. These tests are the most expensive to develop and the slowest to execute and represent the most complex layer of the testing pyramid. However, E2E tests often suffer from flakiness due to shifting requirements, resulting in a significant maintenance burden [28]. This challenge becomes even more pronounced in systems that incorporate AI or machine learning components, as these are inherently nondeterministic. Even before the proliferation of large language models (LLMs), software quality practitioners had been exploring various AI techniques to tackle this persistent issue [28,26]. The emergence of LLM-as-a-Judge evaluation paradigms offers another promising solution. Rather than relying on rigid, exact-match assertions that fail when outputs vary slightly, LLM judges can assess semantic correctness and functional adequacy across diverse but acceptable variations in system responses.

In fact, many testing frameworks exist across all levels of the pyramid; however, most of these frameworks work on binary pass/fail metrics. In this work we thus explore an end-to-end testing framework that scores the system using business-relevant metrics.

3.2 Black-box testing

Any testing technique within the pyramid (Figure 1) can be considered on a white-box to black-box axis. In our framework, we emphasise a black-box approach because it is particularly well-suited to LLM evaluation as it aligns with the fundamental opacity of neural language models. Black-box testing examines an application's functionality without inspecting its internal structures. In contrast to white-box testing, the black-box method focuses solely on inputs and outputs [25]. Testers can identify discrepancies between expected and actual results by inputting various inputs into the system and observing the output. This process does not require understanding the complexities of the underlying model or infrastructure. As an example, the context precision metric (described in Section 4.1) examines chunks of text returned by the retrieval system, but does not

consider implementation details such as the associated text embeddings. One further advantage of black-box testing is that it generalises well and allows us to dynamically evaluate different business applications.

3.3 Verification & validation testing

The final distinction to consider is between verification and validation. *Verification* refers to the set of tasks ensuring the software system implements a specified function correctly. *Validation* refers to a set of tasks that ensure the software system meets the requirements and expectations of end-users and stakeholders [20,25]. In the context of a RAG application, we can phrase this as:

Verification: "Are we answering the questions correctly?"
Validation: "Are we answering relevant questions?"

The former assesses the accuracy, completeness, and relevance of generated responses against ground truth or expert expectations, offering measurable quality indicators over time. The latter ensures the system meets user needs, operates effectively within its domain, and aligns with business requirements.

4 Proposed framework

Many evaluation frameworks for RAG and other LLM-based systems are available, with many still being created and refined, such as RAGAS, [4] DeepEval, [5] Galileo [6] and others. They often focus on providing flexible, modular metrics for assessing aspects like faithfulness, context recall, answer relevance, and factual correctness. Other frameworks, including CCRS (a zero shot LLM-as-a-Judge approach) [19] and THELMA (holistic task-based evaluation) [23], extend this by incorporating LLM reasoning for nuanced assessments, often without the need for labeled data. While these tools are similar to our work in their use of LLM-as-a-Judge for scalable metrics that evaluate retrieval and generation components of RAG systems, they can fall short in providing tailored, time- and cost-saving guidelines or best practices for specific organisational contexts. The result is that LLM-based systems practitioners often encounter challenges related to the implementation and scalability decisions requisite for evaluating their systems. Furthermore, after implementation, practitioners are often left unsure of the coverage and accuracy of their solutions.

With these challenges in mind we propose a framework based on three fundamental design principles: Firstly, the framework addresses these gaps through organisation specific defaults, suggestions, and standardised best practices, ensuring that teams across use cases can efficiently evaluate without reinventing the

[4] https://www.ragas.io/
[5] https://www.confident-ai.com/
[6] https://galileo.ai/

wheel. For example, reusable definitions of tonality and security that are appropriate for the organisation and built-in prompts, models, and hyperparameters that avoid some of the pitfalls of using LLM-as-a-Judge.

Secondly, the framework is a comprehensive toolkit designed to evaluate the reliability and veracity of these systems. It streamlines the testing process by providing a suite of metrics specifically tailored for assessing various aspects of RAG implementations. The core functionality revolves around a set of metrics that measure key performance indicators such as answer correctness, hallucination, and tonality. These metrics leverage LLM-as-a-Judge to provide scalable evaluations, enabling developers to identify areas for improvement and ensure that their systems meet the desired quality standards.

Finally, our framework recognises the fact that different use cases may prioritise different evaluation criteria. To address this, it follows a modular design, allowing users to select and combine relevant metrics based on their specific requirements. For example, applications focused on typical RAG systems may emphasize metrics related to answer correctness and factual accuracy. In contrast, conversational agents can prioritize a tonality metric to ensure natural and engaging interactions.

4.1 Framework metrics

In combination with the design priniciples, the framework offers a comprehensive suite of metrics to evaluate various aspects of a RAG system's reliability. Some of the key metrics included are:

- **Answer Correctness Metric**: Evaluates the overall correctness of a generated answer against an expected answer, providing a holistic assessment of response quality.
- **Analytic Answer Correctness Metric**: Breaks down the generated response into individual statements and evaluates each statement's factual alignment with the expected answer, offering precision and recall metrics.
- **Context Precision Metric**: Assesses the relevance of the source documents retrieved by the RAG system for constructing the answer, ensuring that the most pertinent information is sourced.
- **Hallucination Metric**: Identifies factual inconsistencies through detecting unsupported or fabricated content in the generated response by comparing it against the content from the retrieved sources.
- **Tonality Metric**: Verifies the stylistic and tonal consistency of the generated text based on predefined guidelines, ensuring that the output adheres to the desired brand voice or tone.

4.2 Applying system-level metrics

As mentioned, many evaluation frameworks include predefined performance indicators that are, in principle, generally applicable to different types of systems

such as RAG. However, acknowledging the important considerations for applying LLM-as-a-Judge that affect the agreement with manual evaluators (such as biases and domain-specific sensitivity), most frameworks recommend that users create their own prompts for evaluation, putting the onus and responsibility back on practitioners. Due to the often complex considerations discussed in Section 2, this means that practitioners need to conduct due diligence to test the reliability of their LLM-as-a-Judge implementations and to clearly define their evaluation goals regardless of which framework is used.

Performance indicators (based on LLM-as-a-Judge or otherwise) can be characterised in terms of *which sources of information* are being compared and *which criteria are used* for comparison. For example, in typical RAG systems, for a given query, it may be useful to compare the generated response with the retrieved knowledge sources to determine the prevalence of false positives which indicates the propensity of the generative model for *hallucination* in the application context, or to compare an expected set of knowledge sources with what was actually retrieved to evaluate the retrieval component. The standard information retrieval (IR) measures including precision, recall, and F-score can be applied in many of these scenarios.

Ideally, evaluators would select and combine an appropriate set of performance indicators to answer the two system-level questions (involving validation and verification as discussed in Section 3):

1. *Is the system working as expected?* The interpretability of scores for individual test cases may be important to developers to identify system issues or areas for enhancement.
2. *Is the system improving or regressing?* The relationship between aggregate scores and product owners' (business) expectations regarding system quality is important here as they often evaluate outcomes as a whole.

4.3 Answer correctness as a case study

Due to the burden of verifying the alignment of multiple LLM-as-a-Judge tasks, it is useful to consider the most direct way to determine the overall system performance, that is, to compare the generated response with an expected answer – *answer correctness* – as a case study. Figure 2 shows answer correctness applied to a RAG system with a simple Venn diagram. This illustrates that to answer the two system-level questions may require understanding the larger context of the comparison. For example, to accurately assess the answer correctness one may have to consider inadequacies such as underspecification of expected outputs or the relative importance of individual statements or facts in each of the data sources. Furthermore, in addition to the quantitative information-centred aspects, the relative quality of different outputs may also depend on qualitative aspects such as fluency, coherence, or style, depending on the application.

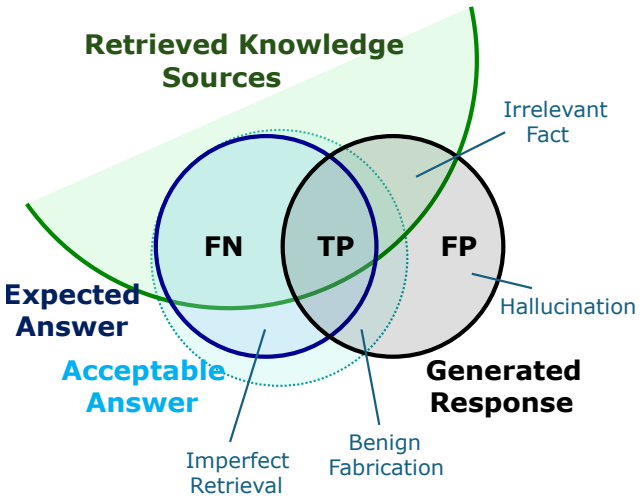

Fig. 2. A quantitative information-centred view of *answer correctness* applied to a RAG system with characterisation of system behaviours. This illustrates a number of considerations when arriving at a score which may agree with business-level assessments. Firstly, the set of statements constituting an *acceptable answer* is almost always larger than defined expectations and is often not restricted to the formal knowledge sources; leading to acceptable statements that may be scored as false positives (shown as "benign fabrication"). Secondly, inaccuracies in the information retrieval process may lead to false negatives or false positives (respectively indicated as "imperfect retrieval" and "irrelevant fact"). Lastly, unsubstantiated statements that are not part of an acceptable answer are often grouped together as "hallucinations".

5 Experiments

We evaluate two classes of implementations of answer correctness that we expect to align with the broad goals of different stakeholders (discussed in the previous section):

1. *Scored Answer Correctness* is based on a single-prompt task description which requests a graded score ranging from 0 to 10 (normalised to the range $[0.0, 1.0]$) to represent the appropriateness of a response given a query and expected answer (also including a generated "reason"). This approach is expected to leverage the judge model's sophisticated internal representations to weigh different relevant aspects given the query context.
2. *Analytic Answer Correctness* proceeds in two independent steps to (1) split the expected answer and response into factual statements, and (2) generate verdicts as to whether the statements are aligned to determine true positives, false positives, and false negatives from which an F-score is calculated.
3. *Contextualised Analytic Answer Correctness* is conceptually identical to (2) but is implemented with a single prompt to generate statements with verdicts

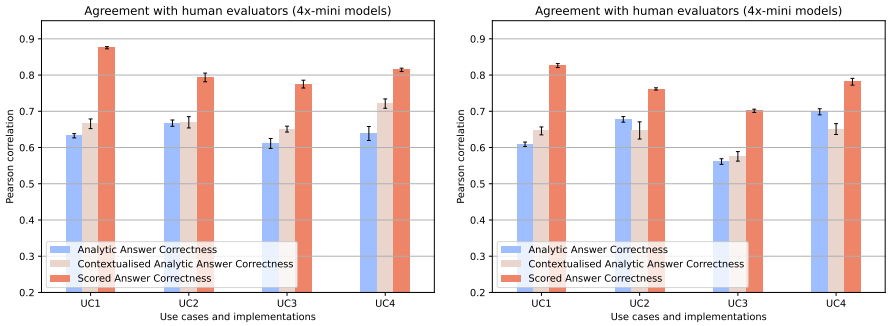

Fig. 3. Mean Pearson correlation coefficients for the different methods compared to manually provided scores (error bars indicate 95% confidence intervals). On the left are results for the stronger GPT-4x models and on the right for the smaller "mini" models.

in a single step. We expect this contextualised implementation to perform more appropriate (relevant) factual segmentation if the model is capable of following the more complex instructions.

We then perform two distinct analyses: firstly, we determine the degree to which human and generated scores are correlated, and secondly, we consider different trends in manual and generated factual statement splitting. Details are presented in the following sub-sections.

5.1 Setup

Four generative AI use case (UC) owners were asked to provide test sets with queries and golden answers for their respective RAG-based systems. The content ranged from quantitative data to legal and operational descriptions in English and German:

- *UC1*: A "search and chat" interface to business intelligence data.
- *UC2*: A customer-facing chatbot relating to product specifications.
- *UC3*: A human resources question answering system.
- *UC4*: A procurement question answering system.

Each test set consisted of 50 test scenarios for a total of 200 test cases which were then presented to each of the systems to generate responses for evaluation. For each of the responses, a member of the respective team was also asked to provide an independent score indicating the quality of the response of the their respective system. For three of the use cases (*UC1, UC2, UC3*) we also performed manual splitting of the expected answers and responses into factual statements as well as classification into true positives, false positives, and false negatives.

We then evaluated the test cases over 5 runs with 4 LLMs (GPT-4o, GPT-4oMini, GPT-4.1, and GPT-4.1Mini) and temperature setting of 0.1.[7] Ideally, more deterministic outputs are preferrable but a temperature of 0.0 can cause practical problems in the case where an output format cannot be parsed and needs to be regenerated.

5.2 Results

For the business score correlation analysis we calculated the Pearson correlation coefficient over all scores in each run. For the analytic implementations we calculated the F1-score which weighs the importance of precision and recall equally. Finally correlations over runs and models were aggregated for each use case with 95% confidence intervals, see Figure 3. When considering the results averaged over all four LLM judge models, the simple graded score prompt was significantly more correlated with manually assigned scores for all four use cases and the two analytic implementations perform similarly with no justification for preferring one over the other. However, if one excludes the "mini" models and only considers the stronger models (Figure 3 - left) the contextualised implementation has comparable or better correlation with manual scores (significantly better in three out of four UCs). Regarding GPT-4o compared to GPT-4.1 models, we did not observe consistent improvements using the newer models.

Next we focussed on the different characteristics of the two analytic implementations in terms of the number of statements extracted from the two texts, i.e., the generated responses and expected answers. For the analysis, we consider the manual statement splitting of each test case as the reference and thus determined the difference in number of generated statements in each of the following three categories needed to calculate the F-score (illustrated in Figure 2):

- True positives (TP): statements which are present in both the generated response and expected answer.
- False positives (FP): statements which are only found in the generated response.
- False negatives (FN): statements which are only found in the expected answer.

Figure 4 shows the resulting distributions comparing the two methods. The most significant observation is that the two-stage implementation significantly overgenerates false positive statements, that means that responses that are more verbose than the expected answer are penalised more heavily and there are test cases where the approach generated a large number of additional statements (greater than 25) compared to the manual reference. The contextualised prompt is able to match the manual consideration of relevant facts in the response more closely. The number of true positive statements are very similar, tending to over generate more often than under generate. Lastly, the contextualised prompt

[7] We initially scanned a temperature range of 0.0 to 0.5, finding a weak trend towards higher correlations at 0.1.

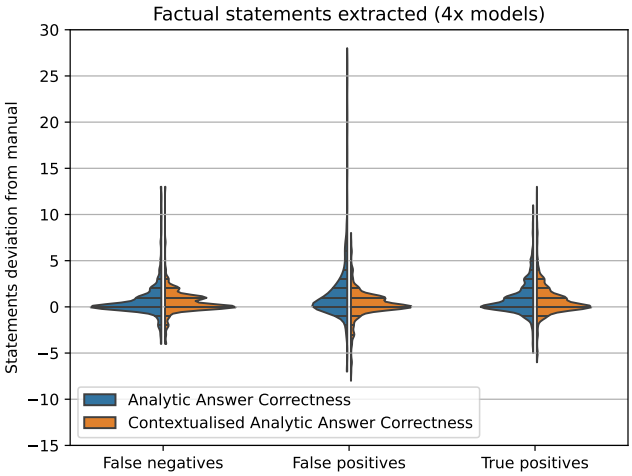

Fig. 4. The violin plots compare the distributions (histograms) of the difference in the number of factual statements, generated compared to manual, organised by their classification. Positive values indicate over generation of statements while negative values mean that fewer distinct statements than expected were identified.

generated more false negatives than the two-stage process, meaning that the current implementation is more strict on ensuring that all details in the expected answer are contained in the response.

5.3 Discussion

The results confirm our expectations that a single-prompt (holistic) task description has a better chance of aligning with expectations of diverse sets of business owners (monitoring improvements or regressions in their systems). However, the more interpretable analytic methods provide a straightforward mechanism to tune for better alignment by setting the relative importance of precision and recall which we have not done here. We did notice different levels of correlation between business scores and precision / recall scores for different use cases; some scores were more correlated with recall than precision.

Secondly, for evaluators interested in an IR-based metric, we again demonstrated the advantage of a contextualised implementation even for conceptual sub-tasks such as factual statement splitting. This allows the LLM judge to use the task context to determine relevant details in the inputs and thus an appropriate granularity of factual statements. We believe that further improvements are still possible for the analytic approach, for example, to assign an explicit indication of the relevance of individual statements.

Finally, even as advanced LLMs become more adept at generalising over tasks and domains, considerable opportunities exist to improve the accuracy of LLM-as-a-Judge evaluation for any particular use case. The first design consequence

of retaining flexibility, by allowing evaluators to customise performance indicators to their use case, remains important. However, a potentially undervalued approach is to ensure that high quality transparent and interpretable evaluation processes are provided where evaluators can understand and adapt outputs (rather than implementations) to their use cases by examining intermediate representations.

6 Conclusion

This work explores the shift in evaluating Large Language Models (LLMs) from traditional model-centric metrics to system-level performance indicators, emphasising their integration into broader applications like RAG-based systems. We assert the value of the LLM-as-a-Judge paradigm as a scalable dynamic black-box testing alternative to human-driven evaluations. This requires a modular design with different performance indicators such as answer correctness, hallucination detection, and tonality that are applicable to different use cases (as exemplified in other evaluation frameworks). However, with our framework we aim to address some of the practical shortcomings of existing frameworks by explicitly validating its performance against the two system-level objectives: (1) identifying defects and opportunities for improvement, and (2) determining the overall quality of the system and tracking broad improvements. To this end, we presented empirical results that demonstrate the superior correlation of holistic scoring methods with business expectations and illustrate how analytic approaches can be designed to provide interpretability and ease of customization for different scenarios. We argue that where it is not possible to optimally address both aspects in a single implementation, there is value in implementing transparent and interpretable variants alongside those that are tuned for alignment with business expectations of quality.

Future work should extend the types of analyses presented here to further improve the alignment of especially interpretable evaluation methods for *answer correctness*, but also to other useful RAG performance indicators such as *hallucination* and *context precision*. We foresee that the implementation of these alignment efforts will evolve from prompt engineering, to evaluate systems against pre-defined test cases, to designing *evaluation workflows* for agentic evaluation using dynamic test content. While the fundamental design principles of our framework would remain the same, the expected advantages of more dynamic workflows would be in terms of rapid specialisation to different use cases, up-to-date test cases, and better contextualisation during evaluation.

References

1. Badshah, S., Sajjad, H.: Reference-guided verdict: Llms-as-judges in automatic evaluation of free-form text. ArXiv (2024)
2. Banerjee, S., Lavie, A.: METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In: Proceedings of the Workshop

on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization, ACL-2005. Ann Arbor, Michigan (June 2005)
3. Baysan, M.S., Uysal, S., Islek, I., Cıg Karaman, C., Güngör, T.: Llm-as-a-judge: automated evaluation of search query parsing using large language models. Frontiers in Big Data **Volume 8 - 2025** (2025)
4. Benaich, N., Chalmers, A.: State of AI Report 2024. Tech. rep., Air Street Capital (October 2024), https://www.stateof.ai/2024
5. Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., Ye, W., Zhang, Y., Chang, Y., Yu, P.S., Yang, Q., Xie, X.: A Survey on Evaluation of Large Language Models. ACM Trans. Intell. Syst. Technol. **15**(3) (Mar 2024)
6. Cohn, M.: Succeeding with Agile: Software Development Using Scrum. Addison-Wesley Professional, 1st edn. (2009)
7. Croxford, E., Gao, Y., Pellegrino, N., Wong, K., Wills, G., First, E., Liao, F., Goswami, C., Patterson, B., Afshar, M.: Current and future state of evaluation of large language models for medical summarization tasks. npj Health Systems **2**(6) (2025)
8. Gu, J., Jiang, X., Shi, Z., Tan, H., Zhai, X., Xu, C., Li, W., Shen, Y., Ma, S., Liu, H., Wang, S., Zhang, K., Wang, Y., Gao, W., Ni, L., Guo, J.: A Survey on LLM-as-a-Judge (2025), https://arxiv.org/abs/2411.15594
9. Hu, R., Cheng, Y., Meng, L., Xia, J., Zong, Y., Shi, X., Lin, W.: Training an LLM-as-a-Judge Model: Pipeline, Insights, and Practical Lessons. In: Companion Proceedings of the ACM on Web Conference 2025. p. 228–237. WWW '25, ACM (May 2025)
10. Hudson, S., Jit, S., Hu, B.C., Chechik, M.: A Software Engineering Perspective on Testing Large Language Models: Research, Practice, Tools and Benchmarks (2024), https://arxiv.org/abs/2406.08216
11. Jiang, H., Kim, B., Guan, M.Y., Gupta, M.: To Trust Or Not To Trust A Classifier. In: Advances in Neural Information Processing Systems (NeurIPS) 2018 (2018)
12. Li, A., Lu, Y., Song, N., Zhang, S., Ma, L., Lan, Z.: Understanding the therapeutic relationship between counselors and clients in online text-based counseling using llms. ArXiv (2024)
13. Li, H., Dong, Q., Chen, J., Su, H., Zhou, Y., Ai, Q., Ye, Z., Liu, Y.: LLMs-as-Judges: A Comprehensive Survey on LLM-based Evaluation Methods. arXiv preprint arXiv:2412.05579 (2024), https://arxiv.org/abs/2412.05579
14. Li, S., Xu, C., Wang, J., Gong, X., Chen, C., Zhang, J., Wang, J., Lam, K.Y., Ji, S.: LLMs Cannot Reliably Judge (Yet?): A Comprehensive Assessment on the Robustness of LLM-as-a-Judge (2025), https://arxiv.org/abs/2506.09443
15. Lin, C.Y.: ROUGE : A package for automatic evaluation of summaries. In: Text Summarisation Branches Out. pp. 74–81 (2004)
16. Liu, Y., Iter, D., Xu, Y., Wang, S., Xu, R., Zhu, C.: G-Eval: NLG evaluation using gpt-4 with better human alignment. In: Proceedings of the 2023 Conference on EMNLP. pp. 2511–2522. Association for Computational Linguistics (2023)
17. Liu, Y., Zhou, H., Guo, Z., Shareghi, E., Vulić, I., Korhonen, A., Collier, N.: Aligning with Human Judgement: The Role of Pairwise Preference in Large Language Model Evaluators (2024), arXiv preprint arXiv:2403.16950
18. Miller, J.K., Tang, W.: Evaluating LLM Metrics Through Real-World Capabilities (2025), https://arxiv.org/abs/2505.08253
19. Muhamed, A.: CCRS: A Zero-Shot LLM-as-a-Judge Framework for Comprehensive RAG Evaluation (2025), https://arxiv.org/abs/2506.20128

20. O'Keefe, R.M., Balci, O., Smith, E.P.: Validating expert system performance. IEEE Intelligent Systems **2**(04), 81–90 (Oct 1987)
21. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: A method for automatic evaluation of machine translation. In: Annual Meeting of the Association for Computational Linguistics. pp. 311–318 (2002)
22. Patel, B., Chakraborty, S., Suttle, W.A., Wang, M., Bedi, A.S., Manocha, D.: Aime: Ai system optimization via multiple llm evaluators. ArXiv (2024)
23. Patel, U., Mulkar, R., Roberts, J., Senthilkumar, C.C., Gandhi, S., Zheng, X., Nayyar, N., Kalra, P., Castrillo, R.: THELMA: Task Based Holistic Evaluation of Large Language Model Applications-RAG Question Answering (2025), `https://arxiv.org/abs/2505.11626`
24. Polonioli, A.: Moving LLM evaluation forward: Lessons from human judgment research. Frontiers in Artificial Intelligence **8**, 1592399 (2025)
25. Pressman, R.S., Maxim, B.R.: Software Engineering: A Practitioner's Approach. McGraw-Hill Education, New York, 9th edn. (2020)
26. Radziwill, N.M., Freeman, G.R.: Reframing the test pyramid for digitally transformed organizations. ArXiv (2020)
27. Rahmani, H.A., et al.: LLM4Eval@WSDM 2025: Large Language Model for Evaluation in Information Retrieval and Natural Language Generation. In: Proceedings of the 18th ACM International Conference on Web Search and Data Mining (WSDM '25) (2025)
28. Ricca, F., Marchetto, A., Stocco, A.: AI-based Test Automation: A Grey Literature Analysis. In: 2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). pp. 263–270 (2021)
29. Shankar, S., Zamfirescu-Pereira, J., Hartmann, B., Parameswaran, A., Arawjo, I.: Who Validates the Validators? Aligning LLM-Assisted Evaluation of LLM Outputs with Human Preferences. In: Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology. UIST '24, Association for Computing Machinery, New York, NY, USA (2024)
30. Thakur, A.S., Choudhary, K., Ramayapally, V.S., Vaidyanathan, S., Hupkes, D.: Judging the Judges: Evaluating Alignment and Vulnerabilities in LLMs-as-Judges (2024)
31. Verga, P., Hofstatter, S., Althammer, S., Su, Y., Piktus, A., Arkhangorodsky, A., Xu, M., White, N., Lewis, P.: Replacing judges with juries: Evaluating llm generations with a panel of diverse models. ArXiv (2024)
32. Wang, P., Li, L., Chen, L., Cai, Z., Zhu, D., Lin, B., Cao, Y., Kong, L., Liu, Q., Liu, T., Sui, Z.: Large Language Models are not Fair Evaluators. In: Ku, L.W., Martins, A., Srikumar, V. (eds.) Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 9440–9450. Bangkok, Thailand (Aug 2024)
33. Wang, R., Guo, J., Gao, C., Fan, G., Chong, C.Y., Xia, X.: Can LLMs Replace Human Evaluators? An Empirical Study of LLM-as-a-Judge in Software Engineering (2025)
34. Ye, J., Wang, Y., Huang, Y., Chen, D., Zhang, Q., Moniz, N., Gao, T., Geyer, W., Huang, C., Chen, P.Y., Chawla, N.V., Zhang, X.: Justice or Prejudice? Quantifying Biases in LLM-as-a-Judge (2024)
35. Zheng, L., Chiang, W.L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., Zhang, H., Gonzalez, J.E., Stoica, I.: Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In: Advances in Neural Information Processing Systems. (NeurIPS), vol. 36 (2023)