ChatGPT Incorrectness Detection in Software Reviews

Minaoar Hossain Tanzil
minaoar@gmail.com
University of Calgary
Calgary, Alberta, Canada

Junaed Younus Khan
junaedyounus.khan@ucalgary.ca
University of Calgary
Calgary, Alberta, Canada

Gias Uddin
guddin@yorku.ca
York University
Toronto, Ontario, Canada

ABSTRACT

We conducted a survey of 135 software engineering (SE) practitioners to understand how they use Generative AI-based chatbots like ChatGPT for SE tasks. We find that they want to use ChatGPT for SE tasks like software library selection but often worry about the truthfulness of ChatGPT responses. We developed a suite of techniques and a tool called CID (ChatGPT Incorrectness Detector) to automatically test and detect the incorrectness in ChatGPT responses. CID is based on the iterative prompting to ChatGPT by asking it contextually similar but textually divergent questions (using an approach that utilizes metamorphic relationships in texts). The underlying principle in CID is that for a given question, a response that is different from other responses (across multiple incarnations of the question) is likely an incorrect response. In a benchmark study of library selection, we show that CID can detect incorrect responses from ChatGPT with an F1-score of 0.74 - 0.75.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**; • **Software and its engineering** → *Software testing and debugging*.

KEYWORDS

Large language model, ChatGPT, Hallucination, Testing

1 INTRODUCTION

Thanks to the recent advent of pre-trained large language models (LLMs) like ChatGPT [51], many domains are now undergoing rapid automation transformation. The interactive interface of ChatGPT entices users of different professions to use it as a personal assistant and software engineering (SE) is no exception. Studies employed LLMs in various SE tasks like code generation [26, 37, 45], program repair [55, 58, 74], and code summarization/explanation [2, 36, 48]. However, LLMs can generate incorrect results [7, 23, 27, 77].

In this paper, we conduct a survey and develop a tool to assess the quality of ChatGPT while supporting SE practitioners. We focus on software library selection as a case study for which developers would like to use ChatGPT, because we know that developers rely on online reviews during library selection [63].

In the first phase, we surveyed 135 SE practitioners to learn their preferences and concerns about ChatGPT usage. We find that

- SE practitioners are enthusiastic to use ChatGPT for diverse SE applications, such as code generation, library exploration, etc.
- However, most of them do not fully rely on the responses and consider them as unreliable or somewhat reliable.
- Developers employ various manual but cumbersome strategies to verify ChatGPT responses like further searches in Google or Stack Overflow, seeking additional clarifications from ChatGPT through follow-up questions, etc. They wished for automated tool support to assess the inaccuracies in ChatGPT responses.

In the second phase, we develop a suite of techniques and a tool called CID (ChatGPT Incorrectness Detector) that can automatically determine inaccuracies in ChatGPT responses.

One might argue that the logistic probability of the tokens in LLMs can be indicative of the correctness of the generated results [66]. However, in LLMs, log probabilities mainly indicate uncertainty over specific tokens, rather than capturing epistemic uncertainty related to the correctness of the given information [43]. When a claim can be paraphrased in various ways, each paraphrase may have a low log probability regardless of the epistemic correctness. Lin et al. showed that such models can be calibrated to express their level of epistemic uncertainty (namely, verbalized probability) which, however, requires domain-specific fine-tuning [43]. It is also reported that though the base pre-trained model of GPT-4 (which happens to be the underlying model of ChatGPT) is highly calibrated, the calibration is affected by further post-training process i.e., Reinforcement Learning from Human Feedback (RLHF) [3]. Moreover, all of these approaches are gray/white-box testing, because they either require the original model or its underlying architectural details. Such approaches are not suitable for scenarios when people use third-party chatbots like ChatGPT.

In contrast, for our CID tool, we derive a black-box testing approach. Our approach is based on iterative but automated guided prompting in ChatGPT, where the prompted questions can be mutated without changing the base query, i.e., what we are asking across multiple questions for a given prompt remains the same, but we ask the questions in different ways. We employ metamorphic relationships to mutate the questions. We then calculate the similarity of the generated responses and use the similarity assessment to automatically label whether a response is incorrect. The underlying principle in CID is that for an original ChatGPT response that is incorrect, the subsequent question responses are likely to be

Table 1: Demography of Survey Participants

Current Profession	Years of Experience					Total
	0-5	6-10	11-15	16-20	21-24	
Software Engineer (Dev/QA/Data/Ops)	64	34	11	4	-	**113**
Manager (Dev/PM/Ops)	-	3	10	2	-	**15**
Executive (Company Leadership)	-	-	1	1	2	**4**
Non-Tech (Sales/Marketing/CS/HR)	-	2	1	-	-	**3**
Total	64	39	23	7	2	**135**

(more) inconsistent (i.e., dissimilar). In a benchmark study of software library selection, we show that CID can detect such incorrect responses with an F1-score of 0.74 - 0.75.

2 SURVEY TO ASSESS SOFTWARE DEVELOPER PERSPECTIVES ON CHATGPT USAGE

We surveyed 135 SE practitioners to understand how they use generative artificial intelligence (AI) based chatbots like ChatGPT for SE tasks. We answer three research questions (RQs):

RQ1. Why do software developers use ChatGPT?
RQ2. How much do developers rely on ChatGPT responses?
RQ3. How do developers verify ChatGPT responses?

2.1 Survey Setup

The surveys were conducted after following review and approval from the Research Ethics Board of the associated university.

2.1.1 Survey Questions. We asked each participant 10 questions listed in Table 2. We designed the survey in a semi-close-ended fashion where most questions had the opportunity to write down additional feedback. The survey questionnaire was designed based on a pilot survey. In the pilot survey, we provided the questionnaire to 24 industry professionals from big software companies across the world. The recruitment of these professionals was conducted by mostly convenient sampling from the professional network of the authors. Before distributing the survey, we provided them with a short briefing of our intention. Finally, 20 professionals participated in the survey. Based on their survey responses and post-survey feedback, we re-adjusted our final survey.

2.1.2 Survey Participants. The participants' roles include software engineers, developers, managers, etc. Table 1 shows the distribution of the participants over their profession and years of experience. We used the snowball sampling [28] approach to select the participants. First, we picked 12 senior individuals (from our personal contacts) having at least 15 years of experience in the industry. We recorded their feedback and also requested them to invite appropriate candidates. 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. While such platforms may offer access to a large number of potential participants, the lack of control over the respondents' qualifications and expertise could compromise the reliability of our study. The 20 participants from the pilot survey were omitted from our list of 135 survey participants.

2.2 Reasons for using ChatGPT (RQ1)

Q1 We asked the participants whether they have used ChatGPT or not. The vast majority (133 respondents) reported having already used ChatGPT, while only 2 participants responded in the negative.
- *Yes:* 98.52%
- *No:* 1.48%

Q2 We were interested in understanding the common purposes ChatGPT serve to the software professionals regardless of their specific roles in software engineering. After all, they are human too. As this was a multiple-choice checklist question, respondents were allowed to select multiple options.
- *Just for Fun:* 42.22%
  57 participants used ChatGPT just for fun and entertainment purposes, engaging with its capabilities in a playful manner.
- *As a Search Engine:* 79.26%
  The majority (107) of respondents utilized ChatGPT as a search engine to collect information, highlighting its potential as an AI-powered information retrieval tool.
- *Content Creation:* 46.67%
  63 participants used ChatGPT to generate new content, such as presentations, documents, images, and more.
- *Decision Making:* 48.15%
  65 respondents used ChatGPT for decision-making purposes.
- *Exploration and Data Collection:* 27.41%
  37 participants employed ChatGPT for initial exploratory data collection, utilizing its capabilities to gain insights into a topic.
- *Research for Product/Business Development:* 28.15%
  38 participants used ChatGPT for market research, specifically to explore new products or business development ideas.
- *Learning and Knowledge Acquisition:* 61.48%
  A significant proportion (83) of participants used ChatGPT as an educational resource to learn about new technologies.
- *Others:* 1.32%
  A few respondents mentioned some additional use cases for ChatGPT such as asking questions, generating ideas, and finding solutions to different problems. However, these use cases can also be fitted into the above-predefined options.

Q3 We asked the developers how they (are willing to) use ChatGPT in their work as software professionals. The question was a multiple-choice checklist.
- *Code Generation and Optimization:* 60%
  81 survey participants use ChatGPT for generating code for their applications, demonstrating its value in code completion, generation, optimization, and refactoring tasks.
- *Code Analysis and Review:* 52.59%
  71 respondents leverage ChatGPT for analyzing code-related tasks, such as code comparison, debugging, summarization, and code review. This highlights ChatGPT's significance in assisting developers with code understanding and error identification.
- *Problem-Solving Support:* 86.67%
  A significant proportion of our survey participants (117) turn to ChatGPT to find solutions to coding problems they encounter.

Table 2: 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

This indicates that ChatGPT is a valuable resource for developers when facing challenges during the development process.

- **Exploring Alternative Approaches:** ████████ 73.33%
  99 respondents use ChatGPT to explore alternative (better) ways to develop code or implement specific features. This indicates that ChatGPT plays a crucial role in offering creative insights and ideas to developers during the software development lifecycle.
- **Library Selection:** █████ 46.67%
  ChatGPT is considered by 63 participants to discover appropriate libraries and frameworks, for their development projects.
- **Others:** ▍2%
  Some developers (2%) also mentioned additional use cases for ChatGPT, such as comparing code, using it for non-professional coding needs on websites, performing unit tests, etc.

Q4 Next we asked the participants about their reaction/experience of using ChatGPT for different development tasks. The participants expressed both positive and negative feelings as follows.

- **Exciting:** ███████ 68.15%, **Tech Marvel:** █████ 42.22%
  92 respondents reported being excited by ChatGPT's capabilities and potential to assist them in various tasks while 57 of them remark it as a technological marvel.
- **Unreliable:** ████ 34%, **Over-hyped:** ██ 19.26%, **Suspicious:** ▍9.63%
  46 participants found their experience with ChatGPT to be unreliable. They encountered instances where the tool provided inaccurate or inconsistent responses. 26 respondents feel considered ChatGPT overhyped and the tool's capabilities as exaggerated.

> **Summary of RQ1.** The majority of developers (98.52%) used ChatGPT for various purposes. From code generation (60%) and analysis (52.59%) to problem-solving (86.67%), exploring coding alternatives (73.33%), and libraries (46.67%), ChatGPT served as a versatile tool to them. Other common usages include information retrieval (79.26%), content creation (46.67%), etc.

2.3 Concerns about ChatGPT Responses (RQ2)

Q5 We asked participants to indicate the extent of their reliance on ChatGPT contents, revealing a spectrum of responses ranging from complete trust to cautious validation with other sources. Most participants are found to be doubtful of ChatGPT responses.

- **Not Reliable At All:** ▊5.19%
  7 participants indicated that they do not consider the ChatGPT

responses as reliable at all. They always cross-check the information with other sources for validation.

- **Somewhat Reliable: Need Validation** █████ 54.81%, **Need Augmentation** ███ 35.56%
  The majority (74 participants) stated that while ChatGPT provides them with initial ideas, they rely on other sources for validation. Moreover, 48 participants expressed that they subsequently search for additional knowledge from other sources. Only one participant expressed reliance on the correctness of responses.
- **Others:** ▍3.70%
  A few (5) participants put emphasis on proper prompting to assess inaccuracy, i.e., asking questions to ChatGPT more precisely.

Q6 We asked participants to discuss the pros and cons of ChatGPT usage. Specifically, we asked for their opinion on library selection and comparison. Out of the 135 participants, 72 provided responses to this open-ended question. The results are summarized below.

Pros of Using ChatGPT for Library Selection

- **Efficient Access to Information:** 22 participants mentioned that ChatGPT provides quick and efficient access to information which allows them to receive library suggestions or comparisons.
- **Initial Idea Generation:** 10 respondents found ChatGPT helpful for generating initial ideas or suggestions regarding software libraries, serving as a starting point for further research.
- **Personalized Recommendations:** 3 participants appreciated that ChatGPT could offer personalized recommendations based on specific requirements or preferences.
- **Time-Saving:** 10 respondents highlighted that ChatGPT's ability to provide library suggestions or comparisons quickly saved them time during the decision-making process.

Cons of Using ChatGPT for Library Selection

- **Lack of Up-to-dateness:** 12 participants expressed concerns about ChatGPT's limitations compared to human experts. They mentioned that it might lack up-to-date knowledge, real-world experiences, or insights about specific libraries.
- **Contextual Understanding Challenges:** 11 respondents noted that ChatGPT might struggle to understand nuanced or complex questions related to software libraries, leading to potentially inaccurate or irrelevant recommendations.
- **Reliability Concerns:** 26 participants pointed out that ChatGPT's responses were not always accurate or reliable, suggesting the need to cross-check with other sources for validation.

- *Dependence on Prompt:* 4 respondents mentioned that the quality of ChatGPT's responses relied on the input (prompt) provided by users, leading to biased/misguided recommendations.
- *Not Sufficient for Decision-Making:* 7 participants emphasized that ChatGPT might be inaccurate to be solely relied upon for complex decision-making tasks.
- *Bias of Training Data:* A few participants indicated the possible bias that might exist in ChatGPT's response.

> **Summary of RQ2.** There are concerns about ChatGPT reliability, with some finding it unreliable (34%) and over-hyped (19.26%). Only a small portion (0.74%) of our survey participants fully rely on the responses of ChatGPT (i.e., without double-checking).

2.4 Verification of ChatGPT Responses (RQ3)

Q7 Participants were presented with our conversation with ChatGPT where we asked ChatGPT about a good natural language processing (NLP) library in Python. ChatGPT suggested spaCy, NLTK, and Gensim as notable options. Our conversation with ChatGPT is as follows.

> **Interaction with ChatGPT for Survey Question #7**
>
> **Prompt:** What is a good library in Python for NLP?
>
> **Response:** spaCy is considered a top library for natural language processing (NLP) in Python. It provides a wide range of NLP functionalities, efficient processing, support for multiple languages, and easy integration with deep learning frameworks. Other notable options include NLTK and Gensim.

Participants were then asked whether they would rely on this response or not, and how they would verify it further. While a tiny portion (2.22%) of participants were ready to accept ChatGPT response without further verification, most of them preferred to seek additional information and verify the response as follows.

- *Further Search in Common Medium:* ▉▉▉▉▉ 73.33%
  The majority of participants (99 respondents) preferred to supplement ChatGPT's response by conducting further research on their own (e.g., by consulting Google or Stack Overflow).
- *Further Inquiry to ChatGPT:*
  About spaCy ▉▉▉▉▉ 68.15%, *NLTK&Gensim* ▉▉▉ 45.19%
  92 participants stated that they would ask ChatGPT more questions to further assess whether a suggested library (e.g., spaCy) fits their specific needs.
- *Others:* ▉ 4.44%
  A few respondents mentioned some additional ways to verify the response such as seeing the package/library or GitHub repository, reading the documentation, etc.

Q8 Participants were presented with a specific situation where ChaGPT is asked for more details about spaCy's API and documentation. They were then asked about their level of reliance on this detail. The interaction with ChatGPT is as follows.

> **Interaction with ChatGPT for Survey Question #8**
>
> **Prompt:** How easy it is to use spaCy?
>
> **Response:** spaCy is known for its user-friendly design, making it easy to use for natural language processing (NLP). With a straightforward API, extensive documentation, and many pre-trained models, spaCy simplifies the process of developing applications, making it accessible to beginners and advanced users.

After seeing this additional confirmatory prompt, developers have mixed expressions about the reliability of the previous response (i.e., spaCy to be a top NLP library) in survey question *Q7*. Some of them have shifted towards accepting the response while others still wanted further verification.

- *Rely on Confirmatory Details:* ▉▉▉▉ 55.56%
  75 respondents demonstrated high confidence in the new details and indicated that they would now rely on the information provided. They expressed their intention to jump into the official documentation of spaCy to learn more.
- *Still Search in Common Medium:* ▉▉▉▉ 51.11%
  69 participants still wanted to look for additional opinions and insights from sources like Google and Stack Overflow.
- *Further Inquiry to ChatGPT: About spaCy* ▉▉▉▉ 49.63%
  67 respondents sought further clarification by asking the ChatGPT for sample APIs or methods related to spaCy. They wanted more specifications for better understanding and verification.

Q9 To ensure ChatGPT's reliability in SE real-world situations, we need to assess its ability to- i) interpret specific scenarios and contexts accurately; ii) generate appropriate information based on that. Hence, we provided ChatGPT with a specific question-answer set (about spaCy) from Stack Overflow and asked it to interpret whether the library is easy to use solely based on the provided information. Our interaction with ChatGPT is as follows.

> **Interaction with ChatGPT for Survey Question #9**
>
> **Prompt:** How easy it is to use the library spaCy strictly based on the following conversation (*Question, Answer*).
> *Question: I'm processing 40,000 abstracts with spacy nlp.pipe and its taking 8 mins. Is there a way to speed this up further? I've also disabled ner.*
> *Answer: Try tuning batch_size and n_process params: Note as well, by joining on "" "" you may have some surprises, as spaCy's splitting rules are a bit more complex than that.*
>
> **Response:** spaCy is easy to use, but optimizing performance may involve tuning batch size and process count parameters.

We showed this interaction with ChatGPT to the participants and asked them whether they agree with ChatGPT's interpretation about the usability of Spacy. Most of them remain unconvinced.

- *Unsure of ChatGPT's Interpretation:* ▉▉▉▉ 54.07%
  73 respondents were unsure about the accuracy of ChatGPT's interpretation. They neither fully agreed nor disagreed.
- *Disagree with ChatGPT's Interpretation:* ▉▉▉ 34.81%
  47 respondents disagreed with ChatGPT's response, suggesting

that the conversation actually discussed the complexity of spaCy and not its ease of use.
- ***Agree with ChatGPT's Interpretation:*** ◼ **11.11%**
  15 respondents agreed with ChatGPT's interpretation, stating that the conversation somehow hints that spaCy is easy to use.

Q10 Finally, participants were asked to suggest ways to improve the reliability of ChatGPT responses. The participants' responses can be grouped into the following key ways.
- ***Showing References:*** ▬▬▬▬ **80.74%**
  109 participants suggested that showing (online) references supporting the answer would improve the reliability of ChatGPT responses. One participant proposed showing diagrams and resource links (e.g., YouTube tutorials) as supplementary materials.
- ***Asking Challenging Questions:*** ▬▬ **44.44%**
  60 participants mentioned that repeatedly asking challenging questions to ChatGPT could help us with appropriate direction.
- ***Better Prompt Engineering:*** ▮ **3.70%**
  5 participants highlighted the importance of improving prompt engineering to make ChatGPT responses more reliable.
- ***User Feedback:*** ▮ **1.48%**
  2 participants emphasized the significance of user feedback and learning to improve the ChatGPT's response quality.
- ***Other Chatbots' Feedback:*** ▮ **0.74%**
  1 participant came up with the idea of verifying ChatGPT responses from other similar chatbots (such as Google BARD).

> **Summary of RQ3.** 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

Our survey participants wished to perceive an LLM as a remarkably close approximation of human capabilities. A desirable property of a good LLM is consistency: the ability to make consistent decisions in semantically equivalent contexts. However, it is true that humans are also prone to inconsistency, especially when they are lying or bluffing. There is a general tendency among practitioners (e.g., police officers) to perceive consistent statements as truthful, and inconsistent statements as deceptive [16]. In fact, studies in criminal justice find that increasing the cognitive load on the suspect with unanticipated (e.g., convoluted) questions is effective in outsmarting/detecting liars [38, 67, 68]. A recent study by Dhuliawala et al. demonstrates the effectiveness of similar approaches on LLMs [18]. They proposed the Chain-of-Verification (CoVe) method, which challenges the model to verify its initial responses with independently generated questions to enhance the reliability of the model's output. Therefore, our CID (ChatGPT Incorrectness Detector) tool uses iterative prompting to capture ChatGPT's inconsistency in a similar fashion to an actual Crime Investigation Department (CID).

In a real case, an interrogator might: **1) Enquire** the suspect about the particular case, **2) Challenge** the suspect to reveal any inconsistency, **3) Decide** on the responses of the suspect. Likewise, CID comprises three key components: ENQUIRER, CHALLENGER, and DECIDER. The overview of CID is shown in Figure 1. The ENQUIRER seeks the initial reasoning behind ChatGPT's responses, while the CHALLENGER poses a series of questions to induce potential inconsistencies. The DECIDER employs machine learning (ML) techniques and similarity measures to determine the correctness of ChatGPT's responses based on the detected inconsistency. We discuss each component in detail below.
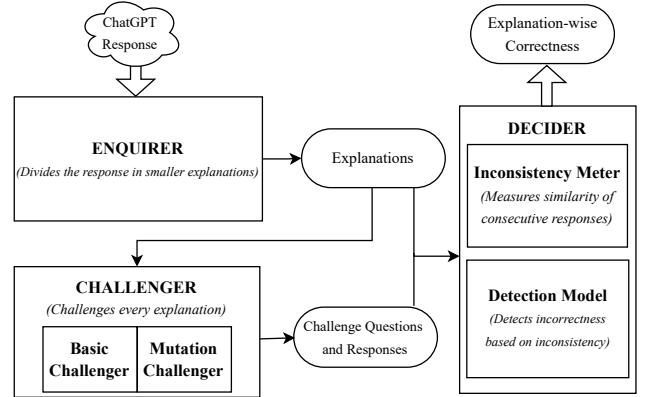

Figure 1: Overview of CID Tool.

3.1 ENQUIRER

Suppose, we ask a question (base-question, Q_B) and want to evaluate the correctness of its response (base-response, R_B). The ENQUIRER targets to obtain ChatGPT's initial reasoning behind the base-response that can be useful to reveal any inconsistency in the next steps of interrogation. When ChatGPT responds to any question, it typically provides multiple pieces of information, which may lead to partial correctness, meaning that some parts of the response are accurate while others are not. To address this issue, the ENQUIRER asks ChatGPT to provide separate explanations (E_i) for each piece of information that exists in the base-response. The ENQUIRER performs this by using the following prompt.

> **Enquiring ChatGPT**
>
> ✂ **ENQUIRER:** 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. The basic intuition is that for a piece of information given by an LLM that is possibly incorrect, the LLM will be (more) inconsistent in such consecutive prompts. The Challenger component works in four steps. In the first step, it generates a dynamic 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
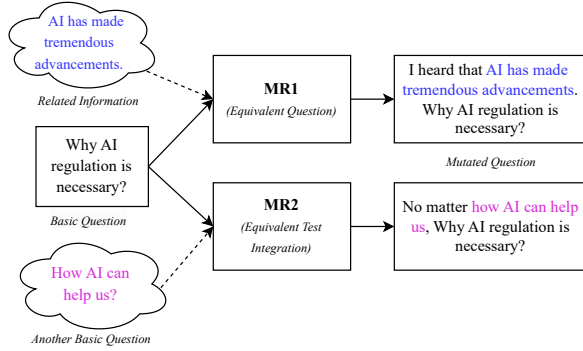
Figure 2: Metamorphic Relations (MRs) used in the Mutation Challenger to mutate questions.

step, it asks the mutated challenge again to ChatGPT. We describe the basic and mutation challenges below.

3.2.1 Basic Challenger. We first ask three basic challenge questions to ChatGPT: *Why?*, *How?*, *Really?* for each explanation (E_i) of its base-response (R_B). To automate the questioning process, the basic challenger leverages a separate LLM (completely unrelated to the original ChatGPT instance that is being evaluated by the CID tool). To replicate a separate LLM, we used ChatGPT with a new separate session. The motive for using a separate session of ChatGPT is to discard the memory of the previous conversation performed (with ChatGPT) before generating the challenge questions (i.e., during the Enquiry phase). This is done to ensure the elimination of potential bias that could be induced from the ENQUIRER memory. We generate the basic challenge questions as follows.

Generating Basic Challenge Questions

🛠 **BASIC-CHALLENGER:** Generate a question that starts with [Why/How/Really] to challenge the following [EXPLANATION].

3.2.2 Mutation Challenger. Returning to our analogy between human deception and LLMs' errors: (human) liars can often be as consistent as truth-tellers since they prepare for the interview by anticipating questions and rehearsing responses to them. However, the interviewer may make the interview more cognitively demanding for liars which disrupts liars' preparations and makes it difficult for them to provide spontaneous responses. Similarly, our confirmatory questions, if simple, might not pose enough challenge for the LLM which means the LLM could answer them consistently even for a wrong base-response/explanation. Hence, we use Mutation Challenger to further modify or complicate the challenge questions so that they can be more difficult for ChatGPT. Therefore, the Mutation Challenger mutates the basic challenge questions to create mutation challenge questions ($Q_{CB} \rightarrow Q_{CM}$).

This approach benefits from existing techniques for evaluating question-answer (QA) language models [12, 30, 59, 78]. Multiple studies proposed reliability testing techniques for the QA software [11, 20, 34, 47, 59, 60]. Among them, Shen et al. [60] outperformed previous state-of-the-art approaches by discovering 23% more bugs or inconsistencies in the target answer-generating models and the Challenger component adopted their question mutation technique

for generating challenging questions. For this, our mutation challenger employs the sentence-level metamorphic testing technique, QAQA, proposed by [60]. It inserts a redundant sentence as a clause to the original (basic challenge) question to generate the mutated question which puts a larger cognitive load on ChatGPT and challenges it further to reveal inconsistency with the responses (R_{CM}). The redundant sentence provides an additional fact that is combined with the original question without altering its semantics or the fact itself. 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 predefined 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. We show examples of both MRs in Figure 2. Different subordinate clauses (such as *'I heard that'*, *'No matter what'*, *'I do not care'*, etc.) are used to combine the redundant information with the original question.

3.3 DECIDER

DECIDER makes the final call about the (in)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 (e.g., Cosine Similarity).
- Detection Model: DECIDER utilizes an ML model (e.g., Support Vector Machine) 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 labelled dataset. By doing so, 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), we need a labeled dataset. We generate the dataset by interacting with ChatGPT and posing various questions to it. Each question should have a known correct answer that serves as the ground truth. For each question in the dataset, we interact with ChatGPT and record its base response to the given question. This base response will serve as the initial answer to be evaluated. Then we use our ENQUIRER to split each base response into multiple explanations (E_i). Finally, these explanations are manually labeled as correct/incorrect by human annotators. In Section 4.1, we discuss how we created such a benchmark.

3.3.2 Inconsistency Meter and Detection Model. Once we have a labeled dataset with explanations (E_i) labeled as correct or incorrect, we proceed with the feature extraction process. We compute standard similarity scores among ChatGPT responses generated in the ENQUIRY and CHALLENGE phases; use them as features for our tool. We then train an ML model that learns the relation between ChatGPT's incorrectness (i.e., labels) and inconsistency (features). We consider 24 features from four categories.

- Explanation-Response (E_i-R_C) Similarity: This category includes similarity scores calculated between the explanation and each of the challenge-responses. Hence, we get six similarity scores here

Table 3: Base question template for the SO posts

Factor	Base Question (Q_B)
Active Maint.	How actively the library is maintained
Documentation	How is the documentation of the library
Ease of use	How easy it is 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

(three for basic and three for mutation challenges). This allows our system to detect potential inconsistencies in the responses with respect to the base explanation.

- Response-Response (R_C-R_C) Similarity: The similarity among every pair of challenge responses is considered in this category. As the challenge responses point to the same piece of explanation, the system aims to identify potential discrepancies or inconsistencies between them. Higher similarity scores between challenge responses could indicate a consistent and well-supported base response/explanation, while lower similarity scores may suggest conflicting information or possible incorrectness. For pair-wise similarity among 3 basic challenge responses, we get $\binom{3}{2}$ = 3 features. Similarly, we get another 3 features for the 3 mutation challenge responses (i.e., total six R_C-R_C features).

- Question-Response (Q_C-R_C) Similarity: This category involves pair-wise similarity calculations between each challenge-question and corresponding challenge-response. The purpose is to see if the responses are relevant to the challenge questions. We again get six similar features from this category.

- Question-Question (Q_C-Q_C) Similarity: While we expect $R_C - R_C$ pairs to be consistent with each other, and consider their similarity scores as features, their inconsistency could be associated to the inconsistency/dissimilarity between the corresponding challenge questions (Q_C). This category aims to inform the model of pair-wise (di)similarities of the challenge questions resulting in six $Q_C - Q_C$ similarity features.

4 EVALUATION OF CID

Given that CID can be applicable for any computing tasks that require the usage of ChatGPT textual responses, within the short space of this paper we focused on evaluating CID for a given SE task: software library selection. We picked this task because our survey responses showed SE practitioners want to use ChatGPT during their selection and reuse of a software library. Moreover, various research papers in SE literature reported that software library selection can be a non-trivial task for developers [14, 15, 21, 31, 39, 42, 46, 63–65, 70, 71, 73, 76]. We thus evaluated CID using a benchmark study where the benchmark has information about correct and incorrect ChatGPT responses regarding software library usage. Using this benchmark dataset, we answer two RQs:

RQ4. How accurate is CID in detecting incorrect responses?

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

4.1 Benchmark Study Setup

4.1.1 Benchmark Dataset Collection. We first selected 100 Stack Overflow (SO) posts regarding which we will ask the questions. The

posts were related to a few popular text processing libraries (spaCy, NLTK, and GSON). From the posts, we collected the titles, questions, and accepted answers. Such accepted answers could provide ground truth to assess ChatGPT responses.

We followed a structured approach to collect the 100 posts using the search interface of SO [53]. In the search page, we provided two criteria: the post must have an accepted answer and the post must have tags of one of the popular text-processing libraries (GSON, NLTK, or spaCy). We sorted the search results so that the posts with the highest scores come first (the SO community sets the score). Then we manually analyzed each post based on two more criteria: the post should discuss any of seven technical aspects (such as Active Maintenance, Documentation, Ease of use, Feature, Performance, Security, and Stability) that developers use for evaluating libraries [39]. We manually analyzed nearly 500 posts and stopped further analysis after collecting 100 posts in total since the analysis was an extensively labor-intensive process. The reasoning of the selection criteria and the aspect-specific post count details is provided in online the appendix [61].

4.1.2 Base Question Generation. We used Chat API interface of ChatGPT model version gpt-3.5-turbo-0301. We created a question template for each of the factors. Table 3 shows the list of base questions we used for different SO posts. All the base questions ask for an opinion from ChatGPT regarding the library factor or feature. Instead of giving generic responses to the base questions, we wanted ChatGPT to provide the response only based on the related SO data. Hence, when we created the basic question for ChatGPT, we also added the SO post content (SO title, SO question, SO answer) as a context in the question. The prompt has the following format: *"Respond in less than 200 words "+Base Question (Q_B) + " strictly based on the following conversation (question, answer): + Context (C)"*. For example, a query for ease of use factor would be:

Enquiring ChatGPT

⚒ **Base Question:** Respond in less than 200 words How easy it is to use the library strictly based on the following conversation (question, answer):
Question: <SO Question> Answer: <SO Answer>

4.1.3 Explanation Generation from ChatGPT. After the base response is generated from ChatGPT, we use the ENQUIRER component to generate further explanation of the base response. The base response received from ChatGPT is an aggregated opinion and some parts of the opinion can be correct and some parts can be incorrect.

To identify the exact correctness, we have to dissect the opinion into multiple reasons. The first part of the following example shows the format of an exploratory question asked to ChatGPT. It contains a prompt to explain the previous base response and also provides the previous base question and answer as a reference. The second part of the example shows the response statements generated by ChatGPT. For simplicity, we omitted the JSON formatting from the response and only provided the statements. We also marked the correctness of those statements with green/red icons (however, the correctness is detected by the last component Detector).
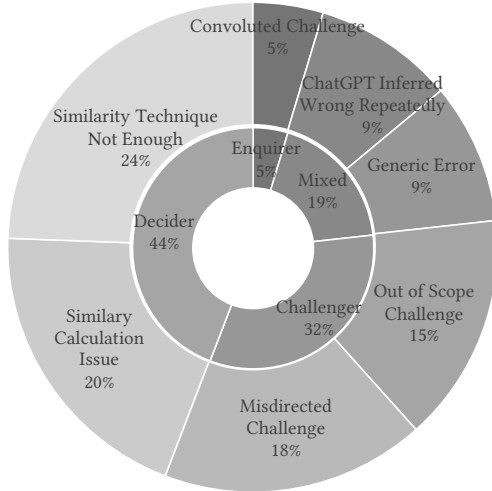
Figure 3: Sources and categories of misclassification.

Table 4: ML model performance to detect ChatGPT incorrectness. P = Precision, R = Recall, A = Accuracy. F1 = F1-score

Model	P	R	A	F1
Logistic Regression (LR)	0.74	0.65	0.65	0.68
Random Forest (RF)	0.73	0.65	0.65	0.68
Support Vector Machine (SVM)	0.74	0.75	0.75	**0.74**

Enquiring ChatGPT

🛠 **Explanatory Question:** Provide explanation for the answer. Each reason separately with title in 4 words and the explanation in 50 words. Respond strictly in JSON ["title":x, "explanation":y] strictly based on the previous conversation (question, answer). Question: <SO Question> Answer: <SO Answer> <Base Question> <Base Reponse>

💬 **Response:**
✓ A bug affects Spacy 2.3.0 and its tagger training functionality.
✓ spaCy dev team maintains and updates the library regularly.
✗ spaCy dev team is responsive and attentive to user feedback.

4.1.4 Explanation Labeling. After asking this enquiring question, ChatGPT responded with 3/4 explanations for each of the questions. Thus, for 100 posts, we found 341 explanations (in total) from ChatGPT. According to our labeling based on the original SO post, 276 explanations (81%) of them were correct.

4.2 Incorrectness Detection Performance (RQ4)

4.2.1 Approach. First, we trained and tested three supervised ML models (logistic regression (LR), random forest (RF), and support vector machine (SVM)) on our benchmark using a 10-fold cross-validation. We report the performance of the ML models on their incorrectness detection using four metrics: precision (P), recall (R), F1-score (F1), and accuracy (A). Second, we manually assess the misclassifications of our best-performing model.

4.2.2 Results. The evaluation results are shown in Table 4. SVM outperformed LR and RF models with an F1-score of 0.74, and accuracy 0.75 which means the decider could identify the correctness of 259 out of 341 ChatGPT responses. We further reviewed all 86 misclassifications from SVM model and identified 7 misclassification categories originating from 4 different sources (Figure 3).

Error Source - Decider (44% of Errors). 24% errors occur because the current **similarity technique is insufficient** to detect incorrect responses. The similarity score can be high even for an incorrect response when all challenge responses admit the mistake unanimously. An additional layer can be added to the decider to identify and filter out such unanimous confession of the challenge responses. Another category is the **similarity calculation issue** (21%) when the sentence-based transformers struggle to accurately detect high similarity among larger text responses from ChatGPT.

Error Source - Challenger (32%). 18% errors are due to **misdirected challenges** where the challenger generates a question that is unrelated to the actual logical reasoning of the explanation. For example, if an explanation is inferring that a library is stable because it is open-source, instead of challenging the inference, the challenger asks irrelevant questions like *"Is the library open-source?"*. Another similar error, accounting for 15% of cases, happens when the challenger asks **out of scope** questions that lead to incorrect predictions, like asking about the number of developers contributing to an open-source project in the above case.

Error Source - Enquirer (5%). Though the Enquirer is expected to split a base-response into an explanation as only one single opinion, sometimes a single explanation still contains multiple opinions (e.g., *"a library performance is great but complex to configure"*). 5% of misclassifications originate from such **convoluted challenges**.

Mixed Error Sources (19%). On 9% of error cases, CID cannot detect incorrect responses based on inconsistency as ChatGPT **continuously infers wrong logical reasoning** in all challenge questions. Other **generic reasons** (9%) include unclear information and lack of conclusive opinions in the explanation responses.

Summary of RQ4. The best-performing ML model to detect incorrect ChatGPT responses in our benchmark was SVM with an F1-score of 0.75. The major source of misclassification occurred due to the inability of the current similarity measures to handle larger and more convoluted texts for comparison.

4.3 Impact of Challenge Prompts (RQ5)

4.3.1 Approach. To understand the separate impact of the basic (or mutation) challenger, we evaluated the performance of CID by only keeping the corresponding 12 similarity attributes related to basic (or mutation) challenges. Moreover, to examine the impact of each of the three basic challenge types (*why, how*), we dropped each of them at once and noted CID's performance variance. We used the best performing model (SVM) for this evaluation.

4.3.2 Results. Table 5 shows the accuracy and F1-score of the prediction models where different question types are dropped. When all question types are present in the challenger, the average accuracy for all the prediction models is 0.75. The *How* question seems to have almost no or opposite impact on the prediction since the accuracy remains the same (0.75) without the *How* question, F1-score

Table 5: Impact of individual challenge prompts

Scenarios	Accuracy	F1-score
With all questions	0.75	0.74
Without *How* questions	0.75	0.75
Without *Really* questions	0.69	0.70
Without *Why* questions	0.70	0.71

Table 6: Impact of mutation & basic challenges

Scenarios	Accuracy	F1-score
With Basic and Mutation Challenges	0.75	0.74
Without Mutation Challenges	0.63	0.65
Without Basic Challenges	0.69	0.69

drops slightly (0.74) without them. *Really* and *Why* questions have higher impact on the prediction than the *How* questions.

Table 6 shows the prediction model performance in the absence of basic and mutation challenges. When all the challenges are present, the average prediction accuracy is 0.75 whereas if the mutation challenges are excluded, the accuracy significantly drops down to 0.63 (16% drop from 0.75). However, when only basic challenges are excluded, accuracy does not drop such drastically, it drops to 0.69 (8% drop from 0.75). Hence, it is obvious that the mutation challenges are more effective than basic challenges in improving the accuracy of the prediction model.

4.3.3 Limitation of How Challenges. Though we expected better impact from the *how* question types, we find that the prediction model performs better without *how* questions. Moreover, in section 4.2.2, we found that the out-of-scope challenges and the misdirected challenges contribute to 15% and 18% of errors respectively. On manual analysis of the challenge questions in our evaluation study, we found that *how* type challenge questions are contributing to such challenger-based errors. For example, ChatGPT provided a wrong response saying that *"Presence of an open-source community ensures stability"* though there is no such direct causation relation between open-source community and project's stability. To challenge this explanation, the challenger component generated two relevant challenging questions with *why* and *really*: *"Why active dev community contribute to stability" "Really, is active dev community solely responsible for stability".* In response to these questions, ChatGPT agreed that *"No direct answer available"* and *"This is hard to determine"* respectively. Both these responses would show that the original statement from ChatGPT was wrong. However, the challenger component generated an out-of-scope challenge using *how* question type: *"How many developers are contributing to the open source project."* and ChatGPT responded with *"No information provided".* Because of this out-of-context question and response, the model got confused and failed to detect the error ultimately.

4.3.4 Impact of Question Mutation. Let's provide an example of how mutated questions are improving prediction accuracy. After being asked about the effectiveness of SpaCy in text categorization task (base-question), ChatGPT responded in the affirmative with the following explanation: *"The user is able to train a TextCategorizer model with labeled data pairs using example code **from Spacy's documentation**".* However, the user was able to train a

model without any reference to spaCy's documentation. Hence, this explanation from ChatGPT was unacceptable. During the basic challenge phase, the challenger asked a question *"How comprehensive is the documentation for training a TextCategorizer model, and are there **any limitations or potential issues users might encounter?**"* However, in response to this challenge, ChatGPT did not address the limitation part. To the contrary, ChatGPT responded to the basic challenge as *"The conversation suggests that the documentation for training a TextCategorizer model in Spacy is helpful and straightforward... No limitations or potential issues are discussed."* Then when the challenger component applied question mutation, it added a clause regarding the *limitation* while generating an equivalent mutated question: *"How comprehensive is the documentation for training a textcategorizer model, and are there any limitations or potential issues users might encounter **without considering how might the limitations of online learning affect the accuracy of the model when adding new entities?**"* As the bold part of this question again points to the limitations of online learning, ChatGPT responds to this mutated challenge this way: *"While the documentation for TextCategorizer training is comprehensive, **users may encounter limitations and accuracy issues when adding new entities**".* Thus, this time ChatGPT changes its response from the original response and subsequently in the similarity calculator and prediction model, this was detected as an inconsistent answer and labeled as an incorrect explanation.

> **Summary of RQ5.** In CID, the mutation challenge prompts based on the metamorphic relationships have a more significant role to detect inconsistencies than the base challenge prompts.

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.

Reliability Issues of LLMs. Despite the efficacy of LLMs including ChatGPT, they are often susceptible to generating non-factual statements and even hallucinating facts [7, 13, 23, 27, 54, 77]. Researchers have explored various factors contributing to this issue, such as the quality of training data [40, 69], source-target divergence [17], inadequate modeling [4, 24], and randomness during inference [19]. Several studies leveraged the logit output values as a measure of the model's 'uncertainty' and used them to detect hallucination [35, 66]. 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 [43]. Hence, if a claim can be paraphrased in many different ways, each paraphrase may have a lower log probability, irrespective of its factual correctness. Recently, Azaria and Mitchell leveraged the internal state (i.e., activation values) of the language model to determine the truthfulness of a statement using a separate classifier [5]. Mielke et al. trained a correctness predictor for LLM responses that also directly uses model's internal representations [50]. This approach requires the internal states of the LLM, which may not be available in a black-box situation (like ChatGPT). In contrast, our

CID tool is applicable to such black-box settings that do not require any internal state of LLM. Rather it employs iterative prompting and challenges the LLM to reveal any inconsistency that can be indicative of possible incorrectness in the LLM response. Previous black-box approaches mainly work by generating multiple responses for the same question and capturing the variations in these generations [44, 49, 75]. For example, Manakul et al. present SelfCheckGPT, a black-box method for detecting hallucinations in generative LLMs by examining inconsistencies across multiple answers for the same question [49]. Such methods may overlook redundant misinformation dispersed across multiple responses, masking the presence of hallucinations. Moreover, the variance in responses is subject to the randomness parameters of the language models (as they are generated for the same question). As a result, such approaches are not applicable to deterministic setups. However, instead of comparing multiple generations for the same question, CID exerts follow-up enquiries and challenge questions to the LLM through iterative prompting to capture inconsistency.

Similar to other LLMs, ChatGPT also suffers from a lack of consistency [32, 33]. The model frequently alters its decisions when presented with a paraphrased sentence, revealing self-contradictory behavior. Existing research considers the correctness and consistency of the LLMs as two separate objectives [22], which may not be correct. In this paper, we show that by assessing how and when responses deviate from the previous responses to the same question, we could detect incorrect responses.

Several studies have explored various approaches to mitigate hallucination in LLMs such as fine-tuning the LLM with human feedback and reinforcement learning [6, 52], factuality-enhanced training [41], external fact-checking tool employment [29], improved prompting [23], and external knowledge incorporation into LLM-generated responses [56]. Existing research treats detection and mitigation of inaccuracies in LLM chatbots as two separate problems. Though mitigation of hallucination or factual inaccuracy of LLM does not fall within the scope of this study, our CID tool can warn the users of any possible reliability issue of a given ChatGPT response that might warrant further verification.

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 [25], CodeT5 [72], CoTexT [57] are designed with a focus on addressing the unique challenges and requirements of software development tasks. These architectures rely on large-scale pre-training of both programming (i.e., source code) and natural language (i.e., code comment) data, enabling them to perform various downstream SE tasks. Other noteworthy contributions include Liu et al.'s self-attention neural architecture for code completion and Wei et al.'s dual training framework for code summarization and code generation on GitHub datasets. Additionally, Nijkamp et al.'s CodeGen presents a family of LLMs trained in natural language and programming data that achieved competitive performance in zero-shot Python code generation. In contrast, more general-purpose LLMs (such as GPT-based models [8, 9], Codex [10], ChatGPT [51]) have showcased their versatility in tackling a wide range of NLP and software engineering tasks. These models are also pre-trained on vast amounts of natural and programming language data available on the internet.

Several studies have leveraged such models for various SE tasks such as code generation [26, 37, 45], code repair [55, 58, 74], code summarization [2, 36, 48]. In addition, such models (specifically, ChatGPT) combine conversational capabilities with code-related tasks, allowing programmers to interact with the model. This allows developers to use such models as programming assistants [62].

We are not aware of any study in SE that focused on understanding practitioners' perspectives on the usage of LLMs or their inaccuracies. To fill this gap in SE literature, we first conducted a comprehensive survey of 135 developers and then developed our tool CID. We showed the effectiveness of CID to assess inaccuracies in ChatGPT responses during software library selection.

6 CONCLUSIONS

Summary. In a survey of 135 SE practitioners, we find that they are trying to use Generative AI-based chatbots like ChatGPT to support their diverse SE tasks. However, they often worry about the truthfulness of ChatGPT responses. We developed a tool called CID to detect incorrect ChatGPT responses. CID repeatedly queries ChatGPT with contextually aligned yet textually divergent questions. For a given question, a response that is different from other responses (across multiple incarnations of the question) is likely to be an incorrect response. In a benchmark study involving a dataset of Q&As for software library selection, we find that CID can detect incorrect responses from ChatGPT with an F1-score of 0.74-0.75.

Limitations. The correctness of ChatGPT responses in our training dataset is determined based on the provided SO post content. While we only collected the accepted answers to ensure their validity, the ground truth may vary among human annotators. For the survey, *selection bias* could arise through the recruitment of survey participants. Our survey sample is quite large (135 industrial SE practitioners), which could mitigate any under/over-representation of participants. Another threat is *response bias*, where participants may provide socially desirable answers or omit critical information. This threat was mitigated by ensuring anonymity and confidentiality in the survey responses. In the tool development, *internal validity* threats relate to biases in the dataset used to train and test the CID tool's components. The training and test sets did not share any common data.

Generality of CID and Future Work. The core principle behind CID revolves around probing for inconsistencies in ChatGPT responses. The probing is done by asking the same question in different ways and inconsistencies are detected in the responses to the question (e.g., when one response diverges from the rest). This philosophy is broad enough to be applied to numerous tasks within the realm of SE and beyond. The current version of CID is tested for "textual" responses from ChatGPT. Intuitively, such responses could offer answers to questions related to diverse domains. However, the focus of our paper was to determine the trustworthiness of ChatGPT usage in SE tasks and the design and evaluation of CID to support their assessment of the trustworthiness of ChatGPT responses. We picked responses as reviews that are related to software library selection. Our choice is influenced by extensive research in the SE domain in the last few years on the usage of opinions and reviews to facilitate the usage and selection of software libraries [63, 65]. However, given the broad nature of the underlying

principles of CID, it can be tuned and applied to other SE tasks that could benefit from textual responses from ChatGPT. Other tasks that are mentioned by our survey participants include - (a) Code Analysis and Review, and (b) Exploring Alternative Approaches. In our future work, we aim to assess the generalizability of CID by tuning it for these other similar SE tasks (e.g., code reviews) and by collecting feedback from diverse software professionals during their usage of ChatGPT and our tool CID.

DATA AVAILABILITY

The code and data used for this paper are shared in this anonymous replication package [61].

REFERENCES

[1] Wasi Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. 2021. Unified Pre-training for Program Understanding and Generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2655–2668.

[2] Toufique Ahmed and Premkumar Devanbu. 2022. Few-shot training LLMs for project-specific code-summarization. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 1–5.

[3] Open AI. 2023. *GPT-4 — openai.com*. https://openai.com/research/gpt-4. [Accessed 01-08-2023].

[4] Rahul Aralikatte, Shashi Narayan, Joshua Maynez, Sascha Rothe, and Ryan McDonald. 2021. Focus Attention: Promoting Faithfulness and Diversity in Summarization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online, 6078–6095. https://doi.org/10.18653/v1/2021.acl-long.474

[5] Amos Azaria and Tom Mitchell. 2023. The internal state of an llm knows when its lying. *arXiv preprint arXiv:2304.13734* (2023).

[6] Michiel Bakker, Martin Chadwick, Hannah Sheahan, Michael Tessler, Lucy Campbell-Gillingham, Jan Balaguer, Nat McAleese, Amelia Glaese, John Aslanides, Matt Botvinick, et al. 2022. Fine-tuning language models to find agreement among humans with diverse preferences. *Advances in Neural Information Processing Systems* 35 (2022), 38176–38189.

[7] Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023* (2023).

[8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[9] Paweł Budzianowski and Ivan Vulić. 2019. Hello, It's GPT-2 - How Can I Help You? Towards the Use of Pretrained Language Models for Task-Oriented Dialogue Systems. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*. Association for Computational Linguistics, Hong Kong, 15–22. https://doi.org/10.18653/v1/D19-5602

[10] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* (2021).

[11] Songqiang Chen, Shuo Jin, and Xiaoyuan Xie. 2021. Testing your question answering software via asking recursively. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 104–116.

[12] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044* (2019).

[13] Wenliang Dai, Zihan Liu, Ziwei Ji, Dan Su, and Pascale Fung. 2023. Plausible May Not Be Faithful: Probing Object Hallucination in Vision-Language Pre-training. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. 2128–2140.

[14] Fernando López de la Mora and Sarah Nadi. 2018. An empirical study of metric-based comparisons of software libraries. In *Proceedings of the 14th International Conference on Predictive Models and Data Analytics in Software Engineering*. Association for Computing Machinery, New York, NY, USA, 22–31.

[15] Fernando López De La Mora and Sarah Nadi. 2018. Which library should i use?: A metric-based comparison of software libraries. In *2018 IEEE/ACM 40th International Conference on Software Engineering: New Ideas and Emerging Technologies Results (ICSE-NIER)*. IEEE, Association for Computing Machinery, Gothenburg, Sweden, 37–40.

[16] Haneen Deeb, Aldert Vrij, Lorraine Hope, Samantha Mann, Pär Anders Granhag, and Leif A Strömwall. 2018. Police officers' perceptions of statement inconsistency. *Criminal Justice and Behavior* 45, 5 (2018), 644–665.

[17] Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling Divergent Reference Texts when Evaluating Table-to-Text Generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 4884–4895. https://doi.org/10.18653/v1/P19-1483

[18] Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2023. Chain-of-verification reduces hallucination in large language models. *arXiv preprint arXiv:2309.11495* (2023).

[19] Nouha Dziri, Andrea Madotto, Osmar R Zaiane, and Avishek Joey Bose. 2021. Neural Path Hunter: Reducing Hallucination in Dialogue Systems via Path Grounding. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2197–2214.

[20] Steffen Eger and Yannik Benz. 2020. From Hero to Z\'eroe: A Benchmark of Low-Level Adversarial Attacks. *arXiv preprint arXiv:2010.05648* (2020).

[21] Rehab El-Hajj and Sarah Nadi. 2020. LibComp: An IntelliJ plugin for comparing java libraries. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. Association for Computing Machinery, New York, NY, USA, 1591–1595.

[22] Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics* 9 (2021), 1012–1031.

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

[24] Yang Feng, Wanying Xie, Shuhao Gu, Chenze Shao, Wen Zhang, Zhengxin Yang, and Dong Yu. 2020. Modeling fluency and faithfulness for diverse neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 59–66.

[25] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. 2020. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. 1536–1547.

[26] James Finnie-Ansley, Paul Denny, Brett A Becker, Andrew Luxton-Reilly, and James Prather. 2022. The robots are coming: Exploring the implications of openai codex on introductory programming. In *Proceedings of the 24th Australasian Computing Education Conference*. 10–19.

[27] Boris A Galitsky. 2023. Truth-O-Meter: Collaborating with LLM in Fighting its Hallucinations. (2023).

[28] Leo A. Goodman. 1961. Snowball Sampling. *Annals of Mathematical Statistics* 32, 1 (1961), 148–170.

[29] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2023. Critic: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738* (2023).

[30] Lynette Hirschman and Robert Gaizauskas. 2001. Natural language question answering: the view from here. *natural language engineering* 7, 4 (2001), 275–300.

[31] Yi Huang, Chunyang Chen, Zhenchang Xing, Tian Lin, and Yang Liu. 2018. Tell them apart: distilling technology differences from crowd-scale comparison discussions. In *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, Association for Computing Machinery, New York, NY, USA, 214–224.

[32] Myeongjun Jang, Deuk Sin Kwon, and Thomas Lukasiewicz. 2022. BECEL: Benchmark for Consistency Evaluation of Language Models. In *Proceedings of the 29th International Conference on Computational Linguistics*. 3680–3696.

[33] Myeongjun Jang and Thomas Lukasiewicz. 2023. Consistency analysis of chatgpt. *arXiv preprint arXiv:2303.06273* (2023).

[34] Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328* (2017).

[35] Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How Can We Know When Language Models Know? On the Calibration of Language Models for Question Answering. *Transactions of the Association for Computational Linguistics* 9 (2021), 962–977. https://doi.org/10.1162/tacl_a_00407

[36] Junaed Younus Khan and Gias Uddin. 2022. Automatic code documentation generation using gpt-3. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 1–6.

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

[38] Gary LJ Lancaster, Aldert Vrij, Lorraine Hope, and Bridget Waller. 2013. Sorting the liars from the truth tellers: The benefits of asking unanticipated questions on lie detection. *Applied Cognitive Psychology* 27, 1 (2013), 107–114.

[39] Enrique Larios Vargas, Maurício Aniche, Christoph Treude, Magiel Bruntink, and Georgios Gousios. 2020. Selecting third-party libraries: The practitioners' perspective. In *Proceedings of the 28th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*. 245–256.

[40] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating Training Data Makes Language Models Better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Dublin, Ireland, 8424–8445. https://doi.org/10.18653/v1/2022.acl-long.577

[41] Nayeon Lee, Wei Ping, Peng Xu, Mostofa Patwary, Pascale N Fung, Mohammad Shoeybi, and Bryan Catanzaro. 2022. Factuality enhanced language models for open-ended text generation. *Advances in Neural Information Processing Systems* 35 (2022), 34586–34599.

[42] Xiaozhou Li, Sergio Moreschini, Zheying Zhang, and Davide Taibi. 2022. Exploring factors and metrics to select open source software components for integration: An empirical study. *Journal of Systems and Software* 188 (2022), 111255.

[43] Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Teaching models to express their uncertainty in words. *arXiv preprint arXiv:2205.14334* (2022).

[44] Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. 2023. Generating with Confidence: Uncertainty Quantification for Black-box Large Language Models. *arXiv preprint arXiv:2305.19187* (2023).

[45] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *arXiv preprint arXiv:2305.01210* (2023).

[46] Mingwei Liu, Xin Peng, Andrian Marcus, Shuangshuang Xing, Christoph Treude, and Chengyuan Zhao. 2021. API-Related Developer Information Needs in Stack Overflow. *IEEE Transactions on Software Engineering* 48, 11 (2021), 4485–4500.

[47] Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. 2021. Entity-based knowledge conflicts in question answering. *arXiv preprint arXiv:2109.05052* (2021).

[48] Stephen MacNeil, Andrew Tran, Dan Mogil, Seth Bernstein, Erin Ross, and Ziheng Huang. 2022. Generating diverse code explanations using the gpt-3 large language model. In *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 2*. 37–39.

[49] Potsawee Manakul, Adian Liusie, and Mark JF Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896* (2023).

[50] Sabrina J Mielke, Arthur Szlam, Emily Dinan, and Y-Lan Boureau. 2022. Reducing conversational agents' overconfidence through linguistic calibration. *Transactions of the Association for Computational Linguistics* 10 (2022), 857–872.

[51] OpenAI. [n. d.]. Online-ChatGPT - Optimizing Language Models for Dialogue. https://online-chatgpt.com/. [Accessed 31-07-2023].

[52] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* 35 (2022), 27730–27744.

[53] Stack Overflow. 2023. *Stack Overflow Search*. Stack Overflow, https://stackoverflow.com/search. Last updated on 01 August 2023.

[54] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334* (2023).

[55] Hammond Pearce, Benjamin Tan, Baleegh Ahmad, Ramesh Karri, and Brendan Dolan-Gavitt. 2023. Examining zero-shot vulnerability repair with large language models. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2339–2356.

[56] Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813* (2023).

[57] Long Phan, Hieu Tran, Daniel Le, Hieu Nguyen, James Annibal, Alec Peltekian, and Yanfang Ye. 2021. CoTexT: Multi-task Learning with Code-Text Transformer. In *Proceedings of the 1st Workshop on Natural Language Processing for Programming (NLP4Prog 2021)*. Association for Computational Linguistics, Online, 40–47. https://doi.org/10.18653/v1/2021.nlp4prog-1.5

[58] Julian Aron Prenner and Romain Robbes. 2021. Automatic Program Repair with OpenAI's Codex: Evaluating QuixBugs. *arXiv preprint arXiv:2111.03922* (2021).

[59] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. *arXiv preprint arXiv:2005.04118* (2020).

[60] Qingchao Shen, Junjie Chen, Jie M Zhang, Haoyu Wang, Shuang Liu, and Menghan Tian. 2022. Natural test generation for precise testing of question answering software. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 1–12.

[61] Minaoar Hossain Tanzil. 2023. *ChatGPT Incorrectness Detection in Software Reviews*. University of Calgary, https://github.com/minaoar/ChatGPT-Incorrectness-Detection-in-Software-Reviews.

[62] Haoye Tian, Weiqi Lu, Tsz On Li, Xunzhu Tang, Shing-Chi Cheung, Jacques Klein, and Tegawendé F Bissyandé. 2023. Is ChatGPT the Ultimate Programming Assistant–How far is it? *arXiv preprint arXiv:2304.11938* (2023).

[63] Gias Uddin, Olga Baysal, Latifa Guerrouj, and Foutse Khomh. 2019. Understanding how and why developers seek and analyze API-related opinions. *IEEE Transactions on Software Engineering* 47, 4 (2019), 694–735.

[64] Gias Uddin and Foutse Khomh. 2017. Automatic summarization of API reviews. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, IEEE Press, Urbana-Champaign, IL, USA, 159–170.

[65] Gias Uddin and Foutse Khomh. 2017. Opiner: an opinion search and summarization engine for APIs. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, IEEE Press, Urbana-Champaign, IL, USA, 978–983.

[66] Neeraj Varshney, Wenlin Yao, Hongming Zhang, Jianshu Chen, and Dong Yu. 2023. A Stitch in Time Saves Nine: Detecting and Mitigating Hallucinations of LLMs by Validating Low-Confidence Generation. *arXiv preprint arXiv:2307.03987* (2023).

[67] Aldert Vrij, Pär Anders Granhag, Samantha Mann, and Sharon Leal. 2011. Outsmarting the liars: Toward a cognitive lie detection approach. *Current Directions in Psychological Science* 20, 1 (2011), 28–32.

[68] Aldert Vrij, Sharon Leal, Pär Anders Granhag, Samantha Mann, Ronald P Fisher, Jackie Hillman, and Kathryn Sperry. 2009. Outsmarting the liars: The benefit of asking unanticipated questions. *Law and human behavior* 33 (2009), 159–166.

[69] Hongmin Wang. 2019. Revisiting Challenges in Data-to-Text Generation with Fact Grounding. In *Proceedings of the 12th International Conference on Natural Language Generation*. Association for Computational Linguistics, Tokyo, Japan, 311–322. https://doi.org/10.18653/v1/W19-8639

[70] Han Wang, Chunyang Chen, Zhenchang Xing, and John Grundy. 2020. DiffTech: A tool for differencing similar technologies from question-and-answer discussions. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. Association for Computing Machinery, New York, NY, USA, 1576–1580.

[71] Han Wang, Chunyang Chen, Zhenchang Xing, and John Grundy. 2021. DiffTech: Differencing Similar Technologies from Crowd-Scale Comparison Discussions. *IEEE Transactions on Software Engineering* 48, 7 (2021), 2399–2416.

[72] Yue Wang, Weishi Wang, Shafiq Joty, and Steven CH Hoi. 2021. CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 8696–8708.

[73] Anthony I Wasserman, Xianzheng Guo, Blake McMillian, Kai Qian, Ming-Yu Wei, and Qian Xu. 2017. OSSpal: finding and evaluating open source software. In *IFIP International Conference on Open Source Systems*. Springer, Cham, Springer International Publishing, Cham, 193–203.

[74] Chunqiu Steven Xia and Lingming Zhang. 2023. Keep the Conversation Going: Fixing 162 out of 337 bugs for $0.42 each using ChatGPT. *arXiv preprint arXiv:2304.00385* (2023).

[75] Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. 2023. Can LLMs Express Their Uncertainty? An Empirical Evaluation of Confidence Elicitation in LLMs. *arXiv preprint arXiv:2306.13063* (2023).

[76] Litao Yan, Miryung Kim, Bjoern Hartmann, Tianyi Zhang, and Elena L Glassman. 2022. Concept-Annotated Examples for Library Comparison. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery, Bend, OR, USA, 1–16.

[77] Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A Smith. 2023. How language model hallucinations can snowball. *arXiv preprint arXiv:2305.13534* (2023).

[78] Zhuosheng Zhang, Hai Zhao, and Rui Wang. 2020. Machine reading comprehension: The role of contextualized language models and beyond. *arXiv preprint arXiv:2005.06249* (2020).