ChatGPT and Software Testing Education: Promises & Perils

5 authors, including:

Sajed Jalil
George Mason University
2 PUBLICATIONS   **11** CITATIONS

Suzzana Rafi
George Mason University
2 PUBLICATIONS   **11** CITATIONS

Thomas D. LaToza
University of California, Irvine
79 PUBLICATIONS   **2,412** CITATIONS

Some of the authors of this publication are also working on these related projects:

Microtask programing View project

ChatGPT and Software Testing Education: Promises & Perils

Sajed Jalil ⬤
sjalil@gmu.edu

Suzzana Rafi ⬤
srafi@gmu.edu

Thomas D. LaToza ⬤
tlatoza@gmu.edu

Kevin Moran ⬤
kpmoran@gmu.edu

Wing Lam ⬤
winglam@gmu.edu

Department of Computer Science
George Mason University
Fairfax, USA

Abstract—**Over the past decade, predictive language modeling for code has proven to be a valuable tool for enabling new forms of automation for developers. More recently, we have seen the advent of general purpose "large language models", based on neural transformer architectures, that have been trained on massive datasets of human written text, which includes code and natural language. However, despite the demonstrated representational power of such models, interacting with them has historically been constrained to specific task settings, limiting their general applicability. Many of these limitations were recently overcome with the introduction of ChatGPT, a language model created by OpenAI and trained to operate as a conversational agent, enabling it to answer questions and respond to a wide variety of commands from end users.**

The introduction of models, such as ChatGPT, has already spurred fervent discussion from educators, ranging from fear that students could use these AI tools to circumvent learning, to excitement about the new types of learning opportunities that they might unlock. However, given the nascent nature of these tools, we currently lack fundamental knowledge related to how well they perform in different educational settings, and the potential promise (or danger) that they might pose to traditional forms of instruction. As such, in this paper, we examine how well ChatGPT performs when tasked with answering practice questions in a popular software testing curriculum. We found that given its current capabilities, ChatGPT is able to respond to 77.5% of the questions we examined and that, of these questions, it is able to provide correct or partially correct *answers* in 55.6% of cases, provide correct or partially correct *explanations* of answers in 53.0% of cases, and that prompting the tool in a shared question context leads to a marginally higher rate of correct answers and explanations. Based on these findings, we discuss the potential promises and perils related to the use of ChatGPT by students and instructors.

Index Terms—**ChatGPT, testing, education, case study**

I. INTRODUCTION

Language modeling of code has been an important topic in software engineering research since the promise of modeling code was first illustrated by Hindle et al. [1]. As techniques for language modeling improved, researchers began to utilize Deep Learning (DL) architectures to learn rich, hierarchical representations of code that could then be used for various downstream tasks [2]. In parallel, the machine learning and natural language processing communities began building large-scale models centered around a specific type of neural architecture, the transformer [3]–[5], trained on massive datasets

of text. Experiments illustrated the representational power of both these large language models (LLMs), and language models tailored specifically for code [6]–[9]. However, such models were largely constrained to specific task settings and did not provide for *natural* forms of interaction with end users – until recently.

In late 2022, OpenAI introduced ChatGPT [10], an AI tool built on top of existing LLMs, that enabled interaction through a conversational interface. To enable this type of interaction, OpenAI made use of reinforcement learning from human feedback, refining methods from past work on InstructGPT [11], which trained LLMs with both unsupervised data and with supervision in the form of task instruction. In essence, the model was initially trained on real, human text-based conversations, then learned to refine its responses based on feedback from human evaluators that rated the quality of answers in a reinforcement learning setting. This process proved very successful in creating an interface where users could easily access the latent "knowledge" of LLMs.

Given the ease of interaction and the seemingly vast amount of knowledge contained within the model, vigorous discussion arose in academic communities about the potential dangers and opportunities of such a tool for both students and instructors. Sentiments ranged from fear that students would use these models to circumvent learning material, to excitement about the new types of learning, assignments, and automation a tool, such as ChatGPT, could introduce into various levels of academia. However, to make informed decisions about how to use (or not use) ChatGPT in various educational settings, we must first have a thorough understanding of its capabilities, strengths, and weaknesses.

In this paper, we aim to study and understand the capabilities of ChatGPT in the context of a traditional software testing course. As such, we conduct a comprehensive empirical study, tasking ChatGPT with answering questions from five chapters of a popular software testing textbook [12], and thoroughly vetting the results across multiple dimensions. We aim to learn (i) how often ChatGPT is correct in answering questions, (ii) how often it can fully and accurately explain its answers, (iii) how different ways of asking questions to ChatGPT can affect its ability to provide correct responses, and (iv) whether ChatGPT's expressed confidence provides

bearing on the correctness of its answers. We find the current capabilities of ChatGPT allow it to properly respond to 77.5% of the testing questions we examined. For the questions to which ChatGPT was able to respond, it provides correct or partially correct *answers* in 55.6% of cases, and provides correct or partially correct *explanations* in 53.0% of cases We also found that prompting the model in a shared context, where similar questions are asked together, lead to marginally better answers, and the tool's claimed confidence level seems to have little bearing on the correctness of the answers. Based on these findings, we discuss the potential promises and perils related to the use of ChatGPT in software testing courses.

In summary, the contributions of this paper are as follows:

- A manually-vetted dataset of ChatGPT's answers to 31 questions from five chapters of a popular software testing textbook. Our dataset includes three responses from ChatGPT for each question.
- A thorough analysis of these answers that examines how often ChatGPT is correct and when it is able to properly explain a given answer.
- An investigation into two prompting strategies, and their effect on answer and explanation correctness, as well as an analysis of whether ChatGPT's proclaimed confidence level impacts answer/explanation correctness.
- An online appendix [13], [14], that includes our data, analysis code, and experimental infrastructure to facilitate replicability and future work on applications of ChatGPT to various topics in computer science education.

II. BACKGROUND

ChatGPT [10] offers a machine learning model designed to engage in conversations with the user. It provides responses to questions asked in a prompt and is able to respond to follow-up questions and correct itself.

To investigate the applicability of ChatGPT to answer questions commonly found in a software testing curriculum, we pose questions from the first five chapters of the textbook by Ammann and Offutt [12]. This book has been used in undergraduate and graduate software testing classes at George Mason University and is used by hundreds of organizations as a source of software testing knowledge. Our study uses:

- Chapters 1 and 2, which discuss software faults, errors, and failures
- Chapter 3, which discusses the Reachability, Infection, Propagation, and Revealability (RIPR) model
- Chapter 4, which discusses Test Driven Development (TDD) and continuous integration
- Chapter 5, which discusses coverage criterion and subsumption

The remainder of this section provides a brief description of the course content found in each chapter.

A. Chapters 1& 2 - Fault, Error and Failure

A *fault* is a static defect in the software. An *error* is an incorrect internal state, which is composed of a program counter and the live variables at that program counter location, and a *failure* is an external, incorrect behavior with respect to the requirements or the description of the expected behavior [12].

B. Chapter 3 - RIPR Model

According to Ammann and Offutt [12], there are four conditions that are needed for a failure to be observed. These conditions together are called the RIPR model. First, a test needs to reach the location of the defective line of code – that is *Reachability*. After the fault is executed, it leads to an incorrect program state, which is called *Infection*. The infection must spread to an incorrect final state – that is *Propagation* and finally, the incorrect portion of the final state must be observable by the tester – that is *Revealability*.

C. Chapter 4 - CI & TDD

Ammann and Offutt [12] describe Continuous Integration (CI) as the process that begins with a developer using a fresh development environment, obtaining the code under test and test code, building the code, and running the tests. After finalizing changes to code, the changes start the CI process, where a fresh environment rebuilds the code and reruns the tests. This process helps developers quickly identify any failures their changes may have introduced.

Test Driven Development (TDD) is a methodology for creating software in which tests are written before the code under test. The methodology is based on repeating a short development cycle, including creating a test, running it to confirm that it fails, writing code under test to make the test pass, creating more tests, and improving the code under test to make more tests pass. TDD aims to produce clean and failure-free code by writing only code under test to make tests pass.

D. Chapter 5 - Coverage Criterion and Subsumption

A coverage criterion is a rule or collection of rules that impose test requirements on a test set. A coverage criterion C1 subsumes C2 if and only if every test set that satisfies criterion C1 also satisfies C2.

III. STUDY SETUP

To investigate the promises and perils of using ChatGPT for software testing education, we study the following research questions (RQs):

- **RQ$_1$**: How often is ChatGPT able to provide correct answers and explanations for different prompting strategies?
- **RQ$_2$**: How often does ChatGPT give answer-explanation pairs with different degrees of correctness?
- **RQ$_3$**: How does ChatGPT's non-determinism affect its ability to provide correct answers and explanations?
- **RQ$_4$**: How does ChatGPT's confidence in its response correlate to the correctness of the response?

A. Dataset

Our dataset contains questions from the widely used software testing book by Ammann and Offutt [12]. In the context of this study, we use all the textbook questions in Chapters 1 to 5 that have solutions available on the book's official website [15]. These solutions are made publicly available to help

TABLE I
OVERVIEW OF THE QUESTIONS IN OUR STUDY.

Chapter	Question	Sub-Question	Code	Concept	Both
1	5.2.a	✓			✓
1	5.2.b	✓			✓
1	5.2.c	✓			✓
1	5.2.d	✓			✓
1	5.2.e	✓		✓	
1	5.4.a	✓			✓
1	5.4.b	✓			✓
1	5.4.c	✓			✓
1	5.4.d	✓			✓
1	5.4.e	✓		✓	
1	7.2.a	✓			✓
1	7.2.b	✓			✓
1	7.2.c	✓			✓
1	7.2.d	✓			✓
1	7.2.e	✓		✓	
1	7.3.a	✓			✓
1	7.3.b	✓			✓
1	7.3.c	✓			✓
1	7.3.d	✓			✓
1	7.3.e	✓		✓	
2	1			✓	
3	4		✓		
3	5			✓	
3	9.a	✓	✓		
3	9.b	✓		✓	
3	9.c	✓	✓		
3	9.d	✓	✓		
3	9.e	✓	✓		
4	1		✓		
5	1.a	✓		✓	
5	1.b	✓		✓	
Count	**31**	**27**	**6**	**9**	**16**

students learn. We omitted questions that do not have student solutions, as publishing our results might expose answers that the authors of the book do not intend to make public. Our study is limited to the first five chapters of the book, which emphasize topics taught in most introductory software testing courses. Often, our selected questions have *sub-questions*, which break down a more comprehensive question into smaller parts. For example, for a given code snippet, there might be multiple related sub-questions that each ask about properties of the snippet. In our study, for simplicity, we refer to all questions and sub-questions as *questions* - given that we treat them all as having equal importance.

We collected a total of 40 exercise questions that meet our requirements. After manual inspection, we identified nine questions that ask for material that is impossible for ChatGPT to generate, as it is capable of generating only text-based responses. For example, we encountered questions that ask for a screen printout of code execution, a project to be fetched from the internet, or a continuous integration server to be set up. Questions with such tasks cannot be fully and correctly answered by ChatGPT's text-based responses.

We removed the nine questions that are outside of Chat-GPT's capabilities to correctly answer and report our results on a total of 31 questions to which ChatGPT may give a correct response. Of 31 questions, six are multi-part questions that collectively contain 27 sub-questions and four are independent questions that do not contain any sub-questions.

Table I lists the characteristics of each question in our dataset. We find that six questions ask only for code in their answers, nine ask to explain a concept, and the remaining 16

ask for both code and a concept explanation. Our dataset is publicly available on GitHub [13] and Zenodo [14].

B. Data Collection Tool

During the data collection process for this study, Ope-nAI [16] had not yet made a ChatGPT API publicly available, therefore, the number of questions we were able to ask Chat-GPT through the online interface during a given period of time was rate-limited. We developed an open-source tool to collect ChatGPT responses for the questions in our dataset [13]. Our tool automatically queries ChatGPT, collects the responses, and waits 10 seconds after receiving an answer before asking the next question. We determined the length of this delay based on experiments with our automated tool (e.g., trying multiple wait times). We find that a wait time of 10 seconds provided us the greatest number of query responses.

C. Methodology

For **RQ**$_1$, we look to understand how often ChatGPT is able to provide correct answers and explanations to our dataset of software testing questions and to determine how ChatGPT performs when sub-questions are asked in a single chat context one by one compared to when they are asked in separate contexts. We refer to these two ways of asking questions as *shared context* and *separate context*, respectively. For **RQ**$_2$, we study how often ChatGPT will give answer-explanation pairs with different degrees of correctness (e.g., correct answer but incorrect explanation). For **RQ**$_3$, we aim to analyze how ChatGPT's non-determinism affects its ability to provide correct answers and explanations by posing each question three times and examining any differences in responses. Lastly, for **RQ**$_4$, we aim to determine whether ChatGPT's self-reported confidence (which can be collected through a *confidence query*) related to an answer/explanation has a bearing on the correctness of that answer. Our findings could be useful for instructors and students to determine the potential utility of a given answer. Below, we define the processes for (1) *shared context queries*, (2) *separate context queries*, (3) *confidence queries*, and (4) *response labeling*.

1) Separate Context Query: In *separate* context queries, we treat each of the 27 sub-questions as an independent question. Each sub-question is asked in a separate chat context. Combining with the four independent questions, a total of 31 questions are asked for each run. To evaluate how separate context compares with shared context, we collect a total of three runs for each question, which results in a total of 93 separate context responses.

2) Shared Context Query: In this query scenario, sub-questions are all asked in a single ChatGPT session during which the context of the conversation is shared (i.e., ChatGPT is able to reference an initial prompt or code snippet, and parts of prior sub-questions) as long as the sub-questions refer to the same code or scenario. For example, consider the `lastZero()` method in Figure 1. `lastZero()` is supposed to find the last index in an array where a zero occurs. One may ask multiple sub-questions based on this code (e.g., give

```java
public static int lastZero (int[] x) {
  for (int i = 0; i < x.length; i++)
    if (x[i] == 0) return i;
  return -1;
}
```

Fig. 1. Code snippet for `lastZero()`.

a test that (1) does not execute the fault or (2) executes the fault, but there are no failures).

In the shared context query, we provide the implementation of `lastZero()` and ask the first sub-question in one chat context. After we obtain the response, we continue to ask the next sub-question in the same, *shared* context. This process is repeated until all sub-questions are asked before the context is destroyed. For the next question with multiple sub-questions, we then open another chat context.

Overall, we obtain 81 responses from running each of the 27 sub-questions three times and 12 responses from running each of the four independent questions three times. For questions with no sub-questions, shared context is the same as separate context, as such, the responses to the four independent questions are identical across the datasets for both the shared and separate contexts, with only the 81 responses to the 27 sub-questions differing. In $\mathbf{RQ_1}$, we compare responses for which both shared context and separate context exists, i.e., 81 responses. We find that shared context is more likely to give correct responses than separate context. In $\mathbf{RQ_2}$-$\mathbf{RQ_4}$ we use the responses from shared context – the 81 shared context responses and the 12 shared/separate context responses.

3) Confidence Query: To determine how confident Chat-GPT is in its answer for RQ4, we ask the following question to ChatGPT after each of its responses – *"How confident are you that your previous response is correct?"*. To answer $\mathbf{RQ_1}$-$\mathbf{RQ_3}$, we asked each question 3 times. However, to simplify our data collection process, we only collected responses to confidence queries the first time that a question was posed to ChatGPT in the shared context setting. We recorded the following replies from ChatGPT: Highly confident, Very Confident, Confident, and Reliable. Note that we do not attempt to rank ChatGPT's confidence replies, as their relative ranking is unclear (e.g., whether "Highly confident" is more or less confident than "Very confident"). We attempted to resolve the ranking of these confidence replies by asking ChatGPT to provide a relative ranking, but ChatGPT was not able to provide a conclusive response. Therefore, for $\mathbf{RQ_4}$, we simply study the likelihood for ChatGPT to give a correct explanation or answer given a specific category of confidence reply.

4) Response Labeling: To determine the correctness of ChatGPT's responses, two authors independently inspected the responses and labeled them. For any disagreements, a third author was involved and the questions were discussed so that all three authors came to an agreement. All three evaluators have either (1) taken a graduate-level software testing course that uses the textbook [12] our study questions are sourced from, (2) taught a software testing course that uses the textbook as a teaching assistant, or (3) taught a software

testing course that uses the textbook as the instructor.

Responses from ChatGPT were broken down into two parts - the *explanation* and *answer*. Our labeling considered two perspectives: (1) whether the overall answer was correct and (2) whether the explanation given was correct. Both perspectives are categorized as *correct*, *incorrect*, or *partially correct* relative to the solution of the question.

A ChatGPT response example to a question is the following:

> **Textbook question [12, p.114]:** Suppose that coverage criterion C1 subsumes coverage criterion C2. Further, suppose that test set T1 satisfies C1 on program P, and test set T2 satisfies C2, also on P. Does T1 necessarily satisfy C2? Explain.

> **ChatGPT's response:** T1 may or may not satisfy C2. The fact that T1 satisfies C1, which subsumes C2, means that C1 is a more comprehensive criterion that includes all of the requirements of C2. However, it does not guarantee that T1 will also satisfy C2. T1 may include test cases that are not necessary to satisfy C2, and it may not include test cases that are necessary to satisfy C2. Therefore, additional test cases may be needed to confirm that T1 also satisfies C2.

The above example response was labeled as having the answer as *Incorrect*, but the explanation as *Partially Correct*. As ChatGPT's answer is that T1 may **not** satisfy C2 (when the correct answer is that T1 **does** satisfy C2), we label the answer as incorrect. The explanation is partially correct, because ChatGPT correctly mentioned that C1 includes all of C2's requirements (based on the definition of subsumption in Section II-D), but also incorrectly explains that this fact does not guarantee that T1 satisfies C2.

IV. RESULTS

A. $\mathbf{RQ_1}$: How often is ChatGPT able to provide correct answers and explanations for different prompting strategies?

1) Answer Correctness: We find that in shared contexts, 49.4% of the time the answer is correct, and 6.2% of the time it is partially correct. In contrast, in separate contexts, responses are correct 34.6% of the time and partially correct 7.4% of the time. As shown in Figure 2, a shared context produces fewer incorrect answers than separate contexts, on average. One explanation for this behavior is that, in a shared context, ChatGPT obtains contextual information from the prior sub-questions. For example, if ChatGPT already identified a fault in a prior sub-question, it is more likely to correctly leverage this fault to answer subsequent questions about errors.

2) Explanation Correctness: The results for explanation correctness are shown in Figure 3. Here, we obtain similar results to answer correctness. Namely, explanation accuracy is higher in the shared context. That being said, when we focus on being fully correct in answers and explanations, we find that shared context is better than separate context. With this finding in mind, our remaining RQs focus solely on our shared context results. Separate context related data for the remaining RQs is on our website [13].
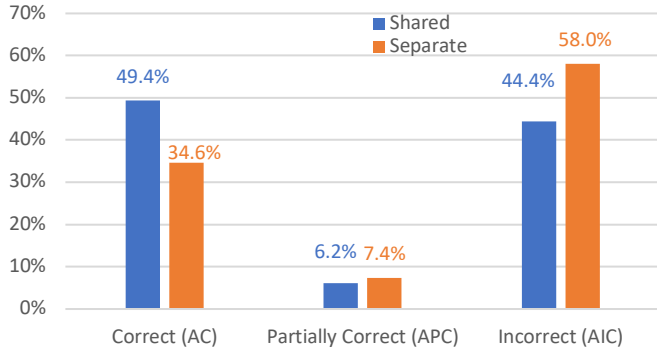
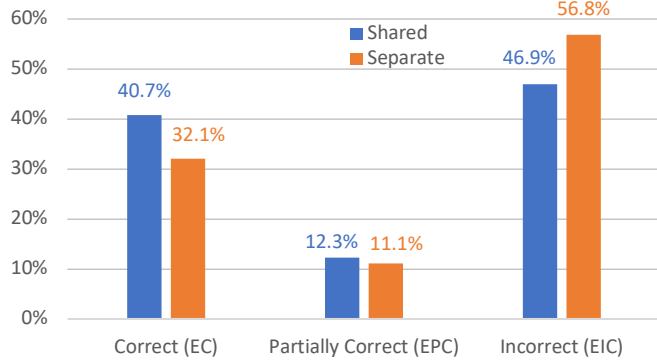Fig. 2. Correctness of ChatGPT answers for shared and separate contexts.

Fig. 3. Correctness of ChatGPT explanations for shared and separate contexts.

> **Shared context is more likely than separate context to be correct. Using ChatGPT in a shared context can result in a correct or partially correct answer 55.6% of the time and a correct or partially correct explanation 53.0% of the time.**

B. *RQ₂: How often does ChatGPT give answer-explanation pairs with different degrees of correctness?*

Table II shows results of correctness for answer-explanation pairs across the three iterations (e.g., three responses for each question) for our defined shared query context. For example, the Answer Incorrect (AIC) and Explanation Partially Correct (EPC) pair means ChatGPT provided an incorrect answer but a partially correct explanation. Of the nine possible pairs, there are three pairs where the degree of correctness of the answer and the explanation are the same: (1) AC-EC where both the answer and the explanation are correct, (2) APC-EPC where both are partially correct, or (3) AIC-EIC where both are incorrect. All other pairs have different degrees of correctness. Our results suggest that, on average, 11.8% of the time, ChatGPT is giving an answer that does not properly match its explanation in terms of correctness.

> **11.8% of the time ChatGPT produces responses where the answer-explanation pairs have different degrees of correctness (e.g., the answer is correct, but the explanation is not).**

Iter.	AC			APC			AIC		
	EC	EPC	EIC	EC	EPC	EIC	EC	EPC	EIC
1	15	0	2	0	1	0	0	2	11
2	15	0	2	0	2	0	0	1	11
3	15	1	2	0	2	0	0	1	10
Sum	45	1	6	0	5	0	0	4	32
%	48.4	1.1	6.4	0.0	5.4	0.0	0.0	4.3	34.4

C. *RQ₃: How does ChatGPT's non-determinism affect its ability to provide correct answers and explanations?*

When asked the same question multiple times, ChatGPT may give a different response each time due to the stochastic nature of its sampling process from a learned probability distribution. We examine how often these differing responses given by ChatGPT will differ in correctness. For example, a question may have a correct answer in one run but an incorrect answer in another run. We find that for 9.7% of questions, the answer's correctness is affected by non-determinism and for 6.5% of questions, the explanation's correctness is affected.

> **The correctness of ChatGPT's answers vary between correct to incorrect for 9.7% of questions, and the correctness of explanations varies for 6.5% of questions.**

D. *RQ₄: How does ChatGPT's confidence in its response correlate to the correctness of the response?*

To determine how confident ChatGPT is in its answers, we asked it to report its confidence. Asking about confidence is typically referred to as "calibration", and a well-calibrated model will be confident when correct, and less confident when incorrect. In our experiment, ChatGPT responded with four different keywords. Figures 4 and 5 display the data for each keyword for answers and explanations, respectively.

ChatGPT expressed varying levels of confidence in its responses. When ChatGPT is "Highly confident" in its response, we find that its answers are incorrect about half the time and explanations are incorrect twice as often as correct. On the other hand, when ChatGPT is "Confident" in its response, we find that its answers or explanations are at least three times as likely to be correct than incorrect. For the other two categories, we find that ChatGPT provides mixed responses.

> **ChatGPTs self-reported confidence does not appear to be particularly useful, as it has little bearing on question correctness. This finding seems to indicate, that, for software testing questions, ChatGPT is not well calibrated.**
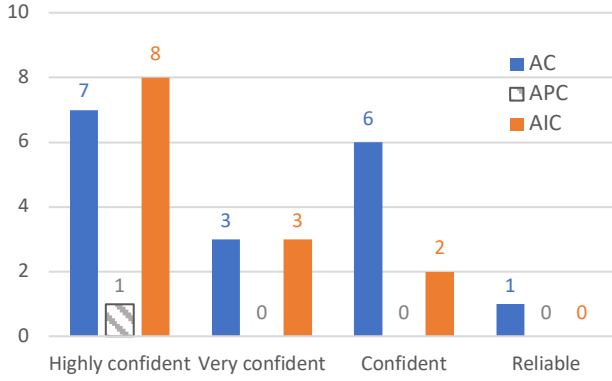
Fig. 4. ChatGPT's reported confidence for correct, partially correct, and incorrect answers.
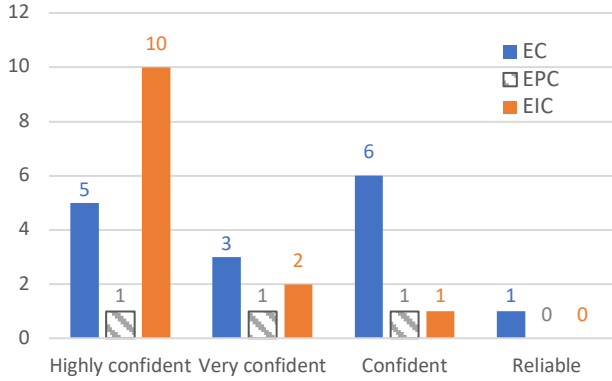

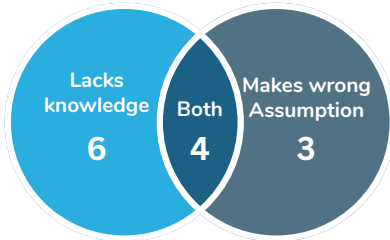Fig. 5. ChatGPT's reported confidence for correct, partially correct, and incorrect explanations.

Fig. 6. Venn diagram of characteristics of 13 incorrect answers from ChatGPT.

V. CASE STUDY

In this section, we present a case study examining (1) the characteristics of incorrect answers, (2) how ChatGPT's responses may change when we provide it with more information, and (3) an example of an inconsistent answer-explanation pair. The goal of our case study is to gather insight into why ChatGPT is wrong and to examine how certain prompting strategies may lead to a higher likelihood of correct answers.

A. Characteristics of Incorrect Answers from ChatGPT

To better understand why ChatGPT is wrong, we categorized 13 incorrect answers (out of 31) from shared context's first iteration, and identified three main reasons for ChatGPT's incorrect responses. Figure 6 summarizes our findings.

1) ChatGPT lacks knowledge: The first category is when ChatGPT may lack the knowledge needed to solve the problems given to it. For questions from chapters 1, 2, and 3, ChatGPT seemed to lack definitions for fault, failure, and error,

resulting in incorrectly treating errors as failures or crashes. For Chapter 5, ChatGPT seemed to lack understanding of how to conclude whether a test set satisfies a coverage criterion. An example of ChatGPT getting the answer incorrect due to lack of knowledge is presented in Section III-C4.

2) ChatGPT makes the wrong assumption: ChatGPT can also focus on an irrelevant part of the question and give an incorrect answer because it makes an incorrect assumption about what is important. See Section V-B for an example.

3) Both: In four cases, ChatGPT seems to both lack knowledge and make wrong assumptions. For example, for one of the four cases, ChatGPT makes a wrong assumption about the fault in a program and lacks the knowledge for what is an error. Therefore, ChatGPT's response about the error in this program was incorrect due to both characteristics.

B. Effect of Additional Information on ChatGPT's responses

To understand the effect of additional information on Chat-GPT, we randomly select one of the 13 incorrect answers mentioned in Section V-A and manipulate the prompt.

Textbook question [12, p.41]:

```
1  public static int oddOrPos (int[] x) {
2      int count = 0;
3      for (int i = 0; i < x.length; i++) {
4          if (x[i]%2 == 1 || x[i] > 0) count++;
5      }
6      return count;
7  }
8  // test: x = [-3,-2,0,1,4]; Expected = 3
```

(a) Explain what is wrong with the given code. Describe the fault precisely by proposing a modification to the code.

The function `oddOrPos()` from the textbook (shown above) is about finding odd or positive numbers. But the given code snippet fails to count negative odd values because a negative odd number produces -1 when it is `mod` with 2. To answer this question, ChatGPT should find the fault and show a possible modification to the program. ChatGPT originally thought the fault is related to not having a null check at the beginning for the input of a null array. But the actual fault is that it does not take into account negative odd values. In its response, ChatGPT advises adding a null check before the `for` loop, which is an incorrect fix.

Modified prompt:

```
1  public static int oddOrPos (int[] x) { ... }
```

(a) ... The fault and the modification do not involve having a null check.

We then changed the sub-question (a) to provide more information (see above). The modified prompt mentions that the fault and the modification are not related to a null check. The textbook solution refers to changing the `if` condition to `x[i]%2 == -1 || x[i] > 0`, where the negative odd

numbers are processed by the first part of the expression. With these modifications to the prompt, ChatGPT's response is correct (see below): it checks for both positive and negative odd numbers with `x[i]%2 != 0`.

> **ChatGPT's response to modified prompt:** The given code has a logical error. The condition x[i]%2 == 1 checks for odd elements but does not include negative odd numbers. To repair the code, we can change the if statement to check if for odd (x[i] % 2 != 0) or positive (x[i] > 0).
>
> ```
> 1  public static int oddOrPos (int[] x) {
> 2     ...
> 3     if (x[i]%2 != 0 || x[i] > 0)
> 4     ...
> 5  }
> ```

C. Answer Correct, Explanation Incorrect (AC-EIC) Example

In Section III-C4, we showed an example of AIC-EPC. In this section, we present an opposite case – an example where the answer is correct, but the explanation is incorrect. This example is another sub-question of the `oddOrPos` code presented in Section V-B.

> **(d)** If possible, give a test case that results in an error, but not a failure. If not, briefly explain why not. Hint: Don't forget about the program counter.

> **ChatGPT's response:** It is not possible to give a test case that results in an error, but not a failure because when the code throws a `NullPointerException`, it results in an error and the program will stop execution. An error state refers to the unexpected behavior of the program. The program counter will not continue execution past the point where the exception is thrown.

According to the textbook, every input that results in an error also results in a failure, because the error states can not be repaired. The answer from ChatGPT is correct because it is impossible to give a test case that results in an error but not a failure. However, the explanation is incorrect as ChatGPT treats the `NullPointerException` as the reason.

VI. THREATS TO VALIDITY

One threat to the validity of our study is the dataset we used. The exercises may not address all the domains required for a software testing class. To address this limitation, we selected a book [12] that is widely used and included all questions with student solutions [15] from the first five chapters.

As ChatGPT was in research preview at the time of this study and is continuously updated, its behavior may differ in future iterations. We identified a few improvements to its performance in the course of our study. In earlier releases, ChatGPT was unable to provide a numeric confidence level, while it is now able to specify a level between 0.0 to 1.0. Responses from ChatGPT are also inconsistent, where repeated invocations of ChatGPT with the same question yield different responses. To reduce the effect of inconsistency on our study, we ran each question three times for generalizability.

Finally, our main results made limited use of prompt engineering (i.e., only varying question context), where differently designed prompts might yield more correct answers. Except for our case study (Section V), our results are based on responses obtained by asking the questions directly as they appear in the book. Book practice questions are designed to focus on human readers and are usually based on the contents of a corresponding chapter. As we have not provided ChatGPT with the actual contents of the chapters, it is possible that ChatGPT might be correct more often if additional context is provided with the questions.

VII. RELATED WORK

Several systems have been proposed to apply large language models (LLMs) to the problem of generating code snippets from natural language requests from developers [17]. Most prominently, GitHub CoPilot, based on Codex, popularized the use of LLMs for real-world programming tasks [18].

Studies have begun to examine how effectively these systems may be used for code generation tasks. AlphaCode was found to generate code that was often similar to human-generated code [6] and achieved a simulated average ranking in the top 54% on Codeforces [17], a programming competition platform. One study found that, on tasks to fix security defects after the defect has already been localized and where additional information is provided through the prompt, LLMs can successfully generate fixes [19].

Some work has specifically examined the potential use of LLM code generation by students in computer science courses. One study found that Codex already performs better than most students on the code writing questions found in typical introductory programming exams [20] as well as more advanced exams on data structures and algorithms [21]. Identical prompts frequently lead to widely varying algorithm choices and code size [20]. A study examining CoPilot's performance on programming assignments from introductory courses found that it achieved scores of 68% to 95% [22].

However, there are substantial challenges with the usability of these systems that may limit their effectiveness for real-world programming tasks. One study found that, despite participants themselves enjoying interacting with the code generation system, there was no measurable productivity benefit in either speed or correctness of programming tasks [23]. Similarly, a second user study found that the productivity benefits of CoPilot were mixed: while it sometimes could make developers faster, it could also lead developers down time-consuming rabbit holes debugging incorrect code [24]. As a result, it had no significant impact on the correctness or task time. Including explanations may help reduce overreliance on potentially incorrect answers, but only in situations when the benefits of engaging with explanations outweigh the costs [25].

As ChatGPT was only recently released, there are only a few studies that have specifically examined the effectiveness of ChatGPT on various tasks. Investigations have found that

ChatGPT produces responses that are at or near the passing threshold for all three parts of the US Medical Licensing Exam without any additional information or prompt engineering [26]. ChatGPT has also been able to achieve a low but passing average grade of C+ in four law school classes [27].

VIII. Discussion: Promises & Perils

In this paper, we examined the potential applicability of ChatGPT to a popular software testing curriculum. We found that ChatGPT is able to provide correct or partially correct answers to 55.6% of questions. Moreover, ChatGPT is a poor judge of its own correctness: its confidence has little bearing on the correctness of its response. Said differently, at least when this study was conducted, ChatGPT's answers will, more likely than not, be incorrect for questions related to software testing courses. That being said, our findings still raise immediate concerns on how the use of ChatGPT might be detected to ensure that questions are meaningfully assessing students' understanding of course materials.

While concerns over student's use of ChatGPT to circumvent assessments represent one potential *peril*, there are also several *promising* directions for integrating ChatGPT into the classroom. We found that ChatGPT is able to provide correct or partially correct explanations to 53.0% of questions. Furthermore, we found that using certain prompting strategies, which provide additional question context, can improve the chances of correct answers and explanations. This finding suggests that for carefully designed in-class activities or labs, ChatGPT, rather than an instructor or TA, can be used to guide students through a set of exercises to improve students' understanding of the material.

Furthermore, we found that certain contexts make it difficult for ChatGPT to answer correctly, and such settings could be used to prevent cheating, especially when access to the internet is necessary. Our dataset contains coding and conceptual questions, and some questions that are both. In our experiments, ChatGPT performed worst with questions involving both code and concepts. It outputs correct answers and explanations most often with coding questions (83.3%), then with conceptual questions (55.6%), and finally with combined questions (31.3%).

Our results are in contrast to the results of applying Chat-GPT in other domains, such as in medicine [26] or law [27] where ChatGPT is shown to pass certain parts of their exams. This difference may be due to the fact that there may be far more content available with which ChatGPT may be trained for these exams, or perhaps due to differences in the nature of the questions themselves. As ChatGPT has only recently been released, a full picture of its capabilities and the impact of such tools is still yet to be determined.

Acknowledgement

References

[1] A. Hindle, E. T. Barr, M. Gabel, Z. Su, and P. Devanbu, "On the Naturalness of Software," *CACM*, 2016.

[2] M. White, C. Vendome, M. Linares-Vásquez, and D. Poshyvanyk, "Toward Deep Learning Software Repositories," in *MSR*, 2015.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is All You Need," *NeurIPS*, 2017.

[4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, and A. Askell, "Language Models are Few-Shot Learners," *NeurIPS*, 2020.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding," *arXiv*, 2018.

[6] S. Lertbanjongngam, B. Chinthanet, T. Ishio, R. G. Kula, P. Leelaprute, B. Manaskasemsak, A. Rungsawang, and K. Matsumoto, "An Empirical Evaluation of Competitive Programming AI: A Case Study of Alpha-Code," in *IWSC*, 2022.

[7] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, and D. Jiang, "CodeBERT: A Pre-Trained Model for Programming and Natural Languages," *arXiv*, 2020.

[8] M. Allamanis, M. Brockschmidt, and M. Khademi, "Learning to Represent Programs with Graphs," in *ICLR*, 2018.

[9] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, and G. Brockman, "Evaluating Large Language Models Trained on Code," *arXiv*, 2021.

[10] OpenAI, "ChatGPT." https://openai.com/blog/chatgpt, 2023.

[11] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, and A. Ray, "Training Language Models to Follow Instructions with Human Feedback," *arXiv*, 2022.

[12] P. Ammann and J. Offutt, *Introduction to Software Testing*. Cambridge University Press, 2016.

[13] "ChatGPT Software Testing Study: GitHub Repository." https://github.com/sajedjalil/ChatGPT-Software-Testing-Study, 2023.

[14] "ChatGPT Software Testing Study: Zenodo Record." https://doi.org/10.5281/zenodo.7700501, 2023.

[15] P. Ammann and J. Offutt, "Introduction to Software Testing Book Solution." https://cs.gmu.edu/~offutt/softwaretest/exer-student.pdf, 2023.

[16] "OpenAI." https://openai.com, 2023.

[17] Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, and A. Dal Lago, "Competition-Level Code Generation with AlphaCode," *Science*, 2022.

[18] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, and G. Brockman, "Evaluating Large Language Models Trained on Code," *arXiv*, 2021.

[19] H. Pearce, B. Tan, B. Ahmad, R. Karri, and B. Dolan-Gavitt, "Examining Zero-Shot Vulnerability Repair with Large Language Models," in *S&P*, 2023.

[20] J. Finnie-Ansley, P. Denny, B. A. Becker, A. Luxton-Reilly, and J. Prather, "The Robots are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming," in *ACE*, 2022.

[21] J. Finnie-Ansley, P. Denny, A. Luxton-Reilly, E. A. Santos, J. Prather, and B. A. Becker, "My AI Wants to Know if This Will be on the Exam: Testing OpenAI's Codex on CS2 Programming Exercises," in *ACE*, 2023.

[22] B. Puryear and G. Sprint, "Github CoPilot in the Classroom: Learning to Code with AI Assistance," *CCSC*, 2022.

[23] F. F. Xu, B. Vasilescu, and G. Neubig, "In-IDE Code Generation from Natural Language: Promise and Challenges," *TOSEM*, 2022.

[24] P. Vaithilingam, T. Zhang, and E. L. Glassman, "Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models," in *CHI*, 2022.

[25] H. Vasconcelos, M. Jörke, M. Grunde-McLaughlin, T. Gerstenberg, M. Bernstein, and R. Krishna, "Explanations Can Reduce Overreliance on AI Systems During Decision-Making," *arXiv*, 2022.

[26] T. H. Kung, M. Cheatham, A. Medinilla, ChatGPT, C. Sillos, L. De Leon, C. Elepano, M. Madriaga, R. Aggabao, and G. Diaz-Candido, "Performance of ChatGPT on USMLE: Potential for AI-Assisted Medical Education Using Large Language Models," *medRxiv*, 2022.

[27] J. H. Choi, K. E. Hickman, A. Monahan, and D. B. Schwarcz, "ChatGPT Goes to Law School," *SSRN*, 2023.