ChatGPT Incorrectness Detection in Software Reviews

A1 - Group 5

Nafis Nahian - 2105007
Sk. Ashrafuzzaman Nafees - 2105008
Abrar Zahin Raihan - 2105009

Bangladesh University of Engineering and Technology

January 6, 2025

Contents

List of Figures

List of Tables

Abstract

The paper *ChatGPT Incorrectness Detection in Software Reviews* by Minaoar Hossain Tanzil, Junaed Younus Khan, and Gias Uddin presents a survey of 135 software engineering (SE) practitioners to investigate the use of Generative AI-based chatbots, such as ChatGPT, for SE tasks. It is revealed that ChatGPT is desired for tasks like software library selection, although concerns about the truthfulness of its responses are frequently expressed. A suite of techniques and a tool named CID (ChatGPT Incorrectness Detector) is introduced to automatically identify incorrect ChatGPT responses. CID employs iterative prompting by generating contextually similar but textually divergent questions, leveraging metamorphic relationships in text. The principle underlying CID is that a response differing significantly from other responses for the same question is likely incorrect. In a benchmark study focusing on library selection, it is demonstrated that CID detects incorrect responses with an F1-score ranging from 0.74 to 0.75.

1 Introduction

The recent advancements in pre-trained large language models (LLMs), such as ChatGPT[15], have revolutionized various professional domains, including software engineering (SE). These models, with their interactive interfaces, are being increasingly used as personal assistants by SE practitioners for tasks such as code generation[7][11][13], program repair[16][17], and code summarization[2][10]. Despite their potential, LLMs often generate inaccurate results[3][5], raising concerns about the reliability of their outputs. This uncertainty has prompted developers to rely on manual verification methods, such as consulting additional online resources or engaging in follow-up queries with ChatGPT, to ensure the accuracy of the information provided.

Recognizing the need for automated tools to address these challenges, this paper explores the development and application of the ChatGPT Incorrectness Detector (CID). CID utilizes an innovative approach of iterative prompting, grounded in metamorphic relationships, to identify inconsistencies in ChatGPT's responses. By analyzing the variability across contextually similar but textually divergent queries, CID effectively highlights instances of incorrectness. The paper examines the design principles, implementation, and evaluation of CID, demonstrating its capability to detect inaccuracies in ChatGPT responses with significant precision, particularly within the context of software library selection—a critical yet complex task for developers.

2 Survey to Assess Software Developers' Perspectives on ChatGPT Usage

2.1 Survey Setup

To gain insights into software engineering (SE) practitioners' use of ChatGPT for various tasks, a comprehensive survey was conducted. This survey aimed to answer three key research questions: why developers use ChatGPT, how much they rely on its responses, and how they verify the information provided. The design and implementation of the survey were carefully planned to ensure both the relevance and reliability of the collected data.

2.1.1 Survey Questions

The survey consisted of 10 questions designed in a semi-closed format to allow participants to provide structured responses while also having the flexibility to share additional feedback. These questions were developed after conducting a pilot survey with 24 industry professionals, whose feedback was instrumental in refining the final questionnaire. The questions covered general usage of ChatGPT, specific applications in SE tasks, the perceived reliability of its responses, and potential strategies for improving its reliability.

Table 1: Survey questions and their mapping to the Research Questions. Here, C/O=Close/Open-ended question, G/S=Generic/Scenario-based question. For scenario-based questions, we used library selection as a case-study.

Q#	**Questions**	**O/C**	**G/S**	**RQ**
1	Did you use ChatGPT?	C	G	1.1
2	In general, which of the cases you used it for?	C	G	1.1
3	As a software professional, how did you or can you use it?	C	G	1.1
4	How would you describe your experience with using it so far?	C	G	1.1
5	How much do you rely on the content/response of ChatGPT?	C	G	1.2
6	Have you considered using ChatGPT to select or compare software libraries? Please share the pros and cons.	O	S	1.2
7	How much would you rely on ChatGPT's response for the given library selection query?	C	S	1.3
8	Would you rely on the ChatGPT's response after further inquiry?	C	S	1.3
9	Do you think the opinion from ChatGPT is correct?	C	S	1.3
10	What can be the ways to improve the reliability of ChatGPT responses?	C	G	1.3

2.1.2 Participants and Sampling

A total of 135 SE practitioners participated in the survey. Their professional roles spanned software engineers, developers, managers, and other relevant positions. Participants were selected using a snowball sampling approach[8], starting with 12 senior professionals with over 15 years of industry experience. The second batch of participants included 123 individuals ranging from developers to managers. We deliberately avoided open social or professional networking sites as a means of participant recruitment. These individuals were asked to provide feedback and invite additional candidates from their professional networks. This method ensured a diverse yet qualified pool of respondents. To maintain data quality, open social media platforms were avoided for participant recruitment due to concerns about respondent qualifications and expertise.

2.1.3 Ethical Considerations

The survey was conducted following approval from the Research Ethics Board of the associated university, ensuring adherence to ethical research practices. Anonymity and confidentiality were maintained throughout the process to encourage honest and unbiased responses from participants.

2.1.4 Demographics

The participants represented a wide range of experience levels, with most having between 0 and 15 years of professional experience. Their distribution across various roles and experience levels highlights the comprehensive nature of the survey, capturing diverse perspectives on the use of ChatGPT in SE tasks.

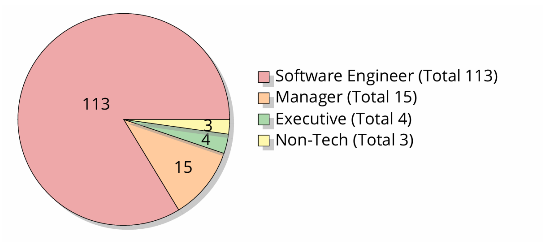


Figure 1: Distribution of participants by current profession.

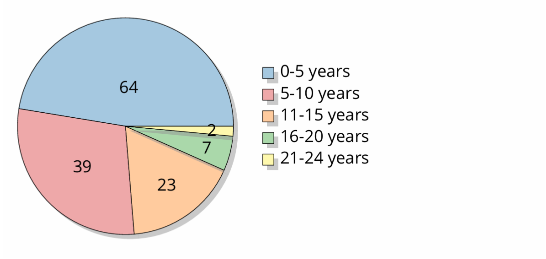


Figure 2: Distribution of participants by years of experience.

2.2 Reasons for Using ChatGPT (RQ1)

Participants were asked about their reasons for using ChatGPT in various software engineering and personal contexts. The responses revealed several key applications, which include:

- **Problem-Solving Support:** Using ChatGPT for troubleshooting and debugging code issues to improve efficiency during development tasks.
- **Learning New Concepts:** Leveraging ChatGPT as an educational tool to understand technologies, algorithms, and concepts related to software engineering.
- **Generating Content:** Employing ChatGPT to create documentation, presentations, and other forms of technical or non-technical content.

- **Exploring Alternatives:** Asking ChatGPT for alternative solutions, methods, or approaches to coding and software design challenges.

The findings indicated that ChatGPT has become a widely adopted tool among SE practitioners. A significant portion (86.67%) of participants used it for problem-solving, while others found it useful for learning (61.48%) or content creation (46.67%). Furthermore, some participants (42.22%) engaged with ChatGPT recreationally, demonstrating its appeal as a conversational AI system.

These results underline ChatGPT's versatility and its growing role in enhancing productivity, learning, and creativity in software engineering and beyond.

2.3 Concerns About ChatGPT Responses (RQ2)

Participants were asked to share their concerns regarding the responses generated by ChatGPT. The findings highlighted three primary areas of apprehension:

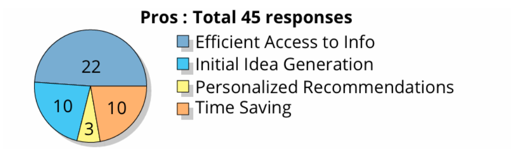
- **Accuracy of Responses:** Concerns about the correctness of ChatGPT's responses were widely reported. Participants noted that the AI sometimes produced incorrect or misleading information, which could lead to potential issues during software development.

- **Lack of Context Understanding:** Several participants mentioned that ChatGPT occasionally failed to fully comprehend the provided context, resulting in responses that were either irrelevant or incomplete.

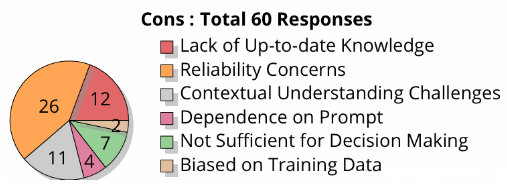
- **Overconfidence in Outputs:** Many participants observed that ChatGPT often presented its responses with high confidence, even when the information was inaccurate or speculative. This could mislead developers into trusting the outputs without further verification.

The findings revealed that although ChatGPT is widely used and appreciated for its utility, its responses are not immune to flaws. A significant portion of participants (54.81%) found the responses somewhat reliable but sought further validation through additional searches. Popular mediums for verification included Google and Stack Overflow (73.33%), where developers cross-checked the information provided. Additionally, 68.15% of participants stated that they would ask ChatGPT more follow-up questions to clarify uncertainties.

These concerns emphasize the need for developers to critically evaluate ChatGPT's outputs, ensuring that its responses are verified before being integrated into professional or critical tasks.



(a) Pros of ChatGPT (RQ2)



(b) Cons of ChatGPT (RQ2)

Figure 3: Participant perspectives on the pros and cons of ChatGPT responses.

2.4 Verification of ChatGPT Responses (RQ3)

Participants were presented with conversations with ChatGPT, where it was asked to provide:

- *Suggestions*: A recommendation for a good library to use for natural language processing (NLP) tasks.
- *More details about a specific situation*: An explanation of SpaCy's API and documentation to clarify how it can be utilized in various NLP applications.
- *Reliability in software engineering real-world situations*: An interpretation of how reliable SpaCy is for real-world software engineering tasks and its practical usability in such scenarios.

The findings were that the majority (54.81%) consider ChatGPT responses somewhat reliable and seek further validation. When verifying them, most developers (73.33%) conduct further searches in common mediums like Google or Stack Overflow to gather additional information. Many participants (68.15%) would ask ChatGPT more questions to assess its responses further.

3 CID: An Automatic ChatGPT Incorrectness Detector

Survey participants desired LLMs to closely approximate human capabilities, though humans themselves can be inconsistent, particularly when lying or bluffing. Practitioners, such as police officers, often view consistent statements as truthful and inconsistencies as deceptive. Techniques like increasing cognitive load with unexpected questions effectively expose deception in humans. Similarly, a recent study showed such methods enhance LLM reliability, inspiring the development of tools like CID (ChatGPT Incorrectness Detector), which uses iterative prompting to detect inconsistencies in ChatGPT's responses.

In a real case, an interrogator might: 1) question the suspect about the specific case, 2) challenge the suspect to uncover inconsistencies, and 3) evaluate the suspect's responses. Similarly, CID consists of three components. The ENQUIRER seeks the initial reasoning behind ChatGPT's responses, the CHALLENGER poses a series of probing questions, and the DECIDER uses machine learning (ML) techniques to assess the correctness of ChatGPT's responses. These components are discussed in detail below.

3.1 Enquirer

Suppose we ask a question, receive a response, and want to evaluate the correctness of the response (base-response, R_B). When ChatGPT responds to any question, it typically provides multiple pieces of information, some of which are accurate and others are not. The ENQUIRER asks ChatGPT to provide separate explanations (E_i) for each piece of information in the base-response using the following prompt:

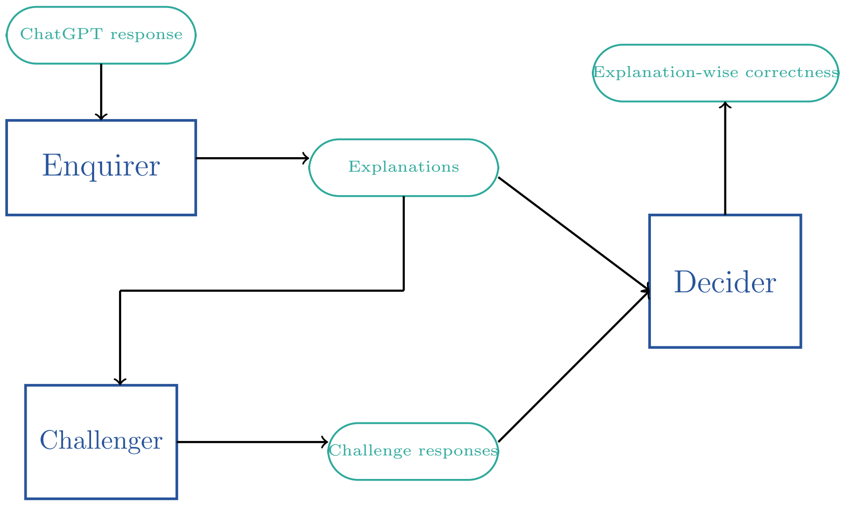


Figure 4: The structure of the CID framework, illustrating the roles of the Enquirer, Challenger, and Decider components.

Enquiring ChatGPT

Justify your answer. If the answer has multiple pieces of information, provide separate reasoning for each of them.

3.2 Challenger

For every piece of explanation (E_i) obtained in the ENQUIRY phase, we further ask a set of challenge-questions (Q_C) to ChatGPT through our CHALLENGER component and record ChatGPT's responses (challenge-responses, R_C) to them.

It has two phases. In the first step, the Challenger generates a dynamic basic challenge question for a previous explanation. In the second step, it asks the generated question to ChatGPT. In the third step, it mutates the challenge, and finally, in the fourth step, it asks the mutated challenge again to ChatGPT.

- **Basic Challenger:** Three basic challenge questions, *Why?*, *How?*, and *Really?*, are asked to ChatGPT for each explanation (E_i) of its base-response (R_B). The questioning process is automated by leveraging a separate LLM that is completely unrelated to the ChatGPT instance being evaluated by the CID tool. To replicate a separate LLM, a new session of ChatGPT is used. The memory of the previous conversation performed during the Enquiry phase is discarded before generating the challenge questions. This is done to ensure that any potential bias induced by the ENQUIRER memory is eliminated.

- **Mutation Challenger:** Liars can often be perceived as being as consistent as truth-tellers because the interview is prepared for by anticipating questions and rehearsing responses. However, the interview may be made more cognitively demanding for liars by the interviewer, which disrupts the preparations of liars and makes it difficult for spontaneous responses to be provided by them. Similarly, if confirmatory questions are simple, insufficient challenge might be posed to the LLM, meaning consistent answers could be given even when a wrong base-response or explanation is involved. Therefore, Mutation Challenger is used to further modify or complicate the challenge questions so that greater difficulty can be introduced for ChatGPT.

The Mutation Challenger utilizes the sentence-level metamorphic testing technique QAQA, as proposed by [18]. This method enhances the original (basic challenge) question by embedding a redundant sentence as a clause, thereby generating a mutated question. This added complexity increases the cognitive load on ChatGPT, further challenging it to expose inconsistencies in its responses. Depending on the source of the redundant sentence, the mutation challenger applies two types of metamorphic relation (MR): Equivalent Question (MR1) and Equivalent Test Integration (MR2).

While MR1 chooses the redundant information from a pre-defined knowledge base, MR2 extracts it from other basic challenge questions from the input list. In both cases, the information (or the question) is selected based on their cosine similarity with the original basic challenge question. Figure 5 illustrates the Metamorphic Relations (MRs) employed by the Mutation Challenger to generate mutated questions, demonstrating how these transformations increase complexity to challenge ChatGPT.

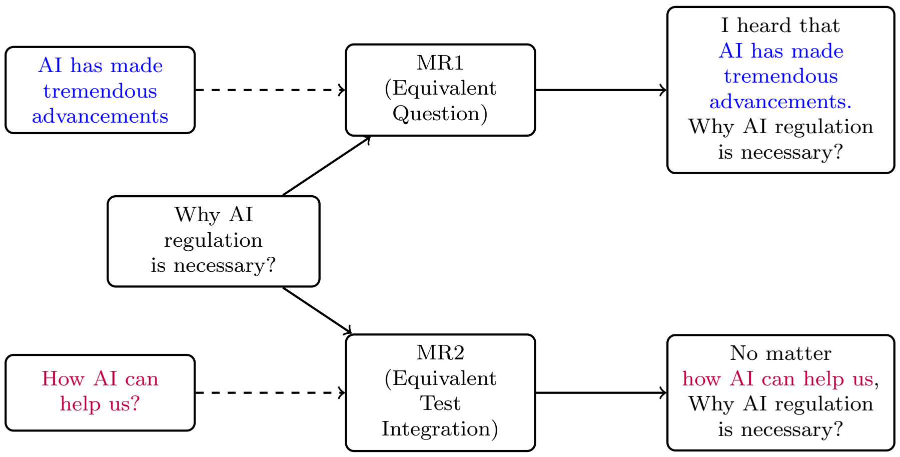

AI has made tremendous advancements
MR1 (Equivalent Question)
I heard that AI has made tremendous advancements. Why AI regulation is necessary?
Why AI regulation is necessary?
How AI can help us?
MR2 (Equivalent Test Integration)
No matter how AI can help us, Why AI regulation is necessary?

Figure 5: Metamorphic Relations (MRs) used in the Mutation Challenger to mutate questions.

3.3 Decider

Decider makes the final call about the correctness of a given response. It has two modules:

- **Inconsistency Meter**: This part of Decider evaluates ChatGPT's inconsistency during the *enquiry* and *challenge* phases using standard similarity metrics.

- **Detection Model**: Decider utilizes a machine learning model to make the final decision on the correctness of ChatGPT's response. This model is trained to learn the relation between ChatGPT's (in)correctness and (in)consistency from a labeled dataset. By doing so, the Decider can effectively detect whether a response is correct or not based on the level of consistency.

3.3.1 Dataset Creation

To train the Decider, specifically its detection model, a labeled dataset is required. The dataset is generated by interacting with ChatGPT and presenting it with various questions. Each question is accompanied by a known correct answer, which serves as the ground truth. For every question in the dataset, ChatGPT's base response is recorded as its initial answer for evaluation. Subsequently, the base response is divided into multiple explanations (E_i) by the Enquirer. Finally, these explanations are manually labeled as correct or incorrect by human annotators.

3.3.2 Inconsistency Meter and Detection Model

Once a labeled dataset with explanations (E_i) marked as correct or incorrect is obtained, the feature extraction process is initiated. Standard similarity scores among ChatGPT responses generated during the *enquiry* and *challenge* phases are computed and used as features for the tool. An ML model is then trained to learn the relationship between ChatGPT's incorrectness (represented by labels) and inconsistency (represented by features). A total of 24 features are considered, divided into four categories.

- Explanation-Response ($E_i - R_C$) Similarity
- Response-Response ($R_C - R_C$) Similarity
- Question-Response ($Q_C - R_C$) Similarity
- Question-Question ($Q_C - Q_C$) Similarity

4 Evaluation of CID

Given that CID can be applied to any computing tasks requiring ChatGPT textual responses, this evaluation focused specifically on software library selection. This choice was motivated by:

- Survey findings showing practitioners' interest in using ChatGPT for library selection
- The non-trivial nature of library selection as documented in SE literature
- The availability of ground truth data through Stack Overflow accepted answers
- The potential impact on software development practices

The evaluation addressed two key research questions:

RQ4. How accurate is CID in detecting incorrect responses?

RQ5. How do base and mutation challenge prompts impact performance?

4.1 Benchmark Study Setup

4.1.1 Dataset Collection Process

The benchmark dataset was created through a structured approach:

1. **Initial Selection:** 100 Stack Overflow posts were selected using:
   - Search interface criteria:
     - Must have an accepted answer
     - Must have tags for target libraries (spaCy, NLTK, GSON)
     - Sorted by highest community scores
2. **Manual Analysis:** Nearly 500 posts were analyzed to select final 100 based on discussion of technical aspects:
   - Active Maintenance
   - Documentation quality
   - Ease of use
   - Feature availability
   - Performance characteristics
   - Security considerations
   - Stability metrics

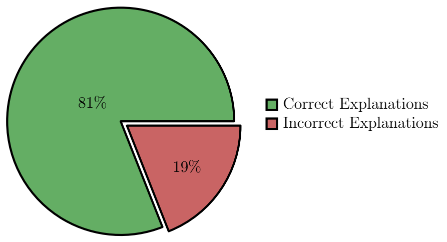


Figure 6: Distribution of Correct and Incorrect Explanations

We picked this task because our survey responses showed SE practitioners want to use ChatGPT during their selection and reuse of a software library [19].

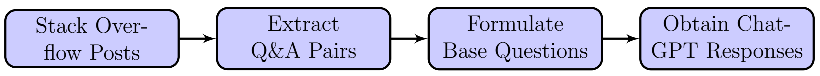


Figure 7: Flowchart of Benchmark Study Setup

Table 2: Base question template for different factors

Factor	Base Question
Active Maint.	How actively is the library maintained
Documentation	How is the documentation of the library
Ease of use	How easy is it to use the library
Feature	How well does this library support [x] feature
Performance	How is the performance of the library
Security	How is the security of the library
Stability	How stable or well tested is the library

4.1.2 Question Templates

Standardized templates were created for each factor:

4.1.3 Data Processing

Each interaction with ChatGPT followed a systematic process:

- **Context Integration:**
  - Stack Overflow post content (title, question, answer) provided as context
  - Word limit of 200 words imposed for consistency
  - Strict instructions to base responses only on provided context
- **Response Generation:**
  - Used ChatGPT model version gpt-3.5-turbo-0301
  - Generated 341 total explanations
  - Manual labeling based on Stack Overflow ground truth
  - 276 explanations (81%) labeled as correct
  - 65 explanations (19%) labeled as incorrect

4.2 Incorrectness Detection Performance (RQ4)

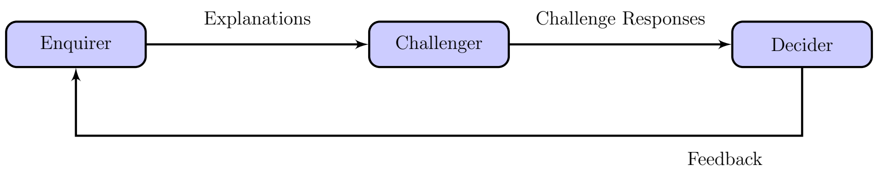


Figure 8: Interaction between CID Components

4.2.1 Model Selection and Training

Three supervised machine learning models were evaluated:

1. **Logistic Regression (LR):**
   - Binary classification approach
   - Linear decision boundary
   - Baseline performance benchmark

   In LLMs, log probabilities mainly indicate uncertainty over specific tokens, rather than capturing epistemic uncertainty related to the correctness of the given information [12, 20].

2. **Random Forest (RF):**
   - Ensemble learning method
   - Multiple decision trees
   - Good at handling non-linear relationships

3. **Support Vector Machine (SVM):**
   - Non-linear classification
   - Kernel trick for higher dimensional mapping
   - Best performing model

4.2.2 Performance Metrics

Table 3: Model Performance Comparison

Model	Precision	Recall	F1-Score	Accuracy
SVM	0.76	0.73	0.74	0.75
RF	0.72	0.70	0.71	0.72
LR	0.70	0.69	0.69	0.70

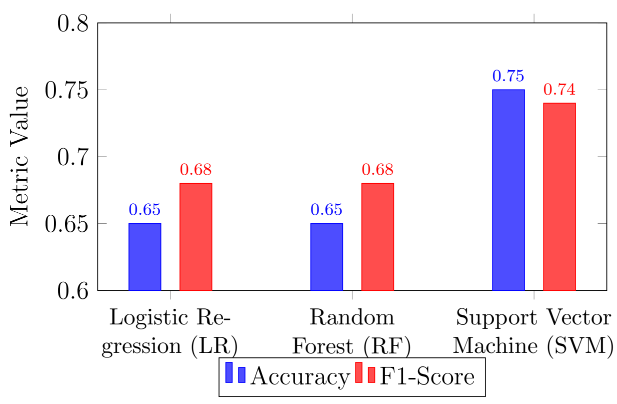


Figure 9: Comparison of ML Model Performances

4.2.3 Error Analysis

Detailed analysis of 86 misclassifications revealed systematic patterns:

1. **Decider-Related Errors (44%):**
   - **Similarity Detection Issues (24%):**
     - High similarity scores in incorrect responses
     - Unanimous mistake confirmation
     - Need for additional filtering layer
   - **Calculation Problems (21%):**
     - Challenges with large text responses
     - Sentence transformer limitations
     - Text comparison accuracy issues
2. **Challenger-Related Errors (32%):**
   - **Misdirected Challenges (18%):**
     - Questions unrelated to core reasoning
     - Example: Asking "Is the library open-source?" when evaluating stability
   - **Scope Issues (15%):**
     - Questions beyond context boundaries
     - Example: Requesting specific developer counts
3. **Enquirer-Related Errors (5%):**
   - Multiple Opinion Handling
   - Convoluted explanations
   - Mixed correct/incorrect statements
   - Difficulty in separation
4. **Mixed Source Errors (19%):**
   - **Continuous Wrong Reasoning (9%):**
     - Consistent but incorrect logic
     - Challenge ineffectiveness
   - **Generic Issues (9%):**
     - Unclear information
     - Inconclusive opinions

4.3 Impact of Challenge Prompts (RQ5)

4.3.1 Component Analysis

Systematic evaluation of different prompt combinations:

- **Full System Performance:**
  - Both prompt types active
  - Baseline accuracy: 0.75
  - F1-score: 0.74
- **Without Mutation Challenges:**
  - Significant performance drop
  - Accuracy decrease: 16%
  - New accuracy: 0.63
- **Without Basic Challenges:**
  - Moderate performance impact
  - Accuracy decrease: 8%
  - New accuracy: 0.69

4.3.2 Question Type Impact

Analysis of individual question types:

1. **"How" Questions:**
   - Minimal performance impact
   - Often led to out-of-scope responses
   - Example: Original statement about library stability leading to irrelevant developer count questions
2. **"Really" Questions:**
   - Significant impact on detection
   - Better at revealing inconsistencies
   - Example: Questioning comprehensiveness of documentation revealed gaps
3. **"Why" Questions:**
   - Strong performance in logic verification
   - Effective at exposing reasoning flaws
   - Example: Challenging performance claims against benchmark data

Table 4: Impact of Different Question Types

Configuration	Accuracy	F1-Score	Impact
All Questions	0.75	0.74	Baseline
Without ”How”	0.75	0.74	Minimal
Without ”Really”	0.71	0.70	Significant
Without ”Why”	0.72	0.71	Moderate

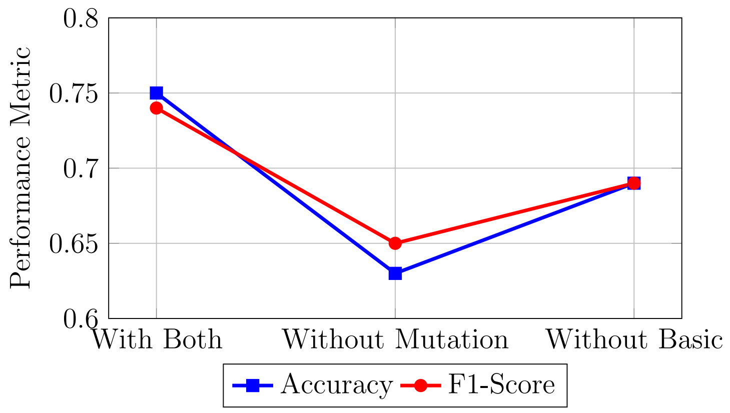


Figure 10: Impact of Mutation and Basic Challenges

4.3.3 Mutation Challenge Effectiveness

To illustrate the effectiveness of mutation challenges, consider this example:

Example of Mutation Challenge Impact

Original Question:
”How comprehensive is the documentation for training a TextCategorizer model?”
Basic Response:
”Documentation is helpful and straightforward with no limitations mentioned.”
Mutated Question:
”How comprehensive is the documentation, considering limitations of online learning?”
Mutated Response:
”Documentation is comprehensive but users may face limitations with new entities.”
Result: Inconsistency detected, leading to correct classification as inaccurate

4.3.4 Key Findings

1. **Detection Accuracy:**
   - SVM achieves 0.75 accuracy

- Reliable for practical application
- Room for improvement in specific areas

2. **Challenge Strategy:**
    - Mutation challenges most effective
    - Combined approach preferred
    - Need for targeted question generation

3. **Error Patterns:**
    - Systematic categorization possible
    - Most issues are addressable
    - Future improvement paths identified

4.3.5 Limitations and Considerations

1. **Dataset Limitations:**
    - Stack Overflow focus
    - Limited library scope
    - Potential bias in accepted answers

2. **Model Constraints:**
    - Text similarity challenges
    - Complex response handling
    - Context boundary issues

3. **Application Scope:**
    - Library selection focus
    - Generalization potential
    - Need for domain adaptation

Summary of Evaluation. The evaluation demonstrates CID's effectiveness in detecting incorrect responses through systematic challenge and analysis of ChatGPT's consistency patterns. The SVM-based model achieved an F1-score of 0.74-0.75, with mutation challenges proving particularly effective at revealing inconsistencies. Similar to other LLMs, ChatGPT also suffers from a lack of consistency [9]. Previous black-box approaches mainly work by generating multiple responses for the same question [14]. Our approach is based on iterative prompting to capture ChatGPT's inconsistency in a similar fashion to an actual Crime Investigation Department (CID) [4].

5 Related Work

We compare our work with (1) the current techniques developed to test the quality of LLMs and (2) the adoption of LLMs in SE tasks.

5.1 Reliability Issues of LLMs

Despite the efficacy of LLMs including ChatGPT, they are often susceptible to generating non-factual statements and even hallucinating facts. Researchers have explored various factors contributing to this issue, such as:

- The quality of training data
- Source-target divergence
- Inadequate modeling
- Randomness during inference

Several studies leveraged the logit output values as a measure of the model's 'uncertainty' and used them to detect hallucination. *Logit output values* refer to the raw probability scores for the generated tokens, which are then transformed using the softmax function and logarithmically scaled to calculate log probabilities. However, log probabilities are not a good estimate of hallucination or epistemic uncertainty as they merely indicate the likelihood of specific tokens or ways of expressing a claim.

Recently, researchers have explored various approaches:

- Azaria and Mitchell leveraged the internal state of the language model to determine truthfulness
- Mielke et al. trained a correctness predictor using model's internal representations
- Previous black-box approaches mainly work by generating multiple responses for the same question

Contrast with CID: Our CID tool differs in that:

- Applicable to black-box settings without requiring internal state
- Uses iterative prompting and challenges
- Focuses on revealing inconsistencies indicative of incorrectness

Similar to other LLMs, ChatGPT also suffers from consistency issues. The model frequently alters its decisions when presented with a paraphrased sentence, revealing self-contradictory behavior. Existing research considers correctness and consistency as separate objectives. In contrast, our work shows that assessing response deviations can detect incorrect responses.

Several mitigation strategies have been explored:

- Fine-tuning with human feedback and reinforcement learning
- Factuality-enhanced training
- External fact-checking tool employment
- Improved prompting
- External knowledge incorporation

Key Distinction: While existing research treats detection and mitigation of inaccuracies as separate problems, our CID tool warns users of potential reliability issues that might warrant verification.

5.2 LLMs for Software Engineering (SE)

Both specialized and general-purpose LMs/LLMs have emerged as powerful tools for various SE applications. Models like PLBART [1], CodeBERT [6] are designed with a focus on addressing the unique challenges and requirements of software development tasks.

5.2.1 Specialized Models

Notable specialized models include:

- PLBART: Unified architecture for programming and natural language
- CodeBERT: Focus on code-specific representations
- CodeT5: Enhanced for code-related tasks
- CoTexT: Handles both programming and natural language

These architectures rely on large-scale pre-training of both programming and natural language data, enabling various downstream SE tasks.

5.2.2 General-Purpose Models

More general-purpose LLMs include:

- GPT-based models
- Codex
- ChatGPT

These models have been applied to various SE tasks:

- Code generation
- Code repair
- Code summarization

Applications in SE: Studies have leveraged such models for:

- Code generation tasks
- Program repair
- Code summarization/explanation
- Interactive programming assistance

Our Contribution: We address a gap in SE literature by:

- Conducting a comprehensive survey of 135 developers
- Developing CID for assessing ChatGPT inaccuracies
- Demonstrating effectiveness in software library selection

Summary of Related Work. Our research uniquely combines developer perspectives with automated detection of ChatGPT inaccuracies. While existing work focuses either on LLM reliability or SE applications independently, we bridge this gap by providing both empirical insights and practical tools for SE practitioners using ChatGPT.

References

[1] Wasi Uddin Ahmad et al. "Unified Pre-training for Program Understanding and Generation". In: *NAACL*. 2021, pp. 2655–2668.

[2] Toufique Ahmed and Premkumar Devanbu. "Few-Shot Training LLMs for Project-Specific Code-Summarization". In: *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 2022, pp. 1–5.

[3] Yejin Bang et al. "A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity". In: *arXiv preprint* arXiv:2302.04023 (2023).

[4] Haneen Deeb et al. "Police officers' perceptions of consistency in suspects' accounts". In: *Applied Cognitive Psychology*. 2018, pp. 841–849.

[5] Philip Feldman, James R. Foulds, and Shimei Pan. "Trapping LLM Hallucinations Using Tagged Context Prompts". In: *arXiv preprint* arXiv:2306.06085 (2023).

[6] Zhangyin Feng et al. "CodeBERT: A Pre-Trained Model for Programming and Natural Languages". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. 2020, pp. 1536–1547.

[7] James Finnie-Ansley et al. "The robots are coming: Exploring the implications of OpenAI Codex on introductory programming". In: *Proceedings of the 24th Australasian Computing Education Conference*. 2022, pp. 10–19.

[8] Leo A. Goodman. “Snowball Sampling”. In: *Annals of Mathematical Statistics* 32.1 (1961), pp. 148–170.

[9] Joel Jang et al. “Consistency Analysis of ChatGPT”. In: *arXiv preprint* arXiv:2304.06892 (2023).

[10] Junaed Younus Khan and Gias Uddin. “Automatic Code Documentation Generation Using GPT-3”. In: *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering.* 2022, pp. 1–6.

[11] Junaed Younus Khan and Gias Uddin. “Combining Contexts from Multiple Sources for Documentation-Specific Code Example Generation”. In: *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER).* IEEE, 2023, pp. 683–687.

[12] Bill Yuchen Lin et al. “Teaching Large Language Models to Self-Debug”. In: *arXiv preprint* arXiv:2304.05128 (2022).

[13] Jiawei Liu et al. “Is your code generated by ChatGPT really correct? Rigorous evaluation of large language models for code generation”. In: *arXiv preprint* arXiv:2305.01210 (2023).

[14] Potsawee Manakul, Aike Liusie, and Mark Giles. “SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models”. In: *ICLR 2024.* 2023.

[15] OpenAI. *Online-ChatGPT - Optimizing Language Models for Dialogue.* `https://online-chatgpt.com/`. [Accessed 31-July-2023]. n.d.

[16] Hammond Pearce et al. “Examining Zero-Shot Vulnerability Repair with Large Language Models”. In: *2023 IEEE Symposium on Security and Privacy (SP).* IEEE, 2023, pp. 2339–2356.

[17] Julian Aron Prenner and Romain Robbes. “Automatic Program Repair with OpenAI’s Codex: Evaluating QuixBugs”. In: *arXiv preprint* arXiv:2111.03922 (2021).

[18] Qingchao Shen et al. “Natural Test Generation for Precise Testing of Question Answering Software”. In: *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering.* 2022, pp. 1–12.

[19] Gias Uddin and Foutse Khomh. “Understanding how and why developers seek and analyze API-related opinions”. In: *IEEE Transactions on Software Engineering* 47.3 (2019), pp. 478–498.

[20] Lav R. Varshney, Aditya Agarwal, and Zhangyang Wang. “Stitch in Language Models: Learning to Combine Multiple Responses”. In: *arXiv preprint* arXiv:2305.14937 (2023).