LLM-formative: Evaluating Trust in LLMs for Software Developers

MATTHEW CHEN, William & Mary, USA
DIPIN KHATI, William & Mary, USA
DENYS POSHYVANYK, William & Mary, USA
JANICE ZHANG, William & Mary, USA

Usage of LLM applications has risen dramatically over the years, with chatbots like ChatGPT becoming ubiquitous across the industry and academia. However, it is important to determine how reliable and trustworthy their outputs are. In particular, software development relies heavily on correct code, as even a single poorly written line could crash the application or introduce a security vulnerability. We merge the concepts of papers that evaluate trust in LLMs and papers that evaluate software development with LLMs to produce the first paper to evaluate trust in LLMs for software developers. Our contributions are a novel benchmark geared for trust in coding with LLMs, a unique hotkey approach to prompting the LLM chatbots ChatGPT and Microsoft Copilot, and a Google Sites website, LLM-Formative, that effectively conveys the results to users. Our survey shows that LLM-Formative is simple yet well designed and improves people's trust in an LLM's coding abilities on average. Moreover, it shows significant potential for expansion in future works.

1 INTRODUCTION

LLM-powered applications are finding increased use in professional and academic programming projects, making it important to determine how reliable and trustworthy their generated outputs are. An LLM survey by Amperly found that 37.3% of people use AI chatbots like ChatGPT every day and 46% use AI chatbots "a couple times per week" [11]. Moreover, an Accenture study found that 40% of working hours across all industries can be affected by LLMs and 65% of all language tasks could use LLMs for automation or augmentation according to data from the Occupational Information Network from the US Department of Labor [1]. Additionally, these effects are not limited to the industry. Within the realm of research, a study that analyzed over 950 thousand papers found that LLM usage in papers has risen dramatically, showcasing that up to 17.5% of computer science papers were written in part by AI. Evidence of this pivotal shift is that upon the advent of generative AI, research papers showed an increasing trend of distinctively intricate words [12]. Clearly, people are recognizing the benefits of using LLM-powered applications, especially ones with Graphical User Interfaces (GUI), and putting them to use in a variety of disciplines.

However, despite the overall positive impression on AI, 12% of Amperly respondents believe AI chatbots decrease the quality of their work, 31% use ChatGPT alternatives instead, and 35% believe the primary issue with LLMs is reliability of generations [11]. This shows that many people still feel distrust towards AI, which is a valid concern given their tendency for hallucinations. Computer science in particular relies heavily on people creating code that is precisely correct, as even a single poorly written line could crash the application, cause bugs in other parts of the program, or introduce security vulnerabilities. Therefore, it is critically important to inform people about how much they should trust the LLMs they use.

Many studies have used benchmarks to evaluate how trustworthy LLMs are by investigating various metrics associated with trust [8] [14]. In addition, other papers have assessed LLM accuracy for code generation, but they use Natural Language Processing metrics as opposed to incorporating trust-specific metrics [5] [17] [15]. Our innovation is to look at LLM usage within the specific context of software development and display the results of metrics that

Authors' addresses: Matthew Chen, William & Mary, Williamsburg, Virginia, USA, mjchen@wm.edu; Dipin Khati, William & Mary, Williamsburg, Virginia, USA, khati@wm.edu; Denys Poshyvanyk, William & Mary, Williamsburg, Virginia, USA, dposhyvanyk@wm.edu; Janice Zhang, William & Mary, Williamsburg, Virginia, USA, yzhang104@wm.edu.

are tuned to combine trust and programming. Other papers have shown that computer scientists care deeply about community experiences with LLMs [6] and appreciate seeing their own usage statistics [16], so this kind of information serves an important need and will affect which LLM-powered applications software developers choose to utilize.

In order to assist programmers learn which LLMs to trust, we plan to investigate and solve the following research questions:

(1) What are important metrics for people to evaluate their trust in LLMs?
(2) How can we use these metrics to create a structured framework that efficiently assesses whether we should trust an LLM's results?
(3) What is the most convenient way to integrate this framework into programming tools for software developers to evaluate an LLM's trustworthiness?

Our solution is LLM-formative, a web-based tool that shows the results of our analysis with various metrics. We use the CodeSearchNet dataset along with manually constructed queries to assess the most ubiquitous AI chatbots with our various metrics. The dataset queries are inputted using an AutoHotkeys script and extracted from the LLM application's user interface. For the first research question, our metrics are constructed to cover both accuracy and more subtle biases that may be missed by state-of-the-art LLM assessment techniques. For the second research question, depending on how much previous research has covered them, we either use a dataset or curate questions ourselves to put together the LLM statistics for the structure of the framework. For the third and final research question, we choose to create a simple Google Website, as this is most easily accessible by programmers around the world using any kind of computer. Our contributions are a comprehensive benchmark for LLM performance in terms of trust and a streamlined way for computer scientists to reference our findings in all parts of the software development lifecycle.

2 RELATED WORKS

While other works have assessed overall trust in LLMs, different categories of trust in computer science, and ways that the Internet affects this trust, ours is the first work that we know of to improve trust in LLMs for developers specifically. There are many conceptual studies that provide trust frameworks for using LLMs. For example, one framework for applying LLMs in the field of trust and safety is setting a clear target for evaluation, creating a labeled dataset for your goals, and creating a distinct testing and validation dataset [18]. Another framework for improving LLM transparency, informed by Human-Centered Interaction and AI research techniques, is model reporting, publishing evaluations, explaining the model, and showing the uncertainty of the generated results [13]. Although these frameworks are useful for informing future research, they do not attempt to provide concrete results, an aspect that this paper will focus on to efficiently inform LLM users about whether to trust the model's results.

The state-of-the-art strategy for learning about and evaluating the trustworthiness of LLMs is benchmarks. These studies divide LLM assessments into various different metrics and attempt to measure each LLM's performance by comparing its generated results with a ground truth dataset that serves as a benchmark. TrustLLM found various LLM characteristics that increase the probability that a model is trustworthy, including utility in specific tasks and having a proprietary code base [8], and Trustworthy LLMs found that LLMs that their owners spent more effort aligning to societal values also performed better in terms of trustworthiness [14]. While these studies cover a plethora of important LLM-related topics, they are broad papers that do not investigate potential applications like software development tasks, as TrustLLM mentions in their Future Works section [8]. In contrast, our paper plans to delve deeper into the development sphere, focusing on each LLM's performance for programming purposes and providing a resource for software developers to understand how much they should trust each LLM for a certain task.

Although there are certain studies that compare how each LLM works for software engineering, such as Enhancing Trust for LLMs [5], these use metrics that focus on model accuracy instead of model trust, which is a slightly different goal than what our paper aims for. This is shown by the significant difference in metrics between SE-focused studies and trust-focused studies, such as one-shot learning and BLEU score for the former [5] versus fairness and robustness for the latter [8]. Other state-of-the-art LLM evaluation metrics include accuracy, model confidence level [17], and innovative strategies like the CONfidence-Quality-ORDer-preserving alignment approach (CONQORD), which is designed to provide better trustworthiness by integrating confidence and order preservation into the model's reward function to enhance its ability to retrieve accurate information [15]. Our study's innovation is to blend these fields together into one, providing a resource that investigates specific aspects of LLM-generated code that humans associate with both accuracy and trust.

Implementing a system for displaying the trustworthiness of LLMs for software developers is a novel idea and there are previous works that have contributed conceptual frameworks for trust of applying LLMs to programming, showing that it is a promising field. A survey-based study showed that a major consideration for people's trust in AI-generated code is whether their community has had a positive experience with it [6]. The same authors found that developers care specifically about practical benefits and trustworthy results for the specific goals of the developers and created a tool that shows people's usage statistics [16]. We seek to improve upon this paper by performing our own analyses beforehand instead of requiring programmers to use the LLMs themselves. In accordance with the PICSE framework for software development tools [10], our tool uses a simple website focusing on mainstream LLMs, provides data-driven metrics, and has a transparent methodology in the form of this paper. Our display of reliable and data-driven metrics will likely have an impact on people's opinions on LLMs and our own survey serves to confirm this.

3 APPROACH

3.1 Datasets

For metrics that use BLEU score to evaluate like Code Quality and Consistency, we use the CodeSearchNet dataset [9]. The HuggingFace corpus was collected and preprocessed by Husain et al. using libraries.io to comb through projects. The search criteria required that the GitHub repositories compiled be publicly available, licensed to allow distribution, open-source, referenced by at least one other project, and not forked from another repository. The data is then sorted in order of popularity, putting projects that more users have starred and forked first. This ensured that only industry-standard, high-quality, and non-duplicate projects with a practical application are incorporated as a part of the collection.

Processing took place with function granularity and excluded any functions that were likely copied from elsewhere, outliers compared to the average function, or generally low quality. The latter category included functions that were shorter than three lines, designed for testing purposes as indicated by a test in the function's name, constructors like __init__, or class extensions like __str__. Duplicate functions are also a cause for concern, and were removed according to the substring-matching algorithm created by Allamanis [2]. Each function's tokens and literals were separated into sets. If two functions had a Jaccard similarity of 0.8 or greater for their token sets and a Jaccard similarity of 0.7 or greater for their literal sets, they were marked as duplicates and excluded from the data.

For metrics that require executing the code, we chose to use Mostly Basic Python Problems (MBPP) [3] instead since it does not require importing any modules. This is a collection of simple Python problems along with three test cases written by crowdsourced workers that were further filtered by the authors to be unambiguous, standard form, and accurately tested.

Finally, for metrics that are difficult to automatically evaluate, we wrote our own natural language prompts for code generation in order to test a specific attribute.

3.2 Metrics

We investigated previous papers on trust for insights on creating our novel metrics that hone in on computer science-specific problems. These include how effectively the code runs, what biases the code contains, and whether the code is original. We will apply these metrics on the LLMs whose trustworthiness we evaluate. While these metrics are inspired by TrustLLM [8] and other papers that investigate LLM performance [5], this is the first paper to our knowledge to apply these software development metrics in the context of LLM trust evaluation.

- Code Quality - This metric investigates how well model-generated code matches standard syntax. We used 300 prompts from CodeSearchNet and compared the generated result to the provided correct code using BLEU score, which looks at the number of substrings in the generated code that are present in the ground truth code. However, this does not fully capture the strength of the LLM's code, since there may be synonymous yet different ways to program the same thing, which we cover with our other metrics.
- Code Performance - This metric looks specifically at whether the model generates code that runs properly. We used 200 prompts from MBPP and ran them on the provided test suite. If the code produced an AttributeError, we concluded that it failed the test. Other errors meant that the code is buggy. Either way, this can be a significant problem for software developers who are relying on the LLM's results.
- Consistency - This trust metric checks whether the LLM gives different results for the same question. We randomly sample 25 prompts from CodeSearchNet and input them five times each to see how much the model output changes, evaluating with BLEU score. If there is significant variation, this suggests that the LLM has a high temperature, or randomness of text. Thus, it may not be reliable for the precision that code requires.
- Code Efficiency - This metric ensures that the LLM generates the optimal code for the problem at hand. We will ask the model to provide an algorithm with a known solution and ensure that its answer has the most efficient time complexity possible.
- Hallucination - We evaluate how often the model replies with information that is blatantly incorrect by asking it to generate code that is impossible to create, such as features that are not present in the particular language, a function that does not exist, or an algorithm with a time complexity that is not yet possible. This serves to test the model's ability to admit that it cannot achieve a task.
- Language Bias - Since many software developers working on different projects use LLMs, it is important to know if the model has any inherent biases towards any languages. We will assess this by asking a programming question that does not specify a language and see whether the model outputs are in a certain coding language more than others.
- Plagiarism - It is essential that programmers can trust that LLMs do not generate plagiarized code, as this could put them in legal trouble. While it is difficult to know exactly what documents each model was trained on, we

can see if the models have a tendency to copy their sources without putting the appropriate references. Thus, we look at whether the LLM provides sources or appears to synthesize multiple sources into a single answer.

3.3 LLM Applications

For this project, our goal is to evaluate the LLMs that people use most frequently for programming purposes, especially in academic and professional settings. While there are many open-source LLMs on HuggingFace being actively developed for research, these models are not yet applied as frequently in real-world settings. We believe it will make more of an impact to enhance people's trust in LLMs they are actively using and may provide more accurate survey results.

A survey found that 69% of their respondents choose ChatGPT as their primary LLM, with this percentage increasing to 87.1% among university students and 92% among individuals earning more than $125 thousand annually [11]. As such, the LLM chatbots we chose to test are ChatGPT (GPT-4o mini) and Microsoft Copilot. We intentionally decided upon mainstream models that are evidently especially well-known among students and professionals instead of ones that are primarily used by LLM researchers in order to gear our project to its audience.

One humorous observation we discovered is that ChatGPT can have hallucinations about itself. In order to find necessary details for this subsection, we attempted to ask ChatGPT what version it was on, and it responded by first giving the correct answer of GPT-4o mini before backtracking and stating that free users can only use GPT-3.5. Upon further questioning, it assumed we must be on the paid plan, despite previously stating it can detect whether users are free or paid. Finally, it came to the correct conclusion upon aggressive confrontation of its hallucinations.

3.4 Implementation

Note that many of the models mentioned in the previous subsection do not possess publicly usable APIs. This a limitation we address with our implementation. We use the open-source language AutoHotKeys [4] to automatically input text to LLM GUIs, such as the ubiquitous ChatGPT prompt. While this allows us to simulate the process of a user typing in several prompts, it also hinders the number of samples we can reasonably utilize. Therefore, we plan to keep the number of our inputs fairly low, which also serves to assist human comprehensibility of the results.

In order to extract the outputs of the model, we will copy the text from the GUI, paste it into a text file, and extract the important components using a Python script. In certain cases, we then compare the outputs to the ground truth; in other cases, since the effect we're looking for is subtle, we will manually review the outputs to categorize them appropriately.

4 CASE STUDY DESIGN

In our case study, we will investigate whether our method of displaying our results is effective and whether the trust analysis makes a noticeable impact in how much software developers trust LLM applications.

4.1 User Interface

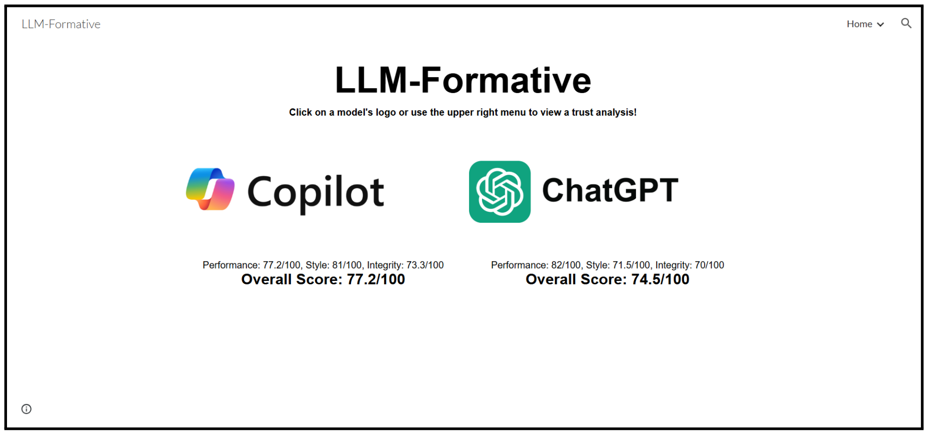
We chose a Google Sites website [7] for our assessment because it is easy to access and displays seamlessly on both PCs and mobile devices, thus promoting our goal of making LLM-formative convenient for users. The home page shows each of the five models we evaluated, and the user can click on each to see their specific statistics. We then translate the model's results for each metric into a percentage score and display the result in a chart, changing the color from green to red depending on how trustworthy the result is. This visual display quickly shows whether to trust that aspect of the model as green conveys trust while red conveys danger, as opposed to requiring the user to understand the data and calculations.

Pie charts serve as a colorful and engaging way to convey the percentage of results from large datasets without forcing users to witness too many numbers. We also add mouseover functions to assist in understanding, such as displaying the percentage and number of samples for each slice and popup text explaining the meaning of metrics that use jargon. We display the more subjective results in boxes with further text descriptions so users can receive a comprehensive overview of the results.

On the title page, we show the logos of the models with a trust evaluation, in this case Copilot and ChatGPT. In addition, we integrate each of the metrics into an overall score and three subscores:

```
Performance = Average of Code Performance, Code Efficiency, & Consistency
Style = Average of Code Quality & Consistency
Integrity = Average of Hallucination, Language Bias, & Plagiarism.
Overall = Average of Performance, Consistency, & Integrity
```

Each metric is rated such that a higher score is more desirable. For this calculation, pie chart metrics use their overall scores which are Average BLEU Score for Code Quality and Consistency as well as pass^3 for Code Performance. Code Quality and Code Performance are weighted over a total of 50 to avoid a deceptively low overall score. LLM-Formative's title page design is shown in Figure 1 and its model page design in Figure 2. ChatGPT and Copilot's model pages have an identical format, albeit with different results.



LLM-Formative

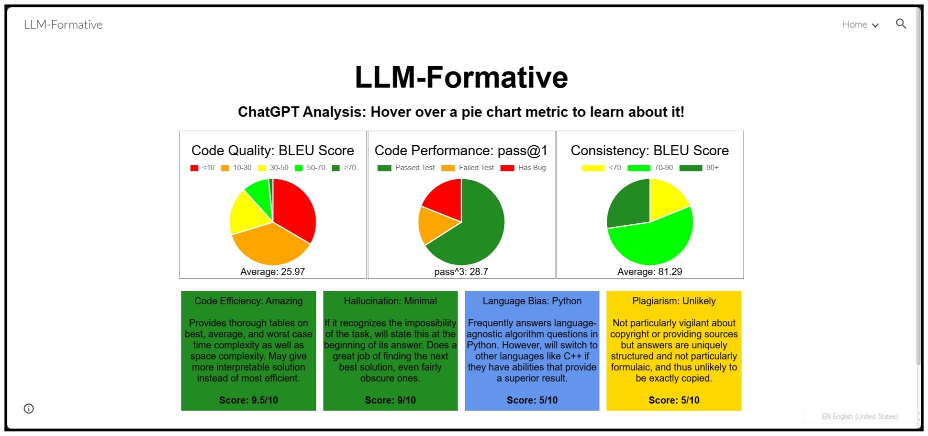
Click on a model's logo or use the upper right menu to view a trust analysis!

Performance: 77.2/100, Style: 81/100, Integrity: 73.3/100
Overall Score: 77.2/100

Performance: 82/100, Style: 71.5/100, Integrity: 70/100
Overall Score: 74.5/100

Fig. 1. LLM-Formative Title Page

Note that the website does not automatically perform the trust analysis and instead displays the result of our completed evaluation. This makes it highly efficient on the user's end and prevents the website from being subject to any updates in the LLM application that may affect our approach, such as a change to the LLM GUI that affects prompt input. However, this has the downside of requiring a full update to the website if we ever need to update our analysis for new models.



LLM-Formative

ChatGPT Analysis: Hover over a pie chart metric to learn about it!

Code Efficiency: Amazing

Provides thorough tables on best, average, and worst case time complexity as well as space complexity. May give more interpretable solution instead of most efficient.

Score: 9.5/10

Hallucination: Minimal

If it recognizes the impossibility of the task, will state this at the beginning of its answer. Does a great job of finding the next best solution, even fairly obscure ones.

Score: 9/10

Language Bias: Python

Frequently answers language-agnostic algorithm questions in Python. However, will switch to other languages like C++ if they have abilities that provide a superior result.

Score: 5/10

Plagiarism: Unlikely

Not particularly vigilant about copyright or providing sources but answers are uniquely structured and not particularly formulaic, and thus unlikely to be exactly copied.

Score: 5/10

Fig. 2. LLM-Formative Model Page

4.2 Survey

Our goal is for LLM-formative is to provide convenient and useful information for users of LLM applications. Therefore, after the website was fully implemented, we showcased our results to programmers in our university with a survey. Each programmer gave their informed consent after recognizing the purpose of this project and understanding what kinds of information we plan to collect for this study. We began the survey by obtaining necessary demographic data, like age, gender, and years of experience with computer science. Each participant likely has extensive experience with utilizing LLMs, making them an ideal audience for this case study.

The survey's purpose is to determine factors including how useful the trust metrics are, their previous opinions on each model, whether viewing the LLM-formative webpage affected their trust in the models, if there were any major surprises in the results, and ways we could improve upon this work. The survey contains questions that are both quantitative, using a rating from 1-5 for gauging user sentiments, and qualitative, allowing participants to provide extensive comments. The qualitative questions are written in a neutral manner and aside from one question asking for suggestions, they do not evoke any particular emotions.

5 RESULTS

5.1 Trust Evaluation

The results from trust evaluations of the LLMs for software development are shown in Table 1, with significant differences where the scores differ by more than 10 emphasized in bold text. Subsection 3.2 contains further details on how we created these trust scores. A visualization of the result can be found in the LLM-Formative webpage, accessible through this link:
https://sites.google.com/view/llm-formative/home

Note that higher scores mean the result is more trustworthy. Copilot and ChatGPT perform similarly at code generation and have a similar level of bias and hallucination. However, the metrics successfully capture noticeable yet subtle differences in LLM performance. Copilot shows better reliability, outperforming ChatGPT at Consistency and Plagiarism, suggesting that it may use the same dataset more frequently and have a lower LLM temperature. Conversely, ChatGPT achieves a significantly higher Code Efficiency, which stems from finding more complicated and efficient

Metric/Model	Copilot	ChatGPT
Code Quality	28.04	25.97
Code Performance	24.4	28.7
Consistency	**93.53**	81.29
Code Efficiency	75	**95**
Hallucination	80	90
Language Bias	40	50
Plagiarism	**100**	50
Overall Score	77.2	74.5

Table 1. LLM Trust Evaluation Results

algorithms as well as explaining them more comprehensively. This does a good job at conveying their different use cases, where Copilot is more useful if you want to ensure that results are written well with trustworthy sources and ChatGPT is more useful if you simply need code that works best in the least time.

5.2 Survey Response

We begin with an overview of the survey demographics. Our survey respondents had a wide variety in gender and ranged in age from 19 to 28. Additionally, many of them had extensive experience in computer science, with an average of 3.3 years and a range from 0 to 9 years. Eight respondents had experience with at least 1 LLM, showing a diverse range of LLM experience.

	Age	Gender	Experience (Years)		
			Computer Science	ChatGPT	Copilot
Respondent 1	22	Male	4	2	0
Respondent 2	20	Female	0	1	0
Respondent 3	22	Female	0	0	0
Respondent 4	20	Male	3	0	0
Respondent 5	23	Female	8	1.5	1
Respondent 6	19	Female	2	1	1
Respondent 7	20	Unsure	0	1	0
Respondent 8	21	Male	9	2	0
Respondent 9	28	Male	0	1	0
Respondent 10	20	Male	7	2	0

Table 2. Survey Respondent Demographics

Next, we dive into an analysis of the ratings. The survey showed that viewing LLM-Formative successfully increased people's trust in LLMs, as the average rating of ChatGPT increased from 2.9 to 3.3 stars while the average rating of Copilot increased from 2.5 to 3.0 stars. This was not swayed by a lack of experience, as those with 0 years of experience with either LLM overwhelmingly rated it 3 stars before viewing LLM-Formative.

Moreover, respondents generally had a high opinion of LLM-Formative, as the average rating of the website design, trust metrics, and overall site was 4.33, 3.4, and 4 stars respectively. We excluded one answer for the website design question as the respondent misunderstood the question as asking about her web design skills.

6 DISCUSSION

In this section, we investigate specific feedback from respondents and identify significant advantages and limitations of LLM-Formative. This is done by performing a holistic analysis, reading through every answer and looking for significant changes in the LLM opinions of survey participants and reactions to each feature of the website.

We begin with the advantages. Respondents praised LLM-Formative's style, lauding its simplicity and clarity for enhancing the website's navigation. Below are a few representative comments:

> Everything is very straightforward, all the important information is right on the home page without additional searching or navigation needed. You give directions at the top on how to get more detailed metrics and it is, once again, a very simple single-click function.

> 5/5 in terms of functionality and navigability, main improvements could come from site aesthetics (colors, texts, etc) but it still remains relatively accessible and that's the priority.

> The website design is very clean and consistent. More details could improve the overall experience, but what is there is very understandable.

Criticisms were similarly consistent, mentioning limitations like a lack of mobile support and insufficient details on the trust metrics. Some of these are quoted here:

> Mobile support and better formatting of the website may be nice. Its currently a white void with icons floating around without much else.

> Provide an overview of the research performed, or a link to the study [...] the website alone has no context.

> The metrics here are a good starting point, though for SE developers, it may be helpful to be able to hover over them and get what the answers mean.

To summarize the survey results, the utility of the website design is the strongest aspect of LLM-Formative. The suggestions for improvement include improving the mobile display of the website, providing explanations on the use cases for the trust metrics, showing the exact methodology behind the LLM evaluation, expanding to more LLMs, and adding more metrics like math and coding paradigms. The primary challenge to address is to balance simplicity with detail, which serves to improve LLM-Formative's weaknesses while maintaining its strengths.

7 CONCLUSION

This paper introduced LLM-Formative, a web-based tool to assist software developers in understanding whether to trust code generated by LLM chatbots. We obtained scores for seven metrics by designing an AutoHotkey script to automatically input prompts into the LLM GUI and analyzing the results with a Python script. Next, we integrated the metric scores into a Google Sites webpage, forming a title page with overall scores and a model page for Microsoft Copilot and ChatGPT. Finally, we shared a survey with respondents consisting of 10 students with computer science experience ranging from 0 years to 9 years. The participants showed an overall increase in trust of about half a star for each of the LLMs we did a trust evaluation on. In addition, rating of our website was 4 stars on average, and more importantly, we received valuable feedback which can guide future iterations of our work moving forward. Future works include improving device generalization, making the trust metrics more accessible, and expanding the study to other LLMs, diverse metrics, and larger datasets.

ACKNOWLEDGMENT

Special thanks to the future Dr. Daniel Otten for encouraging me throughout this project.

8 REPLICATION PACKAGE

We have created a replication package via the following GitHub repository. Please contact the primary author for any further inquiries.
https://github.com/M-Chen-3/LLM-Formative

REFERENCES

[1] Accenture. 2021. A New Era of Generative AI for Everyone. https://www.accenture.com/content/dam/accenture/final/accenture-com/document/Accenture-A-New-Era-of-Generative-AI-for-Everyone.pdf

[2] Miltiadis Allamanis. 2018. The Adverse Effects of Code Duplication in Machine Learning Models of Code. *CoRR* abs/1812.06469 (2018). arXiv:1812.06469 http://arxiv.org/abs/1812.06469

[3] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program Synthesis with Large Language Models. arXiv:2108.07732 [cs.PL] https://arxiv.org/abs/2108.07732

[4] AutoHotkey. 2025. AutoHotkey. https://www.autohotkey.com/

[5] Nik Bear Brown. 2024. Enhancing Trust in LLMs: Algorithms for Comparing and Interpreting LLMs. arXiv:2406.01943 [cs.CL] https://arxiv.org/abs/2406.01943

[6] Ruijia Cheng, Ruotong Wang, Thomas Zimmermann, and Denae Ford. 2024. "It would work for me too": How Online Communities Shape Software Developers' Trust in AI-Powered Code Generation Tools. *ACM Transactions on Interactive Intelligent Systems* 14, 2, Article 11 (May 2024), 39 pages. https://doi.org/10.1145/3651990

[7] Google. 2025. Google Sites. https://sites.google.com/.

[8] Yue Huang, Lichao Sun, Haoran Wang, Siyuan Wu, Qihui Zhang, Yuan Li, Chujie Gao, Yixin Huang, Wenhan Lyu, Yixuan Zhang, Xiner Li, Zhengliang Liu, Yixin Liu, Yijue Wang, Zhikun Zhang, Bertie Vidgen, Bhavya Kailkhura, Caiming Xiong, Chaowei Xiao, Chunyuan Li, Eric Xing, Furong Huang, Hao Liu, Heng Ji, Hongyi Wang, Huan Zhang, Huaxiu Yao, Manolis Kellis, Marinka Zitnik, Meng Jiang, Mohit Bansal, James Zou, Jian Pei, Jian Liu, Jianfeng Gao, Jiawei Han, Jieyu Zhao, Jiliang Tang, Jindong Wang, Joaquin Vanschoren, John Mitchell, Kai Shu, Kaidi Xu, Kai-Wei Chang, Lifang He, Lifu Huang, Michael Backes, Neil Zhenqiang Gong, Philip S. Yu, Pin-Yu Chen, Quanquan Gu, Ran Xu, Rex Ying, Shuiwang Ji, Suman Jana, Tianlong Chen, Tianming Liu, Tianyi Zhou, William Wang, Xiang Li, Xiangliang Zhang, Xiao Wang, Xing Xie, Xun Chen, Xuyu Wang, Yan Liu, Yanfang Ye, Yinzhi Cao, Yong Chen, and Yue Zhao. 2024. TrustLLM: Trustworthiness in Large Language Models. arXiv:2401.05561 [cs.CL] https://arxiv.org/abs/2401.05561

[9] Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. 2019. CodeSearchNet Challenge: Evaluating the State of Semantic Code Search. *CoRR* abs/1909.09436 (2019). arXiv:1909.09436 http://arxiv.org/abs/1909.09436

[10] Brittany Johnson, Christian Bird, Denae Ford, Nicole Forsgren, and Thomas Zimmermann. 2023. Make Your Tools Sparkle with Trust: The PICSE Framework for Trust in Software Tools. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. 409–419. https://doi.org/10.1109/ICSE-SEIP58684.2023.00043

[11] Priit Kallas. 2024. LLM survey 2024: generative AI adoption statistics at work. *Amperly* (2024). https://amperly.com/llm-survey-generative-ai-adoption-statistics/

[12] Weixin Liang, Yaohui Zhang, Zhengxuan Wu, Haley Lepp, Wenlong Ji, Xuandong Zhao, Hancheng Cao, Sheng Liu, Siyu He, Zhi Huang, Diyi Yang, Christopher Potts, Christopher D Manning, and James Y. Zou. 2024. Mapping the Increasing Use of LLMs in Scientific Papers. arXiv:2404.01268 [cs.CL] https://arxiv.org/abs/2404.01268

[13] Q. Vera Liao and Jennifer Wortman Vaughan. 2024. AI Transparency in the Age of LLMs: A Human-Centered Research Roadmap. *Harvard Data Science Review* Special Issue 5 (may 31 2024). https://hdsr.mitpress.mit.edu/pub/aelql9qy.

[14] Yang Liu, Yuanshun Yao, Jean-Francois Ton, Xiaoying Zhang, Ruocheng Guo, Hao Cheng, Yegor Klochkov, Muhammad Faaiz Taufiq, and Hang Li. 2024. Trustworthy LLMs: a Survey and Guideline for Evaluating Large Language Models' Alignment. arXiv:2308.05374 [cs.AI] https://arxiv.org/abs/2308.05374

[15] Shuchang Tao, Liuyi Yao, Hanxing Ding, Yuexiang Xie, Qi Cao, Fei Sun, Jinyang Gao, Huawei Shen, and Bolin Ding. 2024. When to Trust LLMs: Aligning Confidence with Response Quality. arXiv:2404.17287 [cs.CL] https://arxiv.org/abs/2404.17287

[16] Ruotong Wang, Ruijia Cheng, Denae Ford, and Thomas Zimmermann. 2024. Investigating and Designing for Trust in AI-powered Code Generation Tools. In *The 2024 ACM Conference on Fairness, Accountability, and Transparency (FAccT '24)*. ACM, 1475–1493. https://doi.org/10.1145/3630106.3658984

[17] Haoyan Yang, Yixuan Wang, Xingyin Xu, Hanyuan Zhang, and Yirong Bian. 2024. Can We Trust LLMs? Mitigate Overconfidence Bias in LLMs through Knowledge Transfer. arXiv:2405.16856 [cs.CL] https://arxiv.org/abs/2405.16856

[18] Doohee You and Dan Chon. 2024. Trust Safety of LLMs and LLMs in Trust Safety. arXiv:2412.02113 [cs.AI] https://arxiv.org/abs/2412.02113