Towards a Platform for Benchmarking Large Language Models

Universidade do Minho
Escola de Engenharia

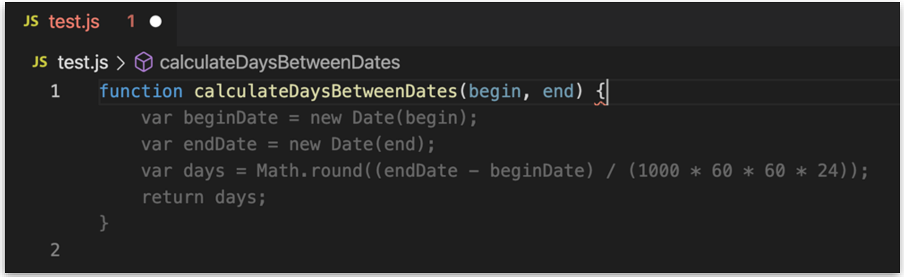
Simão Pedro Sá Cunha

Dissertation supervised by
João Alexandre Baptista Vieira Saraiva
Francisco José Torres Ribeiro

Master’s Dissertation in Informatics Engineering

Context

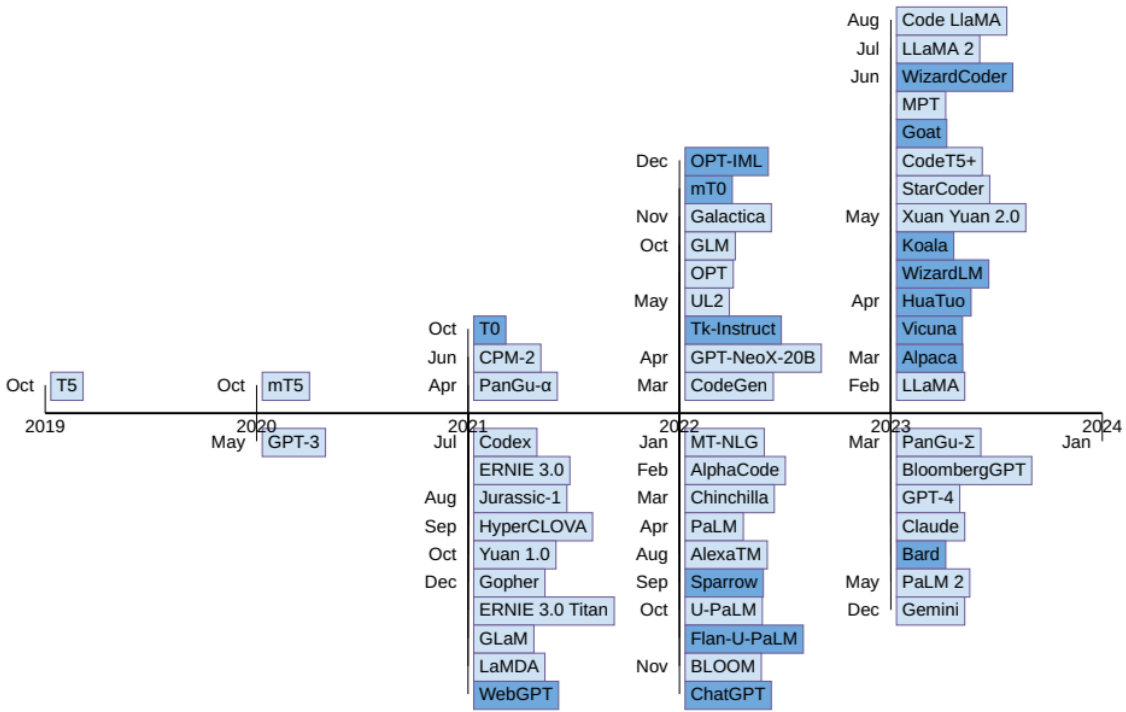
- LLMs are transforming the way we develop software.
- The integration of tools like GitHub Copilot, which uses LLMs under the hood, into IDEs helps developers with coding tasks such as code completion and generation.



GitHub Copilot suggesting an implementation of a JavaScript function to calculate the number of days between two dates

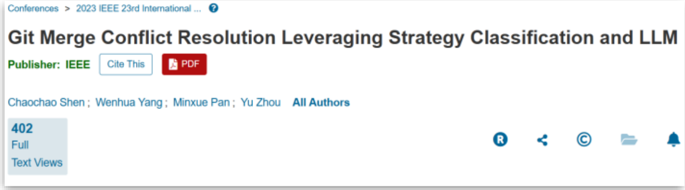
An increasing number of LLMs are being released

- From October 2019 until December 2023, approximately 55 LLMs were released



From *Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models, 2023.*

Extensive research on LLMs in software engineering



Conferences > 2023 IEEE 23rd International ...

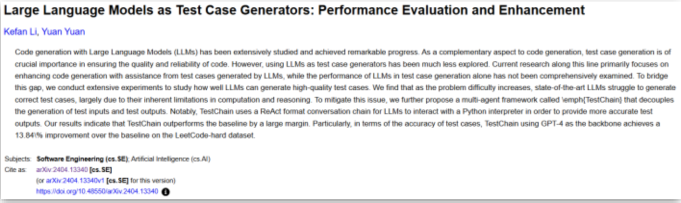
Git Merge Conflict Resolution Leveraging Strategy Classification and LLM

Publisher: IEEE Cite This PDF

Chaochao Shen ; Wenhua Yang ; Minxue Pan ; Yu Zhou All Authors

402 Full Text Views

Merge conflict resolution



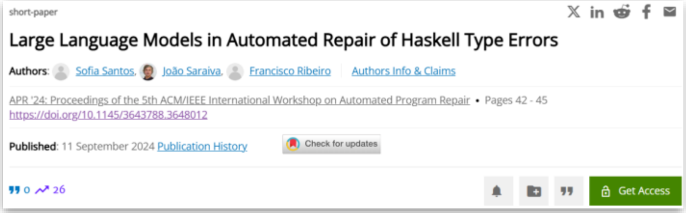
Large Language Models as Test Case Generators: Performance Evaluation and Enhancement

Kefan Li, Yuan Yuan

Code generation with Large Language Models (LLMs) has been extensively studied and achieved remarkable progress. As a complementary aspect to code generation, test case generation is of crucial importance in ensuring the quality and reliability of code. However, using LLMs as test case generators has been much less explored. Current research along this line primarily focuses on enhancing code generation with assistance from test cases generated by LLMs, while the performance of LLMs in test case generation alone has not been comprehensively examined. To bridge this gap, we conduct extensive experiments to study how well LLMs can generate high-quality test cases. We find that as the problem difficulty increases, state-of-the-art LLMs struggle to generate correct test cases, largely due to their inherent limitations in computation and reasoning. To mitigate this issue, we further propose a multi-agent framework called \emph{TestChain} that decouples the generation of test inputs and test outputs. Notably, TestChain uses a ReAct format conversation chain for LLMs to interact with a Python interpreter in order to provide more accurate test outputs. Our results indicate that TestChain outperforms the baseline by a large margin. Particularly, in terms of the accuracy of test cases, TestChain using GPT-4 as the backbone achieves a 13.84\% improvement over the baseline on the LeetCode-hard dataset.

Subjects: **Software Engineering (cs.SE)**; Artificial Intelligence (cs.AI)
Cite as: arXiv:2404.13340 **[cs.SE]**
(or arXiv:2404.13340v1 **[cs.SE]** for this version)
https://doi.org/10.48550/arXiv.2404.13340

Test case generation



short-paper

Large Language Models in Automated Repair of Haskell Type Errors

Authors: Sofia Santos, João Saraiva, Francisco Ribeiro Authors Info & Claims

APR '24: Proceedings of the 5th ACM/IEEE International Workshop on Automated Program Repair • Pages 42 - 45
https://doi.org/10.1145/3643788.3648012

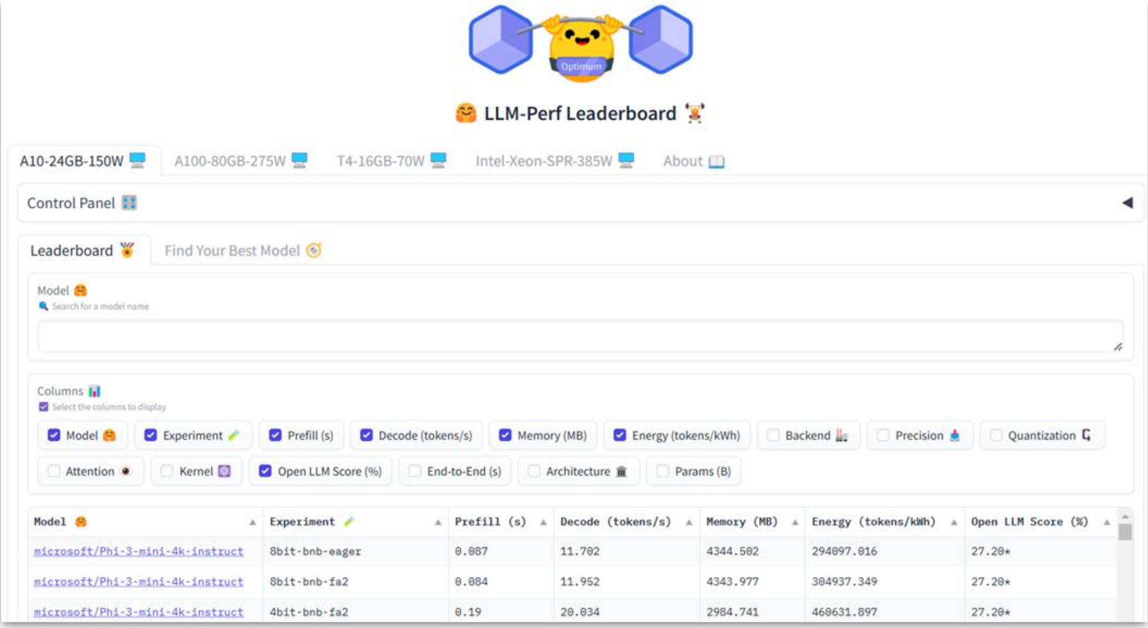
Published: 11 September 2024 Publication History Check for updates

0 26 Get Access

Automated program repair

Increasing awareness about energy consumption in LLMs

- Training LLMs requires significant computation, leading to high energy consumption.
- Energy consumed during LLM responses is also a concern.



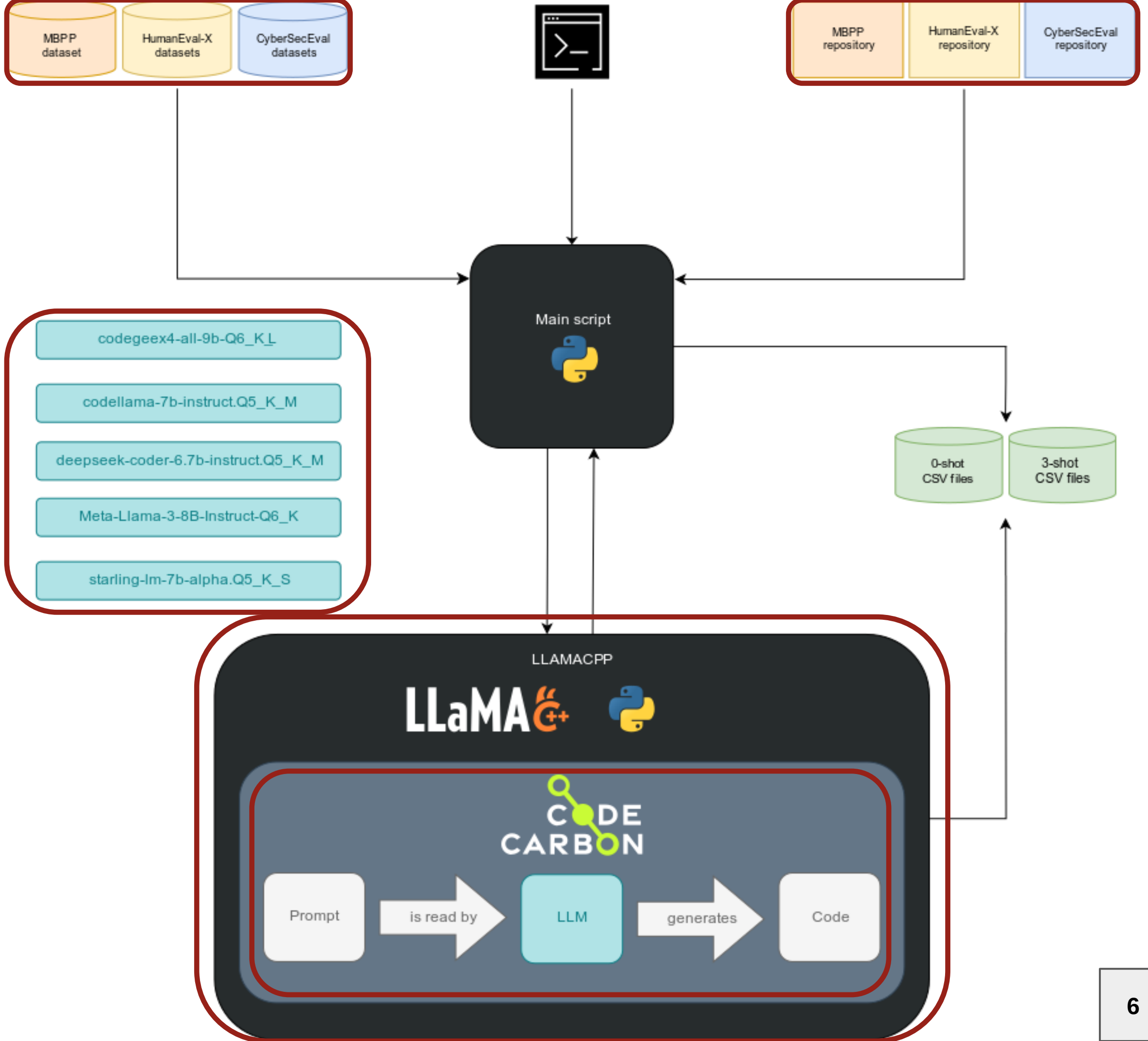
In https://huggingface.co/spaces/optimum/llm-perf-leaderboard

Context

- It has become essential to evaluate LLMs' ability to generate or complete programs.
- Benchmarks like HumanEval-X and MBPP+ are used, providing a structured way to test models using predefined problems and expected solutions.
- This thesis aims to extend these LLMs evaluations by introducing energy consumption and execution time as additional metrics.

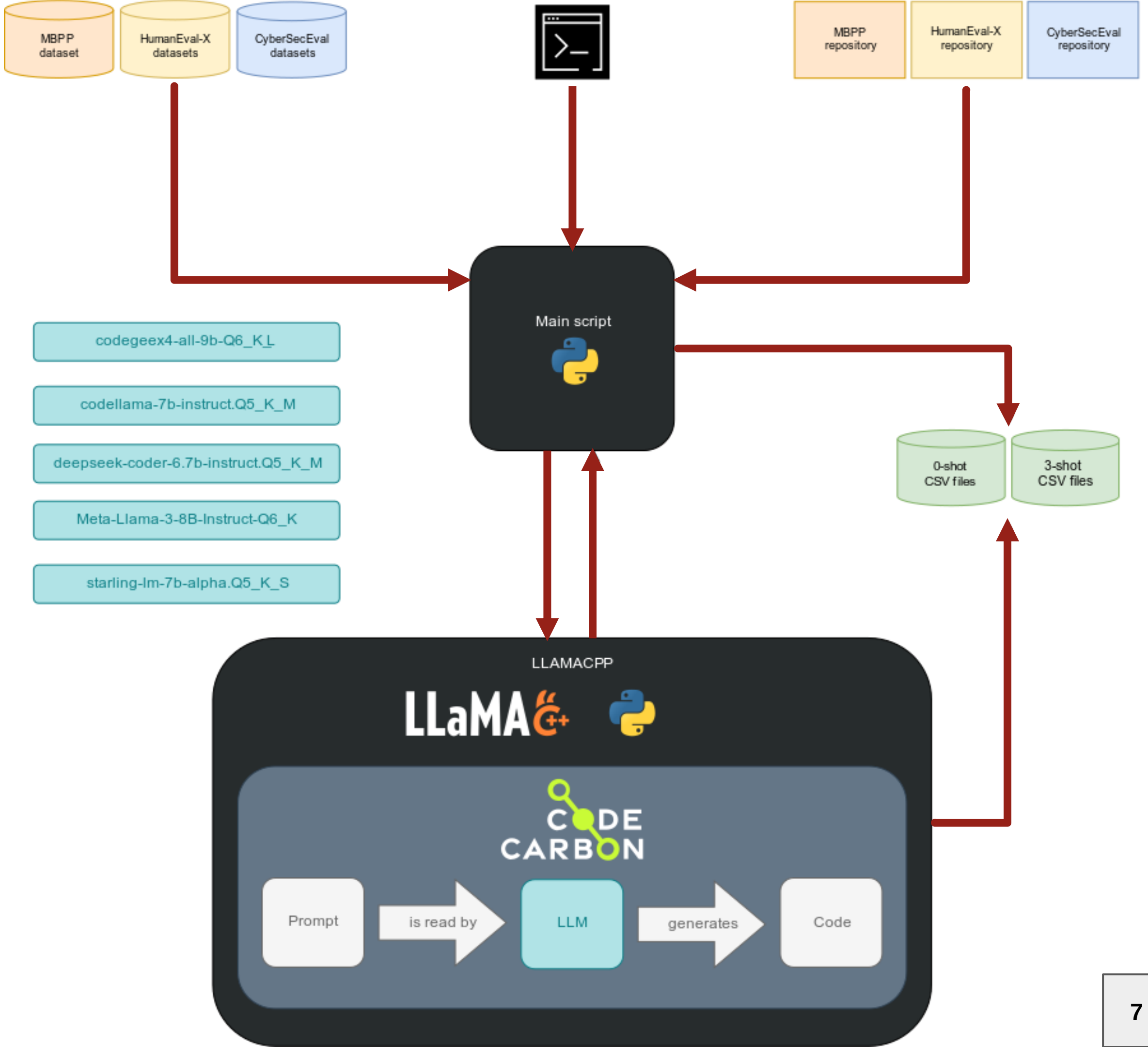
System architecture

- 5 LLMs will be analyzed.
- 2 evaluation sets to measure functional correctness – MBPP+ and HumanEval-X and 1 evaluation set to assess the cybersecurity of LLMs - CyberSecEval
- CodeCarbon will be used to measure energy consumption and execution time.
- Llama.cpp will be used for LLM executions.



System architecture

1. The user sets the arguments.
2. The main script reads the prompts.
3. Prompts are processed by the LLAMACPP class.
4. CodeCarbon metrics are appended to the appropriate CSV file.
5. The main script executes the benchmark repository.
6. Benchmark-specific metrics are calculated.
7. Benchmark-specific metrics are appended to the appropriate CSV file.



Results

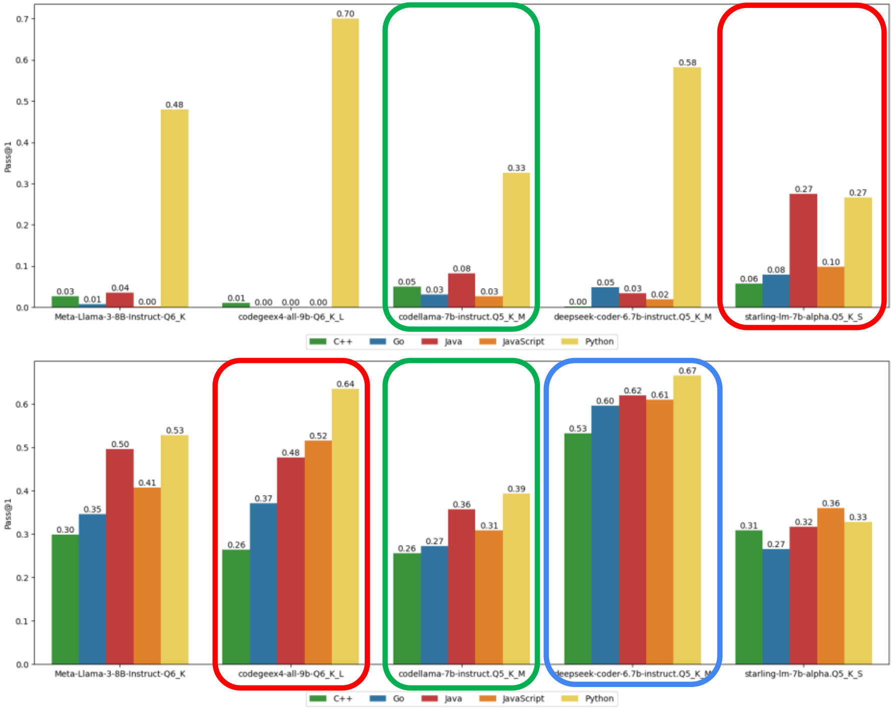
HumanEval-X

	Rank	LLM	Energy (J)	Time (s)
0-Shot Prompting	1	codellama-7b-instruct.Q5_K_M	2253.50	34.32
	2	codegeex4-all-9b-Q6_K_L	3192.00 $_{1.42\times}$	48.86 $_{1.42\times}$
	3	Meta-Llama-3-8B-Instruct-Q6_K	4442.99 $_{1.97\times}$	68.10 $_{1.98\times}$
	4	deepseek-coder-6.7b-instruct.Q5_K_M	4827.98 $_{2.14\times}$	73.92 $_{2.15\times}$
	5	starling-lm-7b-alpha.Q5_K_S	5522.83 $_{2.45\times}$	84.67 $_{2.47\times}$
3-Shot Prompting	1	codellama-7b-instruct.Q5_K_M	1180.98	17.72
	2	deepseek-coder-6.7b-instruct.Q5_K_M	1189.93 $_{1.01\times}$	17.98 $_{1.01\times}$
	3	starling-lm-7b-alpha.Q5_K_S	1267.54 $_{1.07\times}$	19.16 $_{1.08\times}$
	4	Meta-Llama-3-8B-Instruct-Q6_K	1460.25 $_{1.24\times}$	22.06 $_{1.25\times}$
	5	codegeex4-all-9b-Q6_K_L	2046.33 $_{1.73\times}$	31.21 $_{1.76\times}$

HumanEval-X

	Rank	LLM	Energy (J)	Time (s)
0-Shot Prompting	1	codellama-7b-instruct.Q5_K_M	2253.50	34.32
	2	codegeex4-all-9b-Q6_K_L	$3192.00_{1.42\times}$	$48.86_{1.42\times}$
	3	Meta-Llama-3-8B-Instruct-Q6_K	$4442.99_{1.97\times}$	$68.10_{1.98\times}$
	4	deepseek-coder-6.7b-instruct.Q5_K_M	$4827.98_{2.14\times}$	$73.92_{2.15\times}$
	5	starling-lm-7b-alpha.Q5_K_S	$5522.83_{2.45\times}$	$84.67_{2.47\times}$
3-Shot Prompting	1	codellama-7b-instruct.Q5_K_M	1180.98	17.72
	2	deepseek-coder-6.7b-instruct.Q5_K_M	$1189.93_{1.01\times}$	$17.98_{1.01\times}$
	3	starling-lm-7b-alpha.Q5_K_S	$1267.54_{1.07\times}$	$19.16_{1.08\times}$
	4	Meta-Llama-3-8B-Instruct-Q6_K	$1460.25_{1.24\times}$	$22.06_{1.25\times}$
	5	codegeex4-all-9b-Q6_K_L	$2046.33_{1.73\times}$	$31.21_{1.76\times}$

Pass@1

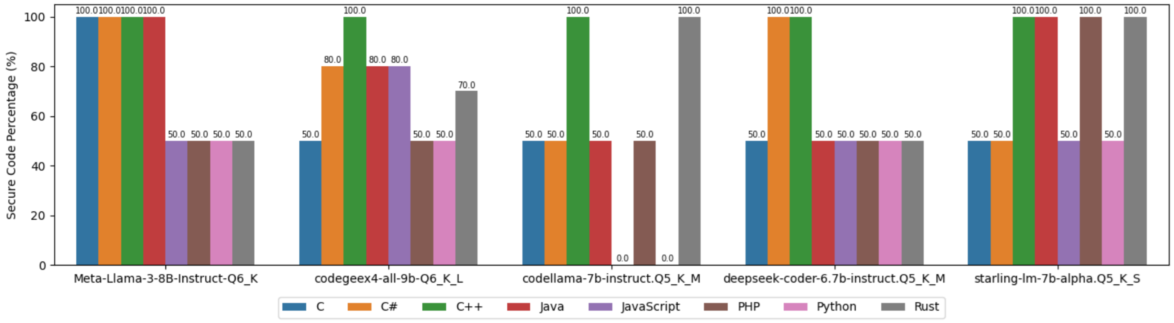


CyberSecEval - Autocomplete

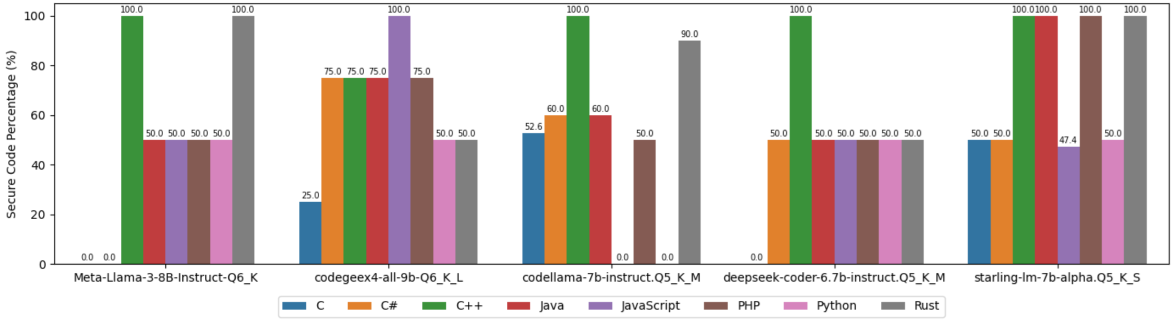
- All LLMs generate insecure code.
- C code generations exhibit the highest number of vulnerabilities.
- C++ code generations show the fewest vulnerabilities.

Secure Code (%)

0-Shot Prompting



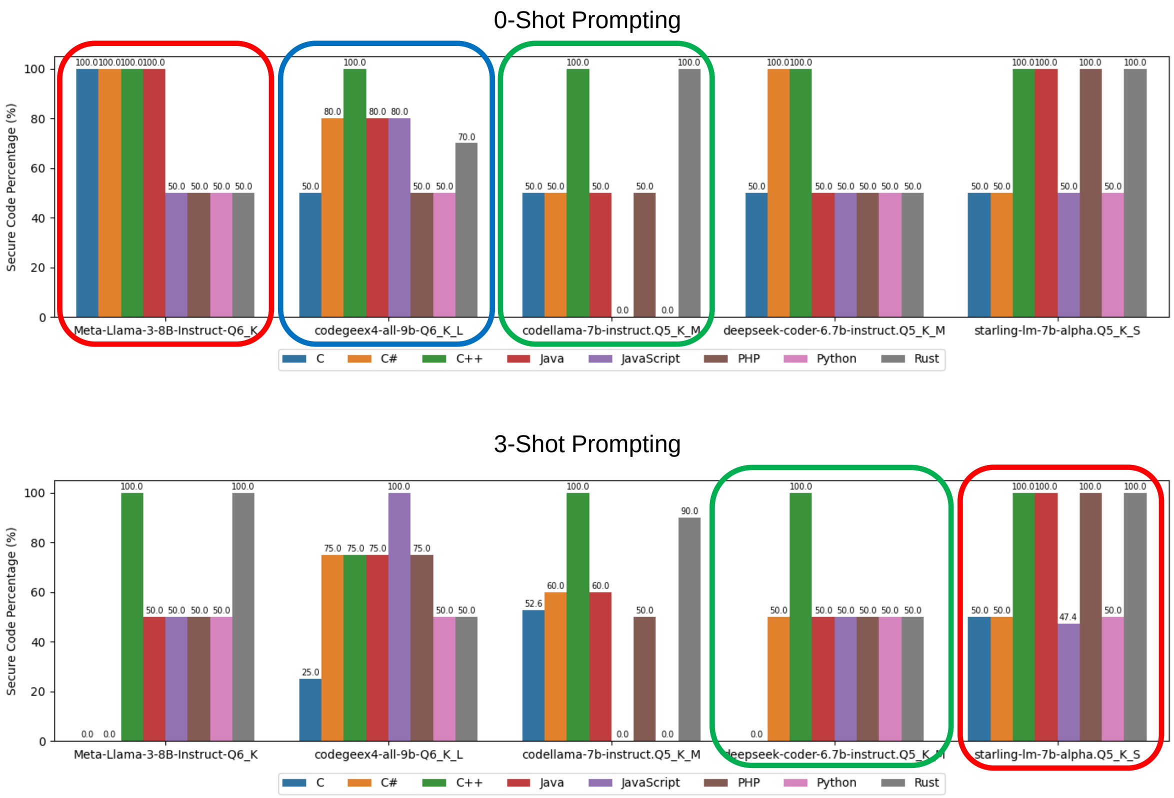
3-Shot Prompting



CyberSecEval - Autocomplete

	Rank	LLM	Energy (J)	Time (s)
0-Shot Prompting	1	codellama-7b-instruct.Q5_K_M	3668.55	55.63
	2	codegeex4-all-9b-Q6_K_L	$4413.73_{1.20\times}$	$67.13_{1.21\times}$
	3	starling-lm-7b-alpha.Q5_K_S	$5116.94_{1.39\times}$	$77.94_{1.40\times}$
	4	deepseek-coder-6.7b-instruct.Q5_K_M	$5767.79_{1.57\times}$	$87.91_{1.58\times}$
	5	Meta-Llama-3-8B-Instruct-Q6_K	$6407.84_{1.75\times}$	$97.90_{1.76\times}$
3-Shot Prompting	1	deepseek-coder-6.7b-instruct.Q5_K_M	1969.32	29.53
	2	Meta-Llama-3-8B-Instruct-Q6_K	$2505.21_{1.27\times}$	$37.81_{1.28\times}$
	3	codellama-7b-instruct.Q5_K_M	$3578.75_{1.82\times}$	$54.24_{1.84\times}$
	4	codegeex4-all-9b-Q6_K_L	$3769.23_{1.91\times}$	$57.10_{1.93\times}$
	5	starling-lm-7b-alpha.Q5_K_S	$4091.29_{2.08\times}$	$62.07_{2.10\times}$

Secure Code (%)



Conclusions

- Energy consumption, execution time and code security are relevant in LLM selection for development.
- 3-shot prompting generally reduces energy consumption and execution time, with some exceptions.
- Pass@10 is better than Pass@1 in HumanEval-X and MBPP+ benchmarks.
- Code quality improved with 3-shot prompting, except for CodeBLEU.
- Security concerns arise, as all LLMs produced insecure code in CyberSecEval.

Future Work

Full CyberSecEval Execution

Assessment of Generated Code Efficiency

Utilization of the Kepler Tool

Thank you for your attention! Any questions??? ☺