Evaluating the Efficacy of AI in Solving Entry-Level Coding Problems

Jaykumar Patel
patel.jay4802@utexas.edu

Laith Altarabishi
laithaustin@utexas.edu

Nidhi Dubagunta
nidhi.dubagunta@utexas.edu

Abstract

...

1. Introduction and Motivation

...

2. Previous Work

...

3. Methodology

3.1 Model Selection

For this study, we selected three Large Language Models (LLMs) of varying architecture and scale: ChatGPT 4o, Gemini 1.5 Flash, and Llama3 7B. These models were chosen based on their widespread use, diverse range of size (number of parameters), and representation of state-of-the-art capabilities in natural language processing and reasoning. The exact sizes of the models (in terms of the number of parameters) are not publicly disclosed. However, estimates suggest that ChatGPT-4o contains approximately 1 trillion parameters, Llama3 7B has about 7 billion parameters, and Gemini 1.5 Flash falls somewhere in between.

By comparing these models, we can assess performance trends across diverse LLMs, providing insights into their strengths and weaknesses in solving algorithmic problems and adapting to challenges such as mutations and categorization tasks.

3.2 NeetCode and Problem Selection

NeetCode is a widely used resource among students preparing for coding interviews. It organizes 150 LeetCode problems into categories such as Arrays and Hashing, Two Pointer, Trees, Graphs, and Dynamic Programming (DP), with further distinctions based on difficulty (easy, medium, and hard). These problems collectively represent common types of questions asked in technical interviews, making them an ideal dataset for evaluating Large Language Models (LLMs).

For our study, we sampled 32 problems from NeetCode's curated list, ensuring representation across multiple categories and difficulty levels. This sampling strategy enables a balanced evaluation of LLM performance, reflecting the diverse nature of algorithmic problem-solving tasks typically encountered by interviewees. By using NeetCode, we also leverage its existing categorization framework, which serves as a benchmark for testing the LLMs' ability to classify problems accurately.

3.3 Testing Accuracy and Consistency of LLMs

To assess the accuracy and consistency of the LLMs in solving algorithmic problems, we followed these steps:

1. Each of the 32 problems was presented to three LLMs: ChatGPT, Gemini, and Llama.

2. For each problem, the LLM was prompted to generate a solution three times. This allows is to test the consistency of the models.

3. The output solutions tested on the LeetCode platform for correctness.

This approach allows us to determine if certain categories are more challenging by nature and how the models perform as the difficulty level of the problem changes. This provides insights into the LLMs' accuracy and consistency in solving algorithmic challenges that span various topics and difficulty levels.

Note: Along with verifying if a solution passes a diverse set of test cases, LeetCode also evaluates its optimality by enforcing runtime and memory usage constraints. This mirrors real-world interviews, where candidates are typically expected to provide solutions that are both correct and efficient.

3.4 Testing Categorization Ability of LLMs

Beyond solving problems, we evaluated the LLMs' ability to correctly categorize them based on NeetCode's framework. For each of the 32 problems, we followed these steps:

1. The problem statement was provided to the LLM, and it was asked to determine the most appropriate category from the following set: Arrays and Hashing, Two Pointer, Trees, Graphs, and DP.

2. The predicted category was compared to the ground truth from NeetCode.

This approach allows us to determine how accurately the LLMs can infer underlying problem structures and map them to problem categories.

3.5 Testing Accuracy and Consistency on Mutated Problems

Recognizing that algorithmic problems are often rephrased or slightly altered during interviews, we introduced mutation testing to evaluate LLM robustness. On LeetCode, problems consist of the problem statement and test cases that specify various inputs and outputs. Mutations were applied to ten problems, including:

1. **Rewording:** Altering phrasing or replacing variable names.

2. **Test Case Modifications:** Modifying test cases to different valid inputs and outputs or deleting one or more test cases.

The LLMs were tested on the mutated problems using the same accuracy and consistency metrics as in Section 2.2. By comparing performance on original vs. mutated problems, we assessed the models' ability to adapt to variations in problem presentation.

Note: We only perform mutations on ten problems in the medium and hard category. We excluded the easy category from mutation testing because such problems typically involve straightforward logic and limited complexity, making them less representative of real-world interview scenarios where subtle variations in problem statements or test cases can challenge a candidate's problem-solving skills. By focusing on medium and hard problems, we ensure that the mutations provide a meaningful test of the LLMs' ability to adapt to nuanced changes in problem.

3.6 Comparing LLM Size on Performance

To indirectly examine the impact of LLM size on performance, we leveraged the inherent differences in sizes among ChatGPT, Gemini, and Llama. We compared their accuracy, consistency, categorization ability, and robustness to mutations. Since ChatGPT, Gemini, and Llama vary in size, this approach allows us to determine the impact of LLM size on performance.

4. Results

...

5. Conclusion and Future Work

...

References