Evaluating AI Generated Code

1st Giancarlo Ramirez
GiancarloRamirez@my.unt.edu
University of North Texas
USA

2nd Ahmad Edaibat
ahmadedaibat@my.unt.edu
University of North Texas)
USA

3rd Dillon Edington
dillonedington@my.unt.edu
University of North Texas
USA

4th Patrick Adebiaye
patrickadebiaye@my.unt.edu
University of North Texas
USA

I. Introduction

Artificial intelligence refers to the ability of a digital computer to perform the same or similar tasks to those usually done by human beings. AI technology has massively advanced in recent years with technologies like OpenAI. In this study we will be addressing certain characteristics which differentiates AI models to other ones in comparison. Despite the prevalent use of AI, there are little formal studies evaluating their coding prowess. Understanding these differences is essential for developers to choose the most suitable model for their specific needs. Elon Musk has emphasized the importance of computing power and access to data in evaluating AI capabilities, suggesting that a good measure of an AI's strength is its ability to discuss complex technologies such as battery advancements. Our paper will bring many features into light leading to accessing what AI model is best suited for certain prompts/problems. This is vital for both researchers and the public who often rely on AI tools. In addition, we hope to gain insight on how AI weaknesses/strengths that weren't initially obvious were concluded upon. In this study, the role that software testing will serve will be in creating the different prompts that we will feed into our AI along with any deliberate ways in which we do so. Expanding on this, we will be testing across varied outputs across many AI models. In this paper, we will test both whether a problem can be solved by an AI model explicitly and whether it can detect a wrong approach to a problem.

Accessibility in digital environments refers to designing and developing products, services, and content that can be perceived, understood, and operated by a wide range of users, regardless of their physical, intellectual, or cognitive abilities. Despite the widespread accessibility adoption in web applications over the years, studies have shown that most web applications still present many accessibility violations[1].

II. Background

These will include deep learning, neural network, and machine learning models. During this study, software testing will help us gain some perspective on the performance of these models. The following factors we will be exploring access to information, ... Access to information refers to how much data a given AI model can retrieve. This will differ between AI models and lead to different responses to the same prompts across AI models. In addition, access to information can also refer to AI models' interfaces being designed in a way where a user can easily use it effectively. The relevancy of information is also crucial to this aspect of AI models, specifically that information is accurate and is updated continuously. Finally access to information can also refer to how data is used, oftentimes data from an user, ensuring that given prompts with sensitive information are not accessible to other users. Expanding upon more, some AI models focus on information generation where as others might focus on information retrieval. Models trained to be based on information generation might often be outdated if not properly maintained. AI models based on information retrieval can have higher accuracy. In more specificity, some AI models might use deep learning methods like neural networks succeed in classifying large data that might not correlate to each other. These models learn from patterns given data and making these models good for tasks like natural language processing. AI models might answer the same prompt in different ways as ML models use tokens that map to each item in the output. This is important to note when investigating AI models as we will be using a criteria that should not be affected by the variance in outputs to prompts.

[1]https://webaim.org/projects/million/

III. RelatedWork

ChatGPT had one notable difference in that it does not have the ability to read a current projects entire workspace/files [2] unlike GitHub Copilot and Amazon CodeWhisperer. During this study we will also use dummy function names to assess how large language models perform under such prompting [2]. From this study we learned that using tools like ChatGPT require active prompting and proper explanation from the user and this in respect to word prompts or to code itself[2].

IV. Merit

Our research paper tackles the ambiguity that many student that use AI face. In educating anyone, not just computer science students on how AI works, will elevate the prompts that a person gives and hence better results are likely to be generated. In addition, the researching of AI models within this paper, will help anyone who is building their own AI model,projects related to machine learning, and related to deep learning. Despite not having direct knowledge of how the inner

models work, we hope to unveil characteristics that a student constructing smart intelligence can keep in mind. Our paper could also provide a basis for those wanting surface level engagement with AI topics. This material could be refined into AI introduction workshops at a college.

V. Study Design

To best compare skill at generating code among currently utilized LLMs, we composed **[number]** prompts and judged the response according to a rubric (Table I) of our design.

A. Models

We sought to select an array of models that represent current market strength. To this end, we selected some models that represent long-standing giants of the industry as well as up-and-comers that seek to topple said giants. The following four models represent this array:

- OpenAI's ChatGPT o-3
- DeepSeek's DeepSeekR1
- Google's Gemini 2.0 Flash
- Anthropic's Claude 3.7 Sonnet

B. Prompts

To holistically judge each model on its ability to generate code, we chose a range of prompts that we believe represent a full range of tasks that users might present to LLMs. The prompts fall into **[number]** categories, with the full list of prompts available in Figure **??**. LLMs are given one attempt at each prompt with no corrections or regenerations.

C. Prompt List

We developed the following working list of prompts to evaluate the models across different programming tasks and scenarios:

- **Visual Programming:** Generate code for animating a short video with continuous image rendering
- **Algorithm Implementation:** Create a functional bubble sort algorithm
- **Simulation Development:** Build a simple physics simulation
- **Code Comprehension:** Comment and explain existing uncommented code
- **Web Development:** Create specific HTML/CSS components from scratch
- **Debugging:** Identify and fix segmentation faults and out-of-bounds errors
- **Accessibility Enhancement:** Implement usability/accessibility features
- **DOM Modification:** Modify an existing HTML page to add screen reader functionality
- **Data Structures:** Implement linked lists, double linked lists, and black trees in C++
- **Esoteric Languages:** Solve problems using less common programming languages
- **Beginner Assistance:** Format and organize poorly structured code for readability

These prompts were specifically selected to encompass a broad range of programming challenges that represent real-world usage scenarios. By covering diverse aspects of software development, from algorithmic tasks to accessibility features, we aim to thoroughly evaluate each model's capabilities across the spectrum of programming tasks.

D. Rubric

The rubric used for evaluation is shown in Table I.

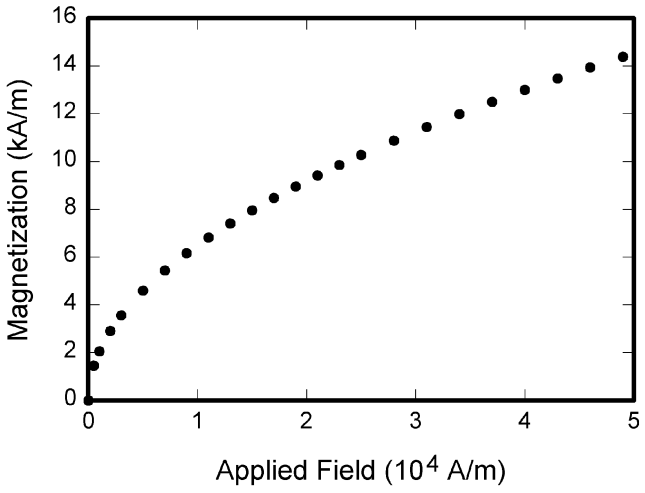

Fig. 1. Example of a figure caption.

Acknowledgment

References

[1] Elon Musk's insights on AI capabilities and the importance of computing power and data access. https://www.reuters.com/technology/artificial-intelligence/elon-musk-says-grok-3-final-stages-outperforming-all-chatbots-2025-02-13/?utm$_source = chatgpt.com$

[2] Yetistiren, Burak et al. "Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT." ArXiv abs/2304.10778 (2023): n. pag.

[3] Wang, Ruotong, et al. "Investigating and designing for trust in AI-powered code generation tools." Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency. 2024.

TABLE I
RUBRIC FOR CODE EVALUATION

Criteria	**Poor (1 point)**	**Middling (3 points)**	**Excellent (5 points)**
Compilation	Doesn't run at all	Modifications needed	Flawless
Optimal Solution	Obtuse solution	Acceptable solution	Optimal solution
Conciseness	$<$50% of lines are essential	$\sim$60% of lines are essential	$>$80% of lines are essential
Self Explanation	Code base is mostly unexplained	Some features of the code are unclear	Code is fully explained
Memory	Core functionality is forgotten	Some variables/functions are redeclared	Original naming conventions are respected, and all variables/functions are recalled

TABLE II
TABLE TYPE STYLES

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		
[a]Sample of a Table footnote.			