An Evaluation of ChatGPT's Ability on Code Generation

Shiyu Xiu
Department of ECE
University of Toronto
Toronto, Ontario, Canada
shiyu.xiu@mail.utoronto.ca

Yuangan Zou
Department of ECE
University of Toronto
Toronto, Ontario, Canada
bill.zou@mail.utoronto.ca

Qian Tang
Department of ECE
University of Toronto
Toronto, Ontario, Canada
qianqian.tang@mail.utoronto.ca

ABSTRACT

ChatGPT is a language model created by OpenAI and trained on a vast amount of text data, allowing it to generate human-like responses to a wide range of topics and questions. [1] Leetcode is an online platform that provides a collection of coding challenges and practices for software developers to improve their coding skills.[2] For our study, we aim to evaluate the ability of ChatGPT in terms of code generation, with a focus on generating solutions for Leetcode questions. This study is motivated by the desire to explore the possibility of applying the state-of-the-art AI tool ChatGPT as an assistant for software developers when they are writing code. In order to achieve this goal, we need to evaluate ChatGPT's ability in terms of generating correct, efficient, concise as well as easy-to-understand code when given a prompt. More specifically, we focus on three main programming languages that are widely used in the industry: Python, Java, and C#. By conducting this study, we hope that we obtain a clear understanding of ChatGPT's advantages and disadvantages in code generation so that it provides some guidelines for developers to apply the assistance of ChatGPT when writing code. Also, observing situations where ChatGPT does not perform well, it provides some insights into the potential ways of improvement for ChatGPT to be trained as a better code writer. For the evaluation, we will do a quantitative analysis for the correctness and resource consumed by running test cases from the Leetcode website, a quantitative analysis for cyclomatic and cognitive complexity using open-source software SonarQube as well as a survey-like quantitative analysis for the readability of the generated code solutions.

KEYWORDS

ChatGPT, Code Generation, Program Synthesis, Empirical Evaluation, Cyclomatic, and Cognitive complexity

1 INTRODUCTION

In recent years, with the fast development of Machine Learning and its wide usage, natural language processing (NLP) has been increasingly applied to improve the efficiency of software development as well as automating some of the development tasks, just as code generation as well as code debugging. Large language models like ChatGPT have shown remarkable capabilities and potential in generating not only human-understandable text but also programming code for software development. In our study, we want to evaluate ChatGPT's ability on generating code solutions based on sets of Leetcode questions, which are well-structured programming challenges offered in the well-known online platform Leetcode. The programming languages studied are Python, Java, and C#. In order to evaluate ChatGPT 's coding ability from various aspects, we design the evaluation criteria to include correctness rate, running time and memory consumption, Cyclomatic and Cognitive complexities [5,6] as well as readability.

The research questions we aim to address in this study are:

1. How does ChatGPT perform in terms of generating correct code solutions to Leetcode questions with different languages?
2. How does ChatGPT perform in terms of running time and memory consumed when solving Leetcode questions with different languages, compared with human-written solutions?
3. How understandable is the code generated by ChatGPT when solving Leetcode questions in Python?

The first RQ focuses on the correctness of the generated code. In order for software developers to be willing to adopt ChatGPT as a code-writing assistant, it is important that the code generated by ChatGPT is functional correct as well as bug-free. Thus, it is crucial to evaluate ChatGPT's ability to generate correct code solutions.

The second RQ focuses on the efficiency of the generated code. We want to evaluate ChatGPT's ability not only to generate functionally correct code but also high-performance code by observing the amount of time and memory consumed by running the code. We believe that it is important to include this evaluation criteria since efficient code can help reduce costs and resource usage, which should be part of a software developer's job. Making a comparison between ChatGPT-generated code and human-written code provides a good overview of how good ChatGPT is compared with human developers.

The last RQ focuses on the understandability of the generated code. Since software development is a highly cooperative process, the understandability of code is important for other developers to read and modify each other's code. Thus, we decided to include

understandability in our evaluation criteria. We are aware that different programming languages apply different algorithms and have different syntaxes, therefore, comparing the understandability across different programming languages is unnecessary. As a result, we only consider Python in this case.

By doing this study, we aim to contribute to the overall understanding of the capabilities of ChatGPT in terms of writing code and providing insights for ChatGPT to be improved as a better code writer.

Our research methodology includes an experiment on observing the correctness rate of the generated code solutions across three different programming languages, evaluating running time and space consumed by observing the feedback from Leetcode website submission results, evaluating cyclomatic and cognitive complexities, as well as a quantitative survey-like study for the readability of the generated code. It has been shown in the studies that the understandability of the code is directly related to cyclomatic and cognitive complexities [5,6].

Our results indicate that ChatGPT performs better in writing Python than Java and C# in terms of correctness rate. In terms of running time and memory consumed, ChatGPT's performance varies across different problems and languages. When compared with human-written solutions, solutions generated by ChatGPT have larger complexity scores in most cases, indicating that they are more complex and harder to understand. The readability for the generated code solutions is quite excellent according to the result of our readability analysis.

2 BACKGROUND AND LITERATURE REVIEW

Since ChatGPT is a recent technology, there is not much work around it. Also, we notice that there are few studies on generating code using ChatGPT either. Therefore, for the related work, we collect some works around evaluating AI code generation tools besides ChatGPT such as the GitHub Copilot, which is also a popular AI tool for generating coding responses similar to ChatGPT. Nhan[7] evaluates the correctness and understandability of GitHub Copilot's suggested code. The author took a similar approach as ours by using 33 Leetcode questions in four different programming languages and evaluating the correctness of the generated code response from GitHub Copilot. The study also evaluates the cyclomatic and cognitive complexities of the code generated. Finally, a comparison across four programming languages is conducted, arriving at the conclusion that Java has the highest correctness rate (57%) while JavaScript has the lowest correctness rate (27%). Also, there are no notable differences in terms of cyclomatic and cognitive complexities between the programming languages. This study is similar to our own study in structure and design, but it focuses on Copilot instead of ChatGPT. Also, this study does not evaluate the readability of the generated code. Since code readability is critical in terms of collaboration, maintenance as well as debugging, we aim to fill this research gap in our own study.

There is also existing work for evaluating ChatGPT's ability in finding bugs in code. Nigar[8] try to explore the use of ChatGPT in tackling programming bugs and examines the characteristics of ChatGPT and how these characteristics can be leveraged for debugging purpose. They looked at several fields including bug prediction and bug explaining to better understand the capacity of ChatGPT's debugging usage. This study provides some insights for our own study since it explores how ChatGPT understands code and possibly its ability to generate new code in order to fix the detected bugs. However, since debugging focuses more on modifying a part of a given piece of code, it is not a good evaluation of ChatGPT's ability on generating complete code modules to solve a well-defined problem. Therefore, our study aims to fill this gap by offering ChatGPT with Leetcode questions instead of pre-written code, asking it to generate a complete piece of code to solve the Leetcode question. We believe that it is a better approach for evaluating the code generation ability of ChatGPT.

Some studies evaluate large language models trained specifically for code generation, such as Codex. Codex is a GPT language model fine-tuned on publicly available code from GitHub, and it is specifically designed for code-writing tasks. Mark[9] performed an evaluation of Codex's ability in code generation specifically for Python by releasing a new evaluation set that measures the functional correctness for synthesizing programs from docstrings. The study also explores ways of improvement for Codex, with repeated sampling as a way of producing working solutions to difficult prompts. This study provides a good overview of Codex's ability to generate correct solutions to different prompts, but it lacks an analysis of how efficient the generated code is as well as how readable it is. Our study is able to fill this gap by performing a running time and space consumption analysis, as well as a readability analysis. Additionally, our study is not limited to Python but also takes other two popular programming languages Java and C# into consideration.

It is also important to introduce some of the terminologies we used in our work.

Leetcode: Leetcode is a website that provides a platform for practicing coding skills and preparing for technical interviews. It has a collection of coding problems that can be solved in several programming languages such as Python, Java, C#, and more.

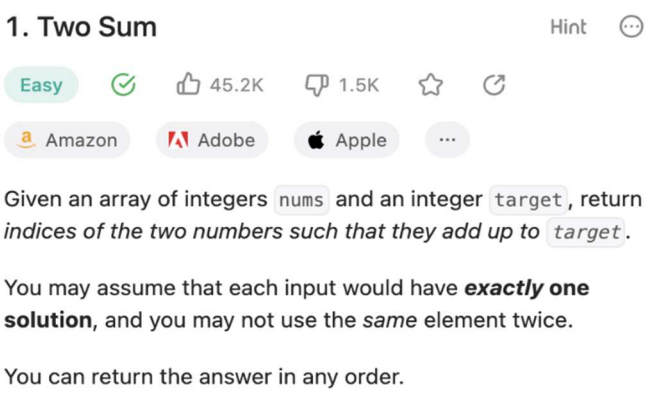

Figure 1: Leetcode example question description

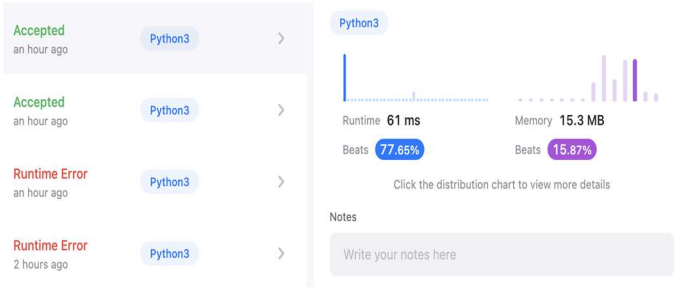

Figure 2: The submission results from the Leetcode website

Figure 2 shows whether the submitted solution is accepted by the Leetcode website. A solution is considered accepted if and only if the solution passes all the test cases provided by Leetcode. There are also other statuses after submission:

1. *Wrong Answer*: The code has failed some test cases due to some mismatches in program output but there are no errors in the code during compilation.
2. *Compile Errors:* The code has some errors and failed during the compilation.
3. *Time Limit Exceeded:* There are no errors in the code but There are some test cases' execution times that exceed the expected time constraint.
4. *Runtime Error:* There are failed test cases because errors happened during the execution.

ChatGPT:

ChatGPT is a large language model created by OpenAI. It is trained with massive amounts of text data including website texts, books, research papers, code as well as user-written content such as Twitter and Facebook posts. It is able to provide users with accurate and relevant responses to their queries.

Fortunately, the OpenAI team has recently published its ChatGPT API endpoint for the general public and enabling us to use ChatGPT for our work in a programmatic way easily.

Cyclomatic and Cognitive Complexities using SonarQube:

SonarQube is open-source software that evaluates the quality of the code, they provide the function to measure Cyclomatic and Cognitive Complexities. These two complexities are concepts in software engineering that help evaluate the complexity and understandability of code.

Cyclomatic complexity quantitative measurement of the number of unique paths through a program's flow and identifies the number of possible decision points and loops within the program. Whenever the control flow of a function splits, the complexity counter is incremented by one. Each function has a minimum complexity of 1.

Cognitive complexity is a quantitative measurement of the mental effort required to understand a program. The Cognitive Complexity score is calculated based on various factors such as nesting depth, code branching, and code duplication. SonarQube adds one for each break in the linear flow of the code or when flow-breaking structures are nested. It also Ignores shorthand constructs that help with readability.

3 METHODOLOGY

3.1 Data Collection

In this section, we will explain how we obtain our data for our research purposes, we utilized data mining techniques to facilitate our data processing process.

First, we use a customized Python script to automatically crawl random 50 Leetcode questions and randomly chose 32 questions from those 50 questions for evaluation. Their associated metadata such as the function body, the question descriptions, and the comments from Leetcode.com are also crawled. This metadata information will help ChatGPT to have a better context and produce more accurate responses potentially.

Second, we use ChatGPT's official API and query the API by using our customized prompts and the Leetcode questions. The prompt we

used for querying the ChatGPT is "*Using {programming language} language, provide only code do not include any explanation for the following prompt: {problem_content}, begin your answer with {header} and do not change it keep the structure. Don't change the basic structure of the header. Avoid runtime error and compile error*", We then saved the generated code solutions using different programming languages for later evaluation.

Next, the code solutions generated in the previous step are submitted on the Leetcode website using Python scripts and run against sets of test cases provided by the Leetcode web. The submission results including error types, the running time, the memory consumed, and the percentage of beats are saved for analysis purposes.

Finally, we evaluate the quality of all the generated code solutions from different metrics including correctness rate, time and space consumption, cyclomatic and cognitive complexity, and readability. For cyclomatic and cognitive Complexity, we will use the SonarQube software to evaluate these two complexity matrices. For readability scores, we will conduct a within-group survey-like study and assign readability scores manually using our customized scoring system, which will be discussed in section 3.2.

3.2 Data Analysis

The collected data is analyzed based on our evaluation criteria: correctness, running time and memory consumed, cyclomatic and cognitive complexities as well as readability.

We perform a quantitative analysis of the correctness of the generated code solutions by examining if there are any failed test cases for a specific question. A generated solution is considered correct only if it does not fail any of the provided test cases. We count the number of correctly and incorrectly generated code solutions for each programming language across all Leetcode questions we have collected and calculate the correctness rates.

In addition, we are going to conduct an experiment with three statistical hypothesis tests on all Leetcode questions we obtained in order to study the difference in correctness rate across different programming languages. The experiment setup is as follows:

- **Null Hypothesis** :
    - When using ChatGPT to generate solutions for Leetcode questions, there is no relationship between the correctness rate of Python and Java language.
    - When using ChatGPT to generate solutions for Leetcode questions, there is no relationship between the correctness rate of Python and C# language.
    - When using ChatGPT to generate solutions for Leetcode questions, there is no relationship between the correctness rate of Java and C# language.
- **Alternative Hypothesis** :
    - When using ChatGPT to generate solutions for Leetcode questions, there is a relationship between the correctness rate of Python and Java language.
    - When using ChatGPT to generate solutions for Leetcode questions, there is a relationship between the correctness rate of Python and C# language.
    - When using ChatGPT to generate solutions for Leetcode questions, there is a relationship between the correctness rate of Java and C# language.
- **Experiment Groups (EG) and Control Groups (CG):** There are three sets of experiment groups and control groups.
    - EG1: the set of Leetcode questions we pass to ChatGPT for which they must be answered in Python.

      CG1: the set of Leetcode questions we pass to ChatGPT for which they must be answered in Java.
    - EG2: the set of Leetcode questions we pass to ChatGPT for which they must be answered in Python.

      CG2: the set of Leetcode questions we pass to ChatGPT for which they must be answered in C#.
    - EG3: the set of Leetcode questions we pass to ChatGPT for which they must be answered in Java.

      CG3: the set of Leetcode questions we pass to ChatGPT for which they must be answered in C#.
- **Experiment Procedure:** For each of the 50 generated solutions, we assigned a label (1 if correct, 0 if incorrect) to it, and then we utilized the Python Scipy library to perform the paired t-test on those labels.
- **Data analysis:** After the experiment, we obtained the p-value. If the p-value is less than 0.05, we reject the null hypothesis and conclude that there is a relationship between the programming language applied and the correctness rate. Otherwise, accept the alternative hypothesis.

The analysis of running time and memory consumed is relatively simple. As shown in Figure 2, we obtain the running time as well as

the amount of consumed memory directly from the response of the Leetcode website after we submit the generated code solution. We then do a comparison between ChatGPT's answer with human answers to Leetcode questions.

We obtain and compare the Cyclomatic and Cognitive complexities scores by submitting ChatGPT's solution and human-written answers for the Leetcode question in Python to the SonarQube software. The human answers are chosen mostly from the official editorial answers and from the discussion section if there are no official answers. There are a few reasons we chose to evaluate Python solutions only. First, SonarQube has the best support for standalone files in Python language. In addition, Python is one of the most popular programming languages used nowadays, therefore, the evaluation results are relevant to a large number of professional developers. Moreover, we believed that the evaluation of code understandability in Python only is sufficient to demonstrate ChatGPT's ability in generating understandable code.

For the readability analysis, we define a scoring system to follow when we assign a score. The scoring system we developed is:

1. Code has consistent formatting: 1 point.
2. Naming conventions are consistent and follow a logical pattern: 1 point.
3. Code is easy to navigate due to consistent structure: 1 point.
4. Code is efficient and avoids unnecessary repetition: 1 point.

We then perform a within-group survey-like study by manually assigning readability scores for each of the generated code solutions in Python. The code that does not meet this standard will be assigned 0 in that category, assigned 1 if meet the standard. The total score of each response ranges from 0 to 4. All members of our team will be assigning scores for each code solution generated by ChatGPT. Finally, an inter-rater agreement analysis is conducted by calculating Fleiss's kappa [10] to check if the team has arrived at a general agreement on the readability of the generated code.

4 EMPIRICAL RESULTS

4.1 RQ1: How does the ChatGPT perform in terms of generating correct code solutions to Leetcode questions with different languages?

Difficulty	Problem #	Accpetance	Test Cases Passed		
			Python	Java	C#
Easy	9	53.60%	100%	100%	100%
	13	58.60%	100%	100%	0%(Compile Errors)
	14	40.90%	100%	100%	100%
	26	51.60%	100%	100%	100%
	27	53.10%	100%	100%	100%
	28	39.10%	100%	100%	100%
	35	43.50%	100%	100%	100%
Medium	2	40.40%	100%	100%	100%
	3	33.80%	100%	100%	100%
	5	32.40%	100%	100%	100%
	11	54.00%	100%	100%	100%
	12	62.10%	100%	100%	100%
	15	32.60%	100%	0%(Compile Errors)	100%
	18	35.90%	100%	0%(Compile Errors)	0%(Compile Errors)
	22	72.60%	100%	100%	100%
	24	61.40%	100%	100%	100%
	31	37.60%	100%	100%	100%
	33	39.00%	100%	100%	100%
	34	41.90%	100%	100%	100%
	36	58.10%	100%	0%(Compile Errors)	0%(Compile Errors)
	39	68.70%	100%	100%	100%
	40	53.40%	100%	100%	100%
	46	75.70%	100%	100%	100%
	47	57.40%	100%	100%	100%
	48	71.10%	100%	100%	100%
Hard	10	28.0%	352/354(99%)(Wrong Answer)	100%	100%
	25	54.70%	100%	100%	100%
	30	31.20%	100%	0%(Compile Errors)	0%(Compile Errors)
	32	32.80%	100%	100%	100%
	37	57.80%	0%(Time limit Exceeded)	0%(Compile Errors)	0%
	41	36.70%	122/175(70%)(Runtime Errors)	136/175(78%)(Wrong Answer)	0%(Compile Errors)
	44	26.90%	100%	0%(Compile Errors)	100%
Accepted			29(91%)	25(78%)	27(84%)
Wrong Answer			1(3%)	1(3%)	0(0%)
Time limit Exceeded			1(3%)	0(0%)	0(0%)
Compile Errors			0(0%)	6(19%)	5(16%)
Runtime Errors			1(3%)	0(0%)	0(0%)

The label in the figure: "Acceptance" means the percentage of solutions from all users that are accepted by Leetcode. "Test Cases Passed" means how many percentages of test cases passed. 100% value means all passed. 352/354 means in a total of 354 test cases the solution passed 352 test cases. The question which contains errors in which type is also identified in the figure. The "Accepted" for Python is the correctness rate of Python solutions.

The result of the experiment is as follows:

```
the paird t-test p-value for Python and Java is  0.3683458324341894
the paird t-test p-value for Python and C# is  0.03412159030060466
the paird t-test p-value for Java and C# is  0.25019902613551404
```

From the three sets of paired t-tests, the p-value for the experiment group in Python and the control group in Java is 0.3683. It is larger than 0.05, indicating that we accept the null hypothesis and conclude that when using ChatGPT to generate solutions for Leetcode questions, there is no relationship between the correctness rate of Python and Java language. Similarly, the p-value for the experiment group in Python and the control group in C# is 0.0341. It is smaller than 0.05, so we reject the null hypothesis and accept the alternative hypothesis, leading to the conclusion that there is a

relationship between the correctness rate of Python and C# language. Similarly, the p-value for the experiment group in Python and the control group in C# is 0.2502. It is larger than 0.05, therefore we accept the null hypothesis, concluding that there is no relationship between the correctness rate of Java and C# language.

To sum up, our experiment indicates that in terms of correctness rate, there is no relationship between using Python and Java, no relationship between using Java and C# and there is a relationship between using Python and C#.

However, due to not having much data for the experiment, a small sample size reduces the power of the test, making it harder to detect a significant difference between the paired samples, even if such a difference exists. In such cases, the results of the test may be inconclusive or unreliable. In future studies, we will increase the dataset to obtain a more convincing result from the t-test.

Based on the data and experiment results shown above, we can tell that ChatGPT demonstrates a strong ability to generate code solutions in Python, Java, and C#, with high test case pass rates across all three languages. However, its performance in Java and C# is lower compared to Python, with a higher rate of compile errors. The model can handle problems of varying difficulty levels but struggles with some hard problems, leading to lower pass rates and issues like wrong answers, time limit exceeded, and runtime errors. Overall, in terms of correctness rate, ChatGPT excels in generating Python solutions (91%), while showing varying performance in Java (78%) and C# (84%). There is room for improvement in addressing more difficult problems and achieving consistent performance across languages.

4.2 RQ2: How does ChatGPT perform in terms of running time and memory consumed when solving Leetcode questions with different languages, compared with human-written solutions?

Problem Numbers	Difficulty	Time(ms)			Time Beat(%)		
		Python	Java	C#	Python	Java	C#
2	Medium	69	2	2	60.13	98.44	98.44
3	Medium	53	2	2	92.1	99.97	99.97
5	Medium	506	19	197	84.22	84.11	28.32
9	Easy	77	9	9	17.19	99.11	99.11
10	Hard	N/A	2	3	N/A	90.62	63.82
11	Medium	757	4	4	64.98	74.38	74.38
12	Medium	44	4	15	86.86	100	19.37
13	Easy	49	4	N/A	61.72	94.12	N/A
14	Easy	32	0	0	81.6	100	100
15	Medium	1405	N/A	37	60.28	N/A	83.1
18	Medium	963	N/A	N/A	47.54	N/A	N/A
22	Medium	42	1	1	24.21	92.97	92.97
24	Medium	28	0	0	88.38	100	100
25	Hard	40	0	0	97.47	100	100
26	Easy	84	1	1	82.98	99.97	99.97
27	Easy	31	0	0	84.24	100	100
28	Easy	23	0	0	97.44	100	100
30	Hard	378	N/A	N/A	63.78	N/A	N/A
31	Medium	46	0	0	42.66	100	100
32	Hard	38	5	1	91.3	66.18	100
33	Medium	42	0	0	69.97	100	100
34	Medium	75	0	0	98.58	100	100
35	Easy	41	0	0	98.5	100	100
36	Medium	95	N/A	N/A	71.95	N/A	N/A
37	Hard	N/A	N/A	N/A	N/A	N/A	N/A
39	Medium	50	2	2	91.5	82.26	82.26
40	Medium	47	4	4	87.46	54.29	54.29
41	Hard	N/A	N/A	1	N/A	N/A	99.97
44	Hard	787	N/A	39	41.74	N/A	35.7
46	Medium	48	2	2	18.64	38.42	38.42
47	Medium	54	1	2	92.4	99.89	89.59
48	Medium	39	0	0	45.77	100	100

The label in the figure: "Times (ms)" means how long it takes to pass all the provided test cases. "Time Beat (%)" means the percentage of Leetcode users' solutions that ChatGPT's solution can outperform. N/A value means answers are not accepted by Leetcode.

From the above figure, we can tell that ChatGPT exhibits varying performance across languages. The average time-beat percentage for Python is 70.54%, while for Java and C#, the averages are 90.99% and 83.70%, respectively. These values indicate that ChatGPT's performance in running time is more consistent in Java and C#, but still demonstrates a competitive edge in Python when compared to humans.

Problem Number	Difficulty	Memory(MB)			Memory Beat(%)		
		Python	Java	C#	Python	Java	C#
2	Medium	13.8	42.9	42.1	74.55	29.6	97.91
3	Medium	14.1	42.4	42.5	39.13	89.12	83.2
5	Medium	13.9	42.4	45.9	53.43	78.77	14.1
9	Easy	13.8	42	42.3	92.79	63.68	29.45
10	Hard	N/A	41.2	41.4	N/A	67.95	51.13
11	Medium	27.4	52.6	52.3	44.22	58.46	82.16
12	Medium	13.8	42	43.2	97.76	85.36	17.96
13	Easy	13.9	42.3	N/A	61.1	95.7	N/A
14	Easy	13.9	40.8	40.2	77.45	41.41	86.87
15	Medium	18.5	N/A	49.3	38.16	N/A	89.15
18	Medium	13.9	N/A	N/A	50.69	N/A	N/A
22	Medium	14.2	42.7	42.1	27.49	51.95	94.69
24	Medium	13.8	40.4	40.9	52.55	38.26	10.81
25	Hard	15.1	42.8	43.3	97.94	36.48	8.75
26	Easy	15.5	43.7	44.3	94.11	88.14	35.67
27	Easy	14	41	41.1	6.88	59.44	59.44
28	Easy	13.8	40.7	40.2	50.46	44.21	87.4
30	Hard	14.1	N/A	N/A	67.1	N/A	N/A
31	Medium	13.9	42.7	43.1	58.51	66.16	20.92
32	Hard	14.7	42.7	41.9	19.87	38.91	97.41
33	Medium	14.2	42.4	41.4	44.9	35.55	99.41
34	Medium	15.4	46.1	45.5	85.42	28.65	81.6
35	Easy	14.8	42.2	42.2	28.6	36.92	27.94
36	Medium	14	N/A	N/A	22.65	N/A	N/A
37	Hard	N/A	N/A	N/A	N/A	N/A	N/A
39	Medium	14	42.9	42.8	21.1	55.87	65.36
40	Medium	14	42.8	43	49.44	54.17	34.5
41	Hard	N/A	N/A	50.4	N/A	N/A	90.91
44	Hard	22.4	N/A	43.1	51.2	N/A	61.48
46	Medium	14	42.5	42.7	49.74	87.94	57.5
47	Medium	14.3	43	43.2	44.2	88.54	69.48
48	Medium	14	42.4	42.3	19.24	63.74	69.16

The label in the figure: "Memory (MB)" means how much memory space it takes to pass all the provided test cases. "Memory Beat (%)" means how many percentage Leetcode users' solutions that ChatGPT's solution can outperform. N/A value means answers are not accepted by Leetcode.

In terms of memory efficiency, the Python solution generated by ChatGPT outperformed a substantial portion of other Leetcode users, with an average memory-beat percentage of 52.44%. For Java and C# solutions, ChatGPT's performance in memory is mixed, with average memory-beat percentages of 59.40% and 60.16%, respectively in some instances, the generated code performs well, outperforming a significant percentage of user's solutions, while in other cases, it does not. This also implies that ChatGPT's performance in memory efficiency might not be as consistent in Java and C#, but still demonstrates a competitive edge in Python. Suggesting that ChatGPT performs well in this regard.

4.3 RQ3: How understandable is ChatGPT's generated code when solving Leetcode questions in Python?

4.3.1 Cyclomatic and Cognitive Complexities Score for Python Solution

Problem file	Cyclomatic complexity		Difficulty
	ChatGPT	Human	
Q2	10	8	Medium
Q3	5	7	Medium
Q5	5	6	Medium
Q9	7	3	Easy
Q10	4	8	Hard
Q11	4	3	Medium
Q12	3	1	Medium
Q13	4	4	Easy
Q14	7	3	Easy
Q15	21	10	Medium
Q18	31	17	Medium
Q22	6	8	Medium
Q24	2	3	Medium
Q25	4	6	Hard
Q26	4	3	Easy
Q27	3	2	Easy
Q28	3	5	Easy
Q30	14	6	Hard
Q31	7	7	Medium
Q32	8	4	Hard
Q33	14	15	Medium
Q34	27	10	Medium
Q35	5	4	Easy
Q36	12	7	Medium
Q39	7	4	Medium
Q40	11	5	Medium
Q41	7	10	Hard
Q44	12	12	Hard
Q46	7	4	Medium
Q47	7	5	Medium
Q48	3	3	Medium

When comparing ChatGPT's generated code with human-written code, we observe differences in Cyclomatic and Cognitive Complexity. For Cyclomatic Complexity, ChatGPT often generates code with higher complexity values than human code, particularly for easy and medium problems.

Problem #	Cognitive complexity		Difficulty
	ChatGPT	Human	
Q2	10	10	Medium
Q3	5	11	Medium
Q5	5	3	Medium
Q9	7	2	Easy
Q10	17	8	Hard
Q11	4	3	Medium
Q12	3	0	Medium
Q13	4	5	Easy
Q14	7	4	Easy
Q15	21	15	Medium
Q18	31	28	Medium
Q22	6	15	Medium
Q24	2	2	Medium
Q25	4	4	Hard
Q26	4	3	Easy
Q27	3	1	Easy
Q28	3	9	Easy
Q30	14	9	Hard
Q31	7	7	Medium
Q32	8	7	Hard
Q33	14	30	Medium
Q34	27	20	Medium
Q35	5	6	Easy
Q36	12	15	Medium
Q39	7	5	Medium
Q40	11	8	Medium
Q41	7	13	Hard
Q44	12	19	Hard
Q46	7	4	Medium
Q47	7	7	Medium
Q48	3	3	Medium

Similarly, for cognitive complexity, ChatGPT's code varies in complexity compared to human-written solutions. In certain instances, ChatGPT produces code with higher cognitive complexity, while in others, it generates simpler code. It was found that code generated by ChatGPT generally exhibits higher complexity values when compared to code written by humans in most cases. The discrepancies in both cyclomatic and cognitive complexity can be attributed to the model's varying ability to produce concise and efficient code solutions.

4.3.2 Readability Survey-like Study for Python Solutions

For evaluating the readability of code generated by ChatGPT, we perform a survey-like study in which every team member offers a readability score for each code solution generated. We feed 32 questions to ChatGPT and obtain the generated solutions. For each solution, each team member assigns a score from 0 to 4 for it based on the criteria we developed. The average score will be taken as a reference of how readable the solution is. Please refer to the following table for details:

A	B	C	D
Problem #	Rater1	Rater2	Rater3
1	2	2	2
2	4	3	3
3	3	3	3
4	3	4	3
5	1	2	2
6	4	3	4
7	4	4	4
8	4	4	4
9	4	4	4
10	3	3	2
11	3	2	2
12	4	4	4
13	4	4	4
14	4	4	4
15	3	3	4
16	4	3	4
17	4	4	4
18	2	3	3
19	4	4	4
20	4	4	4
21	4	4	4
22	4	3	4
23	4	4	4
24	3	3	3
25	N/A	N/A	N/A
26	4	4	4
27	3	4	4
28	4	4	4
29	4	4	3
30	3	4	4
31	4	4	4
32	3	3	2,2,

Based on the assigned scores, the average readability score is calculated to be 3.5/4.0. Therefore, we conclude that in general, the code generated by ChatGPT is easy to understand and well-structured. The format of the code is consistent and concise, the names of variables are meaningful, and, in most cases, there are no duplicate or useless variables.

Finally, we perform an inter-rater agreement analysis by calculating Fleiss's Kappa. Fleiss's Kappa is a statistical measure that quantifies the inter-rater agreement among multiple raters (>=3) and is an extension of Cohen's Kappa which is designed for 2 raters only. The Fleiss's Kappa is calculated to be 0.479 and based on the following table of interpretation from Landis and Koch[10], we see that our team has arrived at a moderate agreement.

κ	Interpretation
< 0	Poor agreement
0.01 – 0.20	Slight agreement
0.21 – 0.40	Fair agreement
0.41 – 0.60	Moderate agreement
0.61 – 0.80	Substantial agreement
0.81 – 1.00	Almost perfect agreement

5 DISCUSSIONS

In this study, we aimed to evaluate ChatGPT's ability to generate code in Python, Java, and C# by analyzing its performance across various metrics, including correctness rate, running time, memory consumed, cyclomatic and cognitive complexities, and readability. Our findings indicate that ChatGPT demonstrates a strong proficiency in generating code solutions, particularly in Python. However, there are areas where it can be improved, such as consistency across different programming languages and tackling more difficult problems.

In terms of running time and memory consumed, ChatGPT's performance is inconsistent across different problems and languages. While the model generally generates memory-efficient Python solutions, its performance in Java and C# is mixed and varies a lot from question to question.

Our analysis of Cyclomatic and Cognitive Complexities revealed that ChatGPT often generates code with higher complexity values than human-written code. This suggests that there is room for improvement in the model's ability to produce concise and efficient code solutions. Moreover, our readability analysis highlighted that while ChatGPT's generated code is generally readable, there is still potential for improvement in terms of consistent formatting, naming conventions, and code structure.

We believe that this study provides a foundational step along the path of applying ChatGPT as an assistant when writing code during software development. The main contribution of our study is a detailed evaluation of ChatGPT's ability in writing code specifically for solving Leetcode questions. We try to fill the research gaps of previous studies by including more-diverse criteria including not only correctness rate but also resource consumption, complexity measurements as well as readability analysis. In this way, we propose a more detailed evaluation of the quality of code generated by ChatGPT. Additionally, we are aware of the difference between read-world development problems and Leetcode questions. Thus, in order for this study to generalize, some future work is needed in order to evaluate ChatGPT in a more realistic, real-world environment.

6 CONCLUSIONS

In conclusion, our study offers a comprehensive evaluation of ChatGPT's code generation ability by testing its performance in terms of generating solutions to Leetcode questions. We conclude that ChatGPT is proficient in generating code solutions, especially in Python, and can achieve a higher correctness rate in Python (97%) than Java (78%) and C# (84%). The running time of the generated solution is also excellent, with Python beating (70%) of the existing solutions, Java beating (90%), and C# beating (83%). The memory consumption of the generated code solutions is around the average level for all three programming languages. The readability of the generated code solutions is relatively good, with an average readability score of 3.5/4.0 based on our proposed scoring system. However, there are still areas where it can be improved such as the consistency across languages and its ability to handle more difficult problems. The running time and memory consumption analysis showed inconsistent performance, suggesting that the model might benefit from more diverse and challenging training data.

As for code understandability, our findings suggest that improvements in code complexity and readability can be made such as including more documentation in code as well as choosing more meaningful index names instead of i or j. Future research can be done by incorporating feedback from developers through interviews and conducting a more detailed readability analysis among a larger population of developers.

7 THREADS TO VALIDITY

a. Internal Validity

There are some threats to internal validity in our study. Due to the nature of ChatGPT being not open source, we can't know whether the training data for the ChatGPT language model has already contained our input data, which is the description and the corresponding solution of the Leetcode questions. Therefore, it would be hard for us to recognize whether ChatGPT is memorizing the code solutions or is actually generating solutions based on the problem description. Future work to address this concern can be to obtain some newly released Leetcode problems for which ChatGPT has not been trained and feed them to ChatGPT to let it generate the solution, or we can design our own problems similar to the ones from Leetcode to test whether ChatGPT is still able to answer them correctly and efficiently. Furthermore, the results on running time might depend on the network speed at the time of running our script since it measures the running time on the Leetcode remote server. If the network is not ideal, the running time might be longer. For the readability analysis, since it is a subjective evaluation, the result might be biased since we only collected feedback from three participants within the academic field. A future study can perform a similar analysis but among a large group of participants in both industry and academia. The experiment we conduct would be influenced by the limitation of our dataset size, specifically a "sampling bias" threat. It would make the t-test lose part of its power. This could be resolved in the future by collecting more data.

b. External Validity

Due to the time constraint and lots of manual processes during data collection, we are able to only evaluate 32 Leetcode questions and this study might have issues with generalizability to a larger number of Leetcode questions covering more topics. Additionally, we notice that there is a huge gap between real-world software development and Leetcode questions since questions from Leetcode are always well-defined and well-structured, while read-world development problems usually involve some design decisions and trade-offs. Therefore, the ability of ChatGPT to solve Leetcode questions may not generalize well to the real-world software development process. Additionally, the compatibility of different code segments is not included in our study since we are evaluating based on individual questions on Leetcode. Since compatibility is important in terms of collaboration between different parts of a large software system, it is also a threat to external validity, making our study challenging to generalize. Future studies can address this concern by designing prompts to simulate a real-world software development environment and let ChatGPT design the structure of code it generates; in this way, we are able to obtain a more realistic understanding of ChatGPT's ability in real-world development.

REFERENCES

[1] Introducing ChatGPT. Introducing ChatGPT. (n.d.). Retrieved March 15, 2023, from https://openai.com/blog/chatgpt

[2] Explore. Leetcode. (n.d.). Retrieved March 15, 2023, from https://Leetcode.com/explore/featured/card/the-Leetcode-beginners-guide/

[3] Ian Editor (Ed.). 2007. *The title of book one* (1st. ed.). The name of the series one, Vol. 9. University of Chicago Press, Chicago. DOI:https://doi.org/10.1007/3-540-09237-4.

[4] David Kosiur. 2001. *Understanding Policy-Based Networking* (2nd. ed.). Wiley, New York, NY..

[5] Dinuka R. Wijendra and K.P. Hewagamage. 2021. Analysis of cognitive complexity with cyclomatic complexity metric of ... (February 2021).

[6] Luigi Lavazza, Abedallah Zaid Abualkishik, Geng Liu, and Sandro Morasca. 2023. An empirical evaluation of the "cognitive complexity" measure as a predictor of code understandability. *Journal of Systems and Software* 197 (2023), 111561. DOI:http://dx.doi.org/10.1016/j.jss.2022.111561

[7] Nhan Nguyen and Sarah Nadi. 2022. An empirical evaluation of GitHub Copilot's code suggestions. Proceedings of the 19th International Conference on Mining Software Repositories (2022). DOI:http://dx.doi.org/10.1145/3524842.3528470

[8] Nigar M. Surameery and Mohammed Y. Shakor. 2023. Use chat GPT to solve programming bugs. International Journal of Information Technology and Computer Engineering, 31 (2023), 17–22. DOI:http://dx.doi.org/10.55529/ijitc.31.17.22

[9] Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. de O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., … Zaremba, W. (2021, July 14). Evaluating large language models trained on code. arXiv.org. Retrieved April 14, 2023, from https://arxiv.org/abs/2107.03374

[10] "Fleiss' Kappa," Wikipedia, 23-Nov-2022. [Online]. Available:https://en.wikipedia.org/wiki/Fleiss%27_kappa#:~:text=Fleiss'%20kappa%20(named%20after%20Joseph,of%20items%20or%20classifying%20items. [Accessed: 15-Apr-2023].