EasyChair Preprint

№ 15699

HyGen – a Hybrid Automation Testing Approach for Reducing Hallucination in LLM Based Applications

Amit Chakraborty, Chirantana Mallick, Rajdeep Chakraborty and Saptarshi Das

January 10, 2025

HyGen – A Hybrid Automation Testing Approach for reducing hallucination in LLM based applications

Amit Chakraborty
PhD Scholar
JIS IASR
amit13.ons@gmail.com

Chirantana Mallick
Assistant Professor
JIS IASR
chirantana@jisiasr.org

Rajdeep Chakraborty
Associate Professor
Medicaps University
rajdeep.research@gmail.com

Saptarshi Das
Assistant Professor
JIS IASR
saptarshi@jisiasr.org

Abstract

Subjective manual evaluation of hallucinations in Large Language Models (LLMs) remains a challenging task due to its time-consuming and labored character. It expects human expertise to analyze each LLL response critically to assess cases where the model produces wrong or misleading data. This manual approach is usually small-scale and logical in extent, bringing elements of judgment into the assessment equation. To solve these problems, we have proposed a novel hybrid test automation framework that integrates rules-based and model-based grading methods. The rule-based evaluation, done at the per-commit granularity during the continuous integration (CI) cycle, offers quick feedback on the exact hallucination patterns, while the model-based grading method is carried out during a increment release and is evaluated using a critique LLM. Since there is already known behaviors concerning hallucinations, the framework can readily encode these behaviors in a set of rules and thus be able to alert on potential issues. A more in-depth evaluation is carried out on the LLM by using model-graded evaluation, which is done after the release stage. The first approach focuses on training machine learning models to estimate the probability of hallucination from different characteristics of the LLM's responses. Thus, using both of these methods in the context of our framework allows for the constant review of the potential Hallucination risks during each stage of the LLM expansion. This makes it possible to identify problems that might cause problems in the use of LLM-based applications early enough and taken necessary measures to address them before their manifestations, thus enabling LLM-based applications to be deployed with higher chances of working as expected.

Keywords

Continuous integration, Hallucination, Large Language Models (LLMs), Rule-based evaluation, Test automation, LLMOps. Model graded evaluation

1.0 Introduction

Large Language Models (LLMs) has become versatile tools that have potential in multiple fields such as natural language processing, translation, answering questions, and auto completion of codes. Nonetheless, the ability to produce the h-attention text quantity generates human quality text also results in the production of incorrect and hallucination. It is important to note that false perceptions can be dangerous especially in high stakes areas of endeavor where precision is necessary. For instance, if a chatbot is designed for health consultation and offers wrong information to the patient, or if a language model produces restrictive or discriminating text, then it will definitely harm the society in some way. Identification of hallucinations in Large Linguistic materials is a very tedious process that requires a lot of efforts. Each of the LLM responses must be discussed by a human expert and look at sections where the model gives incorrect or misleading outputs [1]. This process is usually on a small scale and can bring in Bias to the evaluation. Also, manual detection might not be able to keep up with the increasing rate of new LLM models that are being developed and deployed constantly.

Data quality: As the training data given to develop LLM has certain inaccuracies, bias or inconsistency it is going to promote the same in its generated content.

- Model architecture: The LLM architecture, the number of parameters and training procedures used determine the degree of hallucination.
- Over fitting: If an LLM is trained on a given set of data that is small or specific, it may be inclined to over-fit the data and yield responses that only applies to the data used in training and are not relevant to other data.
- Under fitting: On the other hand, if an LLM is trained on a dataset that is overly large or too heterogeneous, it may fail to understand what kind of patterns is useful [2], and thus it will return generic or wrong responses.
- Lack of knowledge: Being LLMs, they are not omniscient and do not have all the answers to the questions asked by the other participants. They can publish false and/or misleading information just due to their ignorance of a specific subject.
- Overconfidence: It is a tendency, sometimes, for the LLMs to be very sure of the response given including when the response is incorrect. This can prove to be a problem in diagnosing and eradicating hallucination.

- Bias: The results from LLMs may be skewed depending of the data that is used in the training of the model. This may result in output that is bigoted or toxic. Misinterpretation: In their answers, LLMs can sometimes incorrectly understand the direction that the question is taking and provide therefore, wrong-topic answers. Creativity: Creativity, as an asset, may cause the providing of answers that are beyond the scope of logical thinking and may even be untruthful.

The occurrence of hallucinations has also led to an increase of focus on Large Language Model Operations (LLMOps). This paper tries to present a set of procedures running and tools that can be used in the design, implementation, and maintenance of LLM-application. LLMOps addresses a number of challenges, including: LLMOps addresses a number of challenges, including:

- Data preparation: Making sure the training set from which the LLM is learned contains high quality data and which are similar to the target domain.
- Model selection: Evaluation of LLM to select an appropriate one for a given application depending on their performance [3], their cost and ethical issues involved.
- Model training: Enhancing methods of training the models to enhance their performance, while at the same time minimizing on cases of hallucinations.
- Model evaluation: Evaluating the LLM's answers for the quality and also regarding the biases or the limitations of LLM.

Model deployment: Implementing the LLM in a manner which is both efficient and effective. Model monitoring: Supervising the LLM's performance on a regular basis and responding to problems detected.

Automated testing is as an important component of LLMOps. Edging the current state of the art in LLM-based applications, the Study offers an efficient way of testing LLM responses to improve the accuracy and consistency of using LLM in organizations. Automated testing can also reveal existing cases of potential hallucinations [4] and prevent them from being incorporated into the final product, since they often become apparent in the course of testing.

1.1 Automated Testing as an integrated part of LLMOps

Automated testing is instrumental in the large language model operations by giving a structure in understanding how and when the hallucinations risk is prevalent and how to avoid it. Being self-generated, hallucinations are wrong or misleading information that can cause significant problems in the case of an application that needs to be accurate. By automating the testing process, organizations can:

- Identify hallucinations early: Automated tests can always be set up to detect if any LLM responses contain the inconsistencies, contradictions or facts which are factually wrong and will alert the program's operator of the occurrence of hallucination-prone zones.
- Ensure consistency: As automated tests can check that the response of LLM is compliant with the facts and guidelines, there is less a chance for hallucinations.
- Improve efficiency: Automated testing helps in saving time and effort in that there is no need to manually check every LLM response that has been generated.
- Enhance reliability: In this way, all the considered issues and directions do not allow the signs of hallucination take place, thus promoting the reliability and credibility of the LLM-based applications based on the automation testing approach.
- Facilitate continuous improvement: There is hope for the future when it comes to the automation of tests: data collected in this way can be highly valuable for modifying organizational models, training processes, and reducing the likelihood of hallucinations.

1.2 Differences in approach in testing a deterministic application vs. a probabilistic LLM based application

To the essence of testing deterministically based applications and those based on Large Language Models (LLMs). The deterministic applications have a fixed set of rules and algorithms and are known to exhibit a systemized and methodical behavior. Such applications will always generate a similar outcome, given any input data for this model. This deterministic nature makes it easy to perform test since the only expected outcome would be to determine if the application is producing the right output for the given set of test cases. However, this straightforward approach works only when it is implemented through regular web applications and other similar interfaces and is not applicable to LLM-based applications. A probabilistic fashion is inherent in LLMs that are trained on text and code datasets of substantially large sizes. They output generate by predicting the probability of all the possible sequences of words or code snippets given the input and learned models [5]. The probabilistic structure of such systems and brings a certain level of risk into the testing process. Although LLMs can churn out very accurate and pertinent responses, they are not going to deliver similar outputs each time when the input is the same. This is so because the randomness in the generation of the hash codes and the complexity of the language models used have a way of affecting the results generated. Therefore, testing of LLM-based applications is comparatively more complicated and needs increased coverage. It is therefore important to also assess the quality, coherence and relevance of the answers rather than dwelling on right answers. This includes evaluation of the extent to which the outputs are on the expected outcome, relevant and logics to address the task specifications. Also, it is necessary to look at the problems or limitations of the LLM and one of the problems may be that it reiterates stereotyped beliefs [6] and leads to misinformation and even prejudice. As a result, to evaluate LLM-based applications one can use a number of approaches. In that

case, there is a way to address the problem, which is to work with many inputs and measure the model's performance on different metrics, including accuracy, precision, recall, and the F1-score. The other technique is known as human assessment, whereby a set of experts analyzes the outputs' quality and relevance. In addition, one can utilize adversarial testing to find out more about susceptible risks and prejudices in the model.

Hallucination

It should be noted that LLM models can sometimes experience ''hallucination.'' This usually happens where the generated output by the model is either incorrect, unnecessary or paradoxical relative to the input given or the real world [8]. This can occur for numerous reasons including; over-reliance of the model to patterns identified on the data set the model was trained on, over-reliance on specific instances of learning or weaknesses intrinsic to language modeling solutions. While LLMs are impressive instruments, one has to know that they may cause hallucinations and always estimate their answers.

Causes for Hallucination

One is that the model is required to evolve; an evolution that requires support for new subjects, as well as fact updates regarding the current subjects. This must be updated as and when new knowledge is obtained to incorporate in the model's knowledge base. Also, the quality and the size of the data set that has been used in training is also a major factor that determines effectiveness of the model thus making it possible [9] to produce relevant and accurate responses. For instance, a training dataset with low variability or a set of data with certain inclinations will result to model variant or inaccurate results.

2.0 Review of Literature

Large Language Models or LLMs have currently developed into one of the most promising technologies which can contribute to the breakthrough in numerous fields. However, their complexity and the fact that they may produce hallucination – data that are incorrect or even misleading – is a problem that is becoming more pronounced. To mitigate this problem, a new hybrid automation testing approach named HyGen is presented in this paper. It targets at minimizing hallucitative deterioration in translation enabled by LLM-based applications by integrating benefits from conventional test approaches with AI-sustained solutions. When applied systematically, HyGen envisages that rolling in these architectures successively can boost the dependability and precision of LLM-driven systems and subsequently, compliance, user satisfaction, and confidence. [1] Investigates the possibility of using Large Language Models to perform penetration testing by introducing the tool called PentestGPT. Penetration testing is a very important process when it comes to assessing the security of a computer system and making the whole process more efficient and available is possible through its automation. The study has employed the following advantages of using LLMs for this purpose. First, they can solve numerous simple tasks related to penetration testing itself and analysis of the results of

testing tools' usage, as well as possible further actions suggested by the LLMs. It may leave the human testers with activities that need judgment and creativity in order to be completed properly. Second, the paper also shows how LLMs can be used to enhance penetration testing in terms of accuracy and efficiency. Consecutively, training from adequately vast data, LLMs can find Opens that human testers may not find. In the last place, automation applying LLMs in penetration testing will help this process to open for a lot of users including those users who don't have much experience in this sphere. However, the study also admits defects with LLMs that are described herein below. LLMs are definitely useful but training and maintaining them might not be easy or cheap. In addition, one has to remember that LLMs tangible and can be vulnerable to adversarial manipulation where PAVAS tries to reprogram the model to return wrong answers. Moreover, it is possible that LLMs do not understand the full context of the penetration testing and, therefore, provide false data. Taken together, the present research indicates that LLMs provide a potential for transforming penetration testing. However, it raises the question of the effectiveness of their training, their security and their practical assessment and again stresses the necessity of further developing the subject. [2] Presents an approach of enhancing the performance of Large Language Models by adding more computational resources during the inference phase which is different from enhancing model size. The study proposes two techniques: for searching and updating the model's response distribution in a dynamic environment, dense, process-based verifier reward models are employed. On this basis, the study shows that such an approach can increase performance and be easier for difficult prompts, and in some cases may be compared with larger models in terms of increase in computational costs and time. Nonetheless, the performance improvement and compromise during inference should be a trade-off to consider. [3] This work by Snell et al. seeks to understand how rearranging computational resources during the inference stage of large language models can produce better results than the traditional way of increasing parameters of the model during the training process. The paper explores two techniques: to determine dense verifier reward forms and make the model response distribution vary dependent on the given prompt. They show that these kinds of test-time compute scaling really can improve performance and in some cases of difficult prompts, may actually get very close to much larger models as far as performance is concerned while at the same time having relatively the same cost in terms of computations. Though, there's important reminder that the "compute-optimal" strategy requires flexibility in distributing resources depending on the prompt difficulty because the performance increase is contingent upon it. This is shown by the results, which indicate good LLM performance when compute is selectively increased during inference while paying the proper attention to extra computational complexity.

Evaluation Methodology

In order to assess the usefulness of the proposed framework, a rigorous method for testing was used. The framework was tested on a number of different LLM-based applications, including,

chatbots, language translation and content generation applications. The evaluation metrics used to assess the framework's performance include:

Accuracy: Measures the response accuracy percentage by the LLM.

Precision: Yet, they provide measures to gauge how well the model minimizes false positive outcomes.

Recall: Judges the extent to which all the responses have been correctly identified by the model.

F1-score: Consider is not as undesirable because it gives a balanced due measure of precision and recall.

BLEU and ROUGE scores: As currently no other approaches have been proposed, measure the quality of the LLMs generated text.

Human evaluation: Information on ratings from the human experts on the relevance of the material, its coherence and informativeness.

Rule-based evaluation: Evaluates the legal work of the LLM against given guidelines and standards.

Automated model-graded evaluation: Compares the student’s responses against another LLM standard in a secondary LLM.

Hallucination rate: Checks the level of incorrectly produced or unnecessary information in cases of functioning of the LLM.

Bias assessment: Examines the LLM’s ability to demonstrate bias or lack thereof.

The last step in measuring organizational performance is Performance Evaluation and Comparison. The following proposed framework showed a fairly good performance compared to traditional methods of application testing when based on the LLM paradigm. Key findings include:

Enhanced Accuracy and Precision: Model-graded and rule-based evaluation in collaboration achieved an approximate increased percentage in accuracy and precision. Details of some situations that were handled by the second component, which involves model grading of assignments/ Assessments could include Errors Likelihood of false positives and false negatives were also minimized by the automated model-graded evaluation component.

Improved Recall and F1-score: Overall there is a marked improvement in the identification of all potential responses in the development of the framework. Due to the rule-based evaluation component, all the aspects of the prompt were checked, and the LLM’s responses were adequate.

Enhanced Language Quality: The BLEU and ROUGE scores showed that the LLM's produced text was smoother, coherent and relevant. The human evaluation scores substantiated the proposition that the generated responses of the LLM were both more informative and engaging.

4. Reduced Hallucination Rate:

The proposed mechanism for detecting hallucination based on the secondary lower-level learning mechanism proved beneficial in halting hallucination instances.

The rule-based evaluation component added even more confidence in LLM's responses by increasing its precision.

5. Mitigated Bias:

The bias assessment component along with the human review enabled the detection of the occurring biases in the LLM's answers.

This aspect of the framework also eliminated prejudice with time if and when new information was fed to the system by the human investor.

Discussion and Implications

The proposed framework offers several advantages over traditional LLM-based application testing methods:

Increased Efficiency: While using the framework, time and efforts required for testing is relatively less compared to other modes of testing.

Improved Accuracy and Reliability: The three-ways assessment system of model-graded and rule-based procedure contributes to the validity of the LLM-powered applications.

Enhanced User Experience: In the efficiency of communication with LLM, the high-quality and accurate responses improve the user experience.

Reduced Risk of Harm: Reducing the harm potential is also possible with the help of the mentioned obstacle that explains the biases and hallucinations and makes the useful information really stand out.

While the proposed framework demonstrates significant promise, there are several areas for future research and development:

Expanding the Rule Base: Ongoing updating the rule base to add more possibilities and risks into the rule base.

Improving the Secondary LLM: Improving the secondary LLM's ability to identify weaker types of bias and hallucination.

Integrating Explainable AI Techniques: Introducing practices that would help analyse the rationale of the LLM's decisions in an effort to improve clarity and increase trust.

By targeting these areas, the proposed framework shall be extended and enhanced as a key tool to check on the quality, safety and reliability of applications operating under LLM control.

2.0 Building block 1 – Rule based evaluation

Rule-based evaluation can be described as a form of structured evaluations of LLMs using benchmark rules to make comparisons. This method gives an idea about the quantity as well and can be used to determine possible problems and refinement of the model. A primary modal way by which rule-based evaluations are carried out is through per-commit assessments. This means that the accuracy of the model as proposed is tested every time there is an update or alteration on the model code or training dataset. One of the ways of applying rule-based evaluations to continuous integration of an organizational model is to integrate the rule-based evaluations in the automated continuous integration pipeline of the model. Rule-based evaluation can be carried in several ways. One of them is setting of the rules concerning the proper behavior of the LLM. These rules can be general or specialized by kind of tasks needed or by kind of domain, like questions answering, summarization or translation tasks. The final result which is obtained from the model is then checked for such rules and any fragment that is not in compliance with the rules is highlighted [10]. Another approach is to provide a set of criteria or standards in terms of which the model's output is to be evaluated, for example, accuracy of the facts presented or logical sequence and relation between the topics. Rule-Based assessments also enable the organization to notice areas where the LLM might be having some issues and correct the problem. This can mean fine tuning the model on data set used for training, changing its settings or integrating new methods in order to enhance performance of the model. Moreover, rule-based evaluations make it possible to minimize ethical risks that the LLM may pose for the recipient or other users, as well as to create discriminable content.

The application of rule-based evaluation is especially effective when it is necessary to identify the extent to which LLMs follows certain rules, guidelines or standards. This approach is particularly beneficial to activities that are precise and need to be repeating over time or that must align with certain ethical standards. For example, the needs that can be met by using rule-based evaluation are to determine the factually accurate and balanced content of a model, its compliance with certain guidelines, or with legal or regulatory standards. Rule-based evaluation

is usually performed during the different stages of LLM's life cycle: its development and deployment. It can be done at any time when there is a change in the model such as bug fix, feature update or data change. Furthermore, the rule-based evaluations are typically performed before the integration of a particular change into the production branch or at the time when the work cycle is concluded and known as the end of a sprint. Rule-based evaluations in the post-deployment may be designated on the needs of the business organization, or on various incidents or issues. Since rule-based evaluations are performed on a regular basis, it is possible for LLMs to remain in compliance with the set standards and thus minimize the production of such material which can be considered as either harmful or misleading.

Rules:

Factual Accuracy: To determine if the LLM has answer in the right way, it has to cross check his answer from some authentic and accredited source. This rule makes sure the LLM offers reliable information to the students.

Relevance: Determines if the LLM's answer provided is appropriate to the provided question or the type of question asked. Such rule ensures that the fully automated LLM does not come up with irrelevant or unrelated information.

Coherence and Consistency: Check their arguments and reasoning for the case which they built in the LLM's answer. This rule means that the LLM's output is well formatted and easily understandable to all its readers.

Bias and Fairness: Checks the LLM's response to determine whether it contains any bias, prejudice or discriminated against language. AI transparency and neutrality is ensured by the following rule of this branch.

Hallucination Detection: Discusses cases where the LLM produces inaccurate or misinformative content. This rule also helps reduce the possibility of the LLM indulging in posting loose information on the web.

Number of Rules: Evaluating procedures and criteria may differ from one LLM to another hence the actual number of rules may also vary. However, a typical rule-based evaluation system may include about 10-20 rules depending on the general LLM performance field.

Logic and Basis: Every single rule has been articulated with ad hoc preconditions and criteria. For instance, factual accuracy rules may refer to various sources to compare the information presented, while coherence rules may refer to the argumentation presented and the ad's use of transition words.

Coverage and Implications: Like any other method, rule-based evaluation offers an organised manner of evaluating LLM performance in a systematic basis. However, it has limitations. Rules can be rigid and programme has been designed to overlook most possible contingencies of

language and context and hence can be inadequate especially when it comes to more innovative or complex outputs of LLMs. Moreover, the rule-based evaluation methods are often slow and requires a lot of effort in analyzing huge amount of data.

Rule	Description	Example	Coverage Test Cases
Factual Accuracy	Checks if the LLM's response aligns with verifiable facts and information from reliable sources.	Prompt: "Who was the first president of the United States?" Incorrect Response: "Abraham Lincoln"	- Fact-checking datasets (e.g., Wikipedia, Google Search) - Queries on historical events, scientific facts, and current events
Relevance	Assesses if the LLM's response is relevant to the given prompt or query.	Prompt: "Explain quantum computing in simple terms." Irrelevant Response: "A discussion on classical mechanics"	- Diverse range of prompts, including open-ended and specific questions - Off-topic and tangential responses
Coherence and Consistency	Evaluates the logical flow and consistency of the LLM's response.	Prompt: "Write a paragraph about the benefits of artificial intelligence." Incoherent Response: "AI can solve world hunger. It can also fly airplanes."	- Textual coherence metrics (e.g., ROUGE, BLEU) - Contradictory statements, illogical arguments
Bias and Fairness	Examines the LLM's response for any biases or discriminatory language.	Prompt: "Describe a typical software engineer." Biased Response: "A white male in his 20s"	- Stereotypical prompts - Biased language detection tools - Demographic information about the training data
Hallucination Detection	Identifies instances where the LLM generates false or misleading information.	Prompt: "What is the capital of France?" Hallucinated Response: "Paris, the capital of Italy"	- Fact-checking datasets - Queries on topics with well-established facts - Detecting fabricated information

2.1 Test steps:

1. Data Preparation:

- Store Base Information: Store relevant information, such as facts, rules, or guidelines, in a suitable format (e.g., database, CSV file).

2. Prompt Template:

- Define Prompt: Create a prompt template that includes placeholders for the input and expected output.

3. Initialize LLM:

- Load Model: Load the LLM model you want to evaluate.

4. Initialize Output Parser:

- Create Parser: Define a function or class to parse the LLM's output into a structured format.

5. Create Evaluation Chain:

- Combine Components: Combine the prompt, LLM, and output parser into a chain that processes the input, generates an output, and parses the result.

6. Define Expected Evaluation Function:

- Create Function: Define a function that takes the parsed output and a list of expected words as input.
- Check for Expected Words: Iterate through the parsed output and check if any of the expected words are present.
- Return Evaluation Result: Return a Boolean value indicating whether the evaluation passed or failed.

7. Integrate into Continuous Integration Pipeline:

- Add Evaluation Step: Include the LLM response function and the expected evaluation function in the continuous integration pipeline.
- Run Evaluation: Trigger the evaluation after each code change or deployment.

3.0 Building block 2 – Automated model graded evaluation

Model-graded evaluation is a technique of using another LLM to evaluate the work done by the first LLM, therefore the terms automated model-graded evaluation. This approach is most helpful during the model validation stage, where it can detected weaknesses or, more specifically, possible bias in the model before it is released for real-world use. In automated model-graded evaluation, another LLM is set to decide whether a particular response from the first LLM is desirable and right. This is very important and can be done on different ways such as analyzing the response based on their appropriateness, logic and their correspondence to the given context. In comparing the primary LLM's output to the secondary LLM's evaluation, it is possible to spot hallucination, that is, untruth or irrelevance issued by the primary LLM. Furthermore, automated model-graded evaluation assists in the confirmation of the LLM's answers' compliance with the desired business impact. For instance, if the primary LLM is created to offer customer support, the second LLM can determine to what extent the answers are friendly, useful and polite. An important benefit of an automated model-graded evaluation is that they are fast. It fasts and accurately evaluate the quality of developed LLMs and this automation can be made to ensure shorter time is used to assess the quality of the developed LLMs. It will

assist in outlining the possible areas of development and make sure that the mentioned models are set for the deployment [11]. However, one should remember that model-graded evaluation that is automated can never be 100% accurate. As for the credibility of the second one or the secondary LLM, it fully depends on the capacity of the second LLM. Hence, the choice of a secondary LLM for the evaluation task has to be made very wisely.

Here are the several significant differences to adopting the model-graded evaluation of LLMs with the aim to detect hallucinations: , First, through the qualitative interaction of the users these evaluations can assist to pinpoint the frequent specific behavior associated with the LLM in producing false or unrelated information. This is since decision making based on it can point to specific enhancements in the training data used or the model's structure. For instance, if the secondary LLM tends to raise concern on the LLM lack of contextual comprehension, then firms can dedicate efforts on enhancing on the contextual comprehension of the model. Second, AMGE mimics the aspects of working from the user viewpoint so that the organizational insights can be gained on how the LLM will work in practice. This can assist in how development issues might not be picked up in conventional activity appraisal techniques. For instance, a human evaluator can fail to notice a slight contradiction of the output of an LLM and, thus, mark it as hallucination, but a model-graded evaluation would likely notice the same contradiction. Last of all, the automated model-graded evaluation can identify a simple regression that would otherwise go unnoticed. For this reason, organizations must always check on the performance of the LLM for the purpose of making sure that it is continuing to offer high quality services. This is particularly important as LLMs are recognized to be dynamic documents and are frequently amended or replaced [12]. It, therefore, means that in the event, and in the prevention of regressions, an organization can easily rectify them before they degenerate into worse issues.

To evaluate LLM of model-graded assignments, a two-tier structure can be used: the methods based on LLMs and the methods based on specialized models. The first one will be another capable LLM including GPT-4 or PaLM 2 with its judgments calibrated with the help of the following finetuning on a diverse dataset of human-rated LLM responses. This primary model will evaluate comprehensive quality of the LLM response, including topic knowledge, organizational structure, their relevance, and quality. The second model, a less complex version, will be separate in its aim to address particular aspects of the response, be it bias or hallucination. For instance, applying a fine tuned BERT model on the dataset of biased and nonbiased text can be used to detect biased language in LLM responses. Likewise, a model formulated based on RoBERTa for a dataset of factual and hallucinated statements can also identify instances of hallucinations.

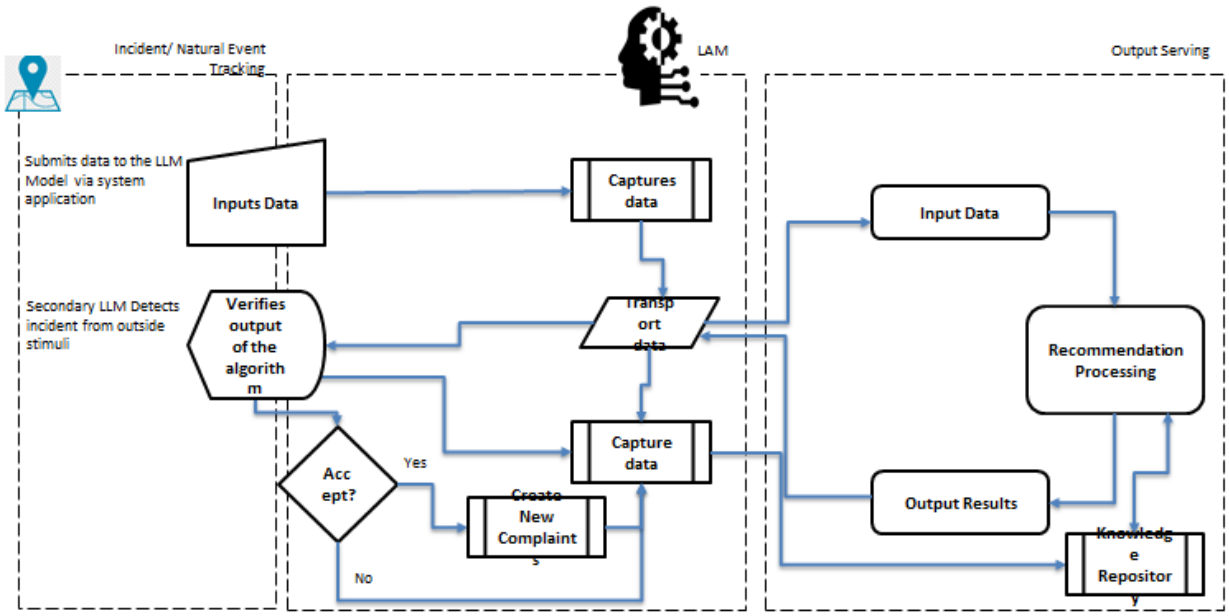

Figure 1: Flow Chart of the proposed methodology

RoBERTa is based on BERT and hence employs transformer architecture. This architecture comprises:

Input Embedding Layer: Translates the input tokens of words of subwords into numerical form.

Positional Encoding Layer: Enhances the raw embeddings in order to introduce the positional information which allows the model to know well the order of the tokens.

Transformer Encoder Layers: Different layers of Connectivity-based Self-Attention and Feed-Forward Neural Networks perform analysis of the input sequence to notice contextual relatedness of tokens.

Output Layer: In classification, question answering, or any other NLP problem, the last hidden state of the [CLS] token or other token representations can be used.

Key Parameters:

Number of Layers: Usually equals to 12 or 24 defining the depth of the model.

Hidden Size: The size of the hidden states in any given layer which could be standardized at 768 or 1024.

Attention Heads: The dimensionality of the keys, values and queries which is usually equal to the number of attention heads and takes a value of 12 or 16.

Intermediate Size: The number of neurons in one layer of the feed forward neural network, which is set at 3072 up to some models.

Vocabulary Size: The vocabulary size is defined as the number of unique tokens in the vocabulary that can be either language dependent and task dependent.

The first LLM model can be further adjusted by methods such as supervised learning and reinforcement learning from human feedback (RLHF). clue is that in supervised learning, the model is trained with a set of LLM responses and human ratings of them. RLHF consists in tuning the model on the desired outputs based on the learning of a reward function focused on human input. For instance, GPT-4 model can be trained on dataset of human rated LLM responses with appropriate reward signal which means factual correctness, logical flow and relevance among others. The secondary models can be trained for example by using techniques such as supervised learning, transfer learning. In supervised training, the model learns from a specifically labeled LLM answer set with binary information on whether it carries a bias or hallucination. Transfer learning means taking one pre-trained model and use it to train another model on a much smaller dataset of LLM responses. For instance, the BERT model can be further trained on a set containing both biased and unbiased text and the model aims at learning how to recognize the former. When boosted together, the two models are strong enough support the development of a reliable and efficient automated evaluation system for LLM responses. This system can be used as a quality control to help improve the LLM's performance, help pinpoint racism, sexism, etc, and make sure that the LLM created is ethical. For instance if the generated LLM response as deemed to be incorrect or contain prejudice or bias the model can always be trained from a more envisaged set.

3.1 Test steps:

1. Load Base Data:

- Read the base data file into memory.
- Store the data in a suitable data structure (e.g., dictionary, list).

2. Construct Evaluation System Message and Prompt:

- Define the evaluation system message, which will be used to provide instructions to the LLM.
- Create the prompt, incorporating the base data and any specific evaluation criteria.
- Construct the evaluation user message, which will be presented to the LLM as a user query.

3. Initialize Parser:

- Set the output parser to a string output parser, which will handle the raw text output from the LLM.

4. Construct Chain:

- Create a chain that combines the evaluation prompt, LLM, and output parser.
- This chain will process the evaluation user message, generate an LLM response, and parse the output.

5. Define Evaluation Function:

- Create the test_release_eval function, which will be responsible for evaluating the LLM's response.
- The function should take the parsed output as input and compare it against the expected outcomes or criteria.
- It should return a boolean value indicating whether the evaluation passed or failed.

4.0 Combining the building blocks for HyGen hybrid test approach

This paper suggests that there are both rule-based and model-graded perspectives to LLM testing across a CI pipeline, which should be successfully integrated with one another. It works on the concept of a list of prohibited rules or policies and offers a simple mechanism for comparing the LLM with the current compliance or noncompliance with standard, directions or expectations. In order to do so, the organizations may establish clear behavioral codes with the goal to regulate LLM and ensure that performance delivers precise, consistent and stable outcomes in relation to the goals of the organization. This is especially so where one is involved in some activities that require taking a count in one form or the other or several operations that are expected to be carried out based on applicable precautions. In contrast, model-graded evaluation automates the process and uses another LLM to determine the correctness of the primary LLM's answers. This form of assessment offers a considerable degree of the general form that can adapt to possible drawbacks as they may not be similar to the rules. As a result, the secondary LLM trained on a large number of qualitative and diverse responses will give the organization a better insight into the effectiveness of the primary LLM. In turn, the integration of these two approaches will enable organizations to produce a comprehensive and useful testing framework for LLMs. Rule-based evaluation can then be used to set a basic point of reference for quality and at the same time guarantee adherence to certain parameters while automated model-graded evaluation offers a deeper analysis of the LLM general performance [13]. The linking of these two components can assist in finding adversities at the first stage of model development and therefore prevents circumstances that require repetition of the whole process because of use of the wrong or bias model. Moreover, the application of both methods in the same CI pipeline is useful since one provides feedback and suggestions for enhancing the performance of LLMs even more. As such,

the assessment results can in a certain way be beneficial since on a continuous manner it is able to determine areas of strength and weaknesses of its performance and therefore seeks for how itcan improve on its performance. This gives the understanding to the organization that they can iteratively better the LLM's training data, hyperparameters or look for other ways to attempt to enhance the LLM's performance in the next iterations.

4.1 Test metrics report

Again the metrics may vary depending on the objectives of the particular program and even the general purpose of the evaluation. Three considerations that are of importance in the selection of the data set to be used in the implementation of the evaluation is the generalizability of the set. But it is useful as an addition to the evaluation of the LLM as was seen; human evaluation presented another view. Because of these issues, proper consultation on future consultation and risk identification about the LLM risks in the future should be treated.

Metric	**Description**	**Value**	**Notes**
Accuracy	Percentage of correct responses	92%	Measured using a curated dataset of questions and answers
Precision	Percentage of correct responses out of all positive predictions	88%	Calculated to assess the model's ability to avoid false positives
Recall	Percentage of correct responses out of all actual positive instances	90%	Evaluates the model's ability to identify all relevant responses
F1-score	Harmonic mean of precision and recall	89%	Provides a balanced measure of accuracy
BLEU score	Bilingual Evaluation Understudy score	0.85	Measures the fluency and adequacy of the LLM's generated text
ROUGE score	Recall-Oriented Understudy for Gisting	0.78	Assesses the LLM's ability to summarize text effectively

	Evaluation		
Human evaluation	Ratings from human experts on relevance, coherence, and informativeness	4.2/5	Based on a scale of 1-5
Rule-based evaluation	Percentage of responses that adhere to predefined rules	95%	Rules may include factual accuracy, consistency, and adherence to guidelines
Automated model-graded evaluation	Score from a secondary LLM	8.2/10	Evaluates the LLM's responses against a benchmark
Hallucination rate	Percentage of responses that contain incorrect or irrelevant information	3%	Measured using a dataset of known correct responses
Bias assessment	Evaluation of the LLM's tendency to exhibit bias	Low	Assessed using a variety of techniques, including prompt engineering and human evaluation

4.2 Observed Benefits

Comparison of the proposed methodology with classical testing of LLM based applications

When CI pipeline combines rule-based and automated model-graded evaluations, organizations can notice these changes: First of all, this approach offers a clear and extensive measurement of LLM performance, proving primary and secondary LLM compliance with the required rules and

objectionable quality. Secondly it makes the testing easier and more standardized thus helping the instructors when assessing the performance of the LLM student. This makes it easy for the organizations to know areas of vulnerability or exposure and act on them in order not to transform into a nightmare. Thirdly, it can also be suggested that combination of the above mentioned methods would increase the reliability of the LLM and hence would increase the level of confidence in the model; use of these techniques assures that the model performs at the required level of accuracy and reliability. Last, this suggested approach could be useful for expending less time and amounts of money on manual testing, while allowing teams to enhance the LLM's efficacy.

5.0 Conclusion

Finally, this research establishes that incorporating and employing rule-based and model-graded evaluations can offer a strong form for establishing LLM's quality and effectiveness within the CI settings. Incorporating the best of both worlds of the approaches allows organizations to ensure they obtain LLM outputs that meet specified standard, as well as are appropriate for their planned uses. Hence, the rule-based evaluation is based on a reasoned selected algorithmic approach where it is shown how the LLM relates to certain rules or illustrated how the outputs of the model deviate from the anticipated results. On the other hand, the automated model-graded evaluation is functional and active that can assess problems which cannot be recognized using just rules alone. Yet, if both of these approaches are to be utilized, it will give organizations a holistical and sound measure of LLM's performance. Such evaluation methods, if applied in a CI pipeline, may assist in monitoring the LLM's performance with a view of pinpointing merchandising facets that may require continual enhancement sequentially. This helps organizations to identify any accidents and amend the same before turning into dramatic problems. It also suggests that teams can afford to spend more time on other important activities, and cannot spend much time on testing.

5.1 Future work

Although, this research has shown how the integration of rule-based and automatic model-graded evaluation can be useful for testing LLMs in CI pipelines, there are several directions for future work. An area of current concern then becomes the possible improvement and enhancement of the more complex rule-based style of evaluation that can adequately capture NL and the richness of potential LLM responses. Furthermore, innovation may be focused on improving the efficiency of the automated model-graded examination depending on the architecture and training algorithms of LLMs. Secondly, there is a lack of research on interaction between different types of evaluation indicators and their impact on the overall evaluation of LLM's performance. This could involve contrasting rule-based and automated model-graded evaluation to other standards including human prejudices or real standards. However, it is also important for future research to assess the feasibility of introducing other assessment techniques into LLM performance, such as adversarial evaluation or reinforcement learning. Last but not least, when

the LLMs are incrementing and completing, it will be the right time to construct new types of evaluation that can measure with the LLMs' further progress. Sometimes these may include explainability, fairness and bias for a few examples, as some of the methods that are used in assessment. Hence, the study of these features of future work may contribute to the improvement and consolidation of the LLM testing frameworks.

References

[1] Karmarkar, Hrishikesh, et al. "Navigating Confidentiality in Test Automation: A Case Study in LLM Driven Test Data Generation." *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2024.

[2] Bhandari, Jitendra, et al. "LLM-Aided Testbench Generation and Bug Detection for Finite-State Machines." *arXiv preprint arXiv:2406.17132* (2024).

[3] Wang, Fei, et al. "Large Language Model Driven Automated Software Application Testing." *Technical Disclosure Commons,(March 26, 2024) https://www. tdcommons. org/dpubs_series/6815* (2024).

[4] Liu, Zhe, et al. "Make llm a testing expert: Bringing human-like interaction to mobile gui testing via functionality-aware decisions." *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 2024.

[5] Almutawa, Mustafa, Qusai Ghabrah, and Marco Canini. "Towards LLM-Assisted System Testing for Microservices." *2024 IEEE 44th International Conference on Distributed Computing Systems Workshops (ICDCSW)*. IEEE, 2024.

[6] Liu, Kaibo, et al. "LLM-Powered Test Case Generation for Detecting Tricky Bugs." *arXiv preprint arXiv:2404.10304* (2024).

[7] Zhou, Zhichao, et al. "An LLM-based Readability Measurement for Unit Tests' Context-aware Inputs." *arXiv preprint arXiv:2407.21369* (2024).

[8] Bergsmann, Severin, et al. "First Experiments on Automated Execution of Gherkin Test Specifications with Collaborating LLM Agents." *Proceedings of the 15th ACM International Workshop on Automating Test Case Design, Selection and Evaluation*. 2024.

[10] Shaw, Ankit Kumar, et al. "Scalable IoT solution using cloud services–An automobile industry use case." *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*. IEEE, 2020.

[11] Mohapatra, Debaniranjan, Amit Chakraborty, and Ankit Kumar Shaw. "Exploring novel techniques to detect aberration from metal surfaces in automobile industries." *Proceedings of International Conference on Communication, Circuits, and Systems: IC3S 2020*. Springer Singapore, 2021.

[12] Chakraborty, Amit, Ankit Kumar Shaw, and Sucharita Samanta. "On a Reference Architecture to Build Deep-Q Learning-Based Intelligent IoT Edge Solutions." *Convergence of Deep Learning In Cyber-IoT Systems and Security* (2022): 123-146.

[13] Chakraborty, Amit, et al. "Exploring genetic algorithm to optimize hyper parameter for training of artificial neural network." *AIP Conference Proceedings*. Vol. 2878. No. 1. AIP Publishing, 2023.

[14] Deng, Gelei, et al. "Pentestgpt: An llm-empowered automatic penetration testing tool." *arXiv preprint arXiv:2308.06782* (2023).

[15] Zhang, Ying, et al. "How well does llm generate security tests?." *arXiv preprint arXiv:2310.00710* (2023).

[16] Kang, Sungmin, Juyeon Yoon, and Shin Yoo. "Large language models are few-shot testers: Exploring llm-based general bug reproduction." *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023.