

A Study of LLMs' Preferences for Libraries and Programming Languages

Lukas Twist Mark Harman Don Syme Joost Noppen
Helen Yannakoudakis Detlef Nauck Jie M. Zhang



Full paper:



Lukas Twist:

LLMs Love Python: they overwhelmingly default to it, even for high-performance tasks where it is explicitly suboptimal. By favouring familiar defaults over actual task suitability, LLMs create negative consequences for security, performance, and open-source diversity.



Motivation

- Code-generation benchmarks mostly ask: does it work?
- But LLMs also make hidden design choices: which library? which language? safe, efficient or maintainable?
- When users omit technical details, *LLMs can silently steer technology adoption.*



Methodology

- Measured library and language choice in two settings: benchmark tasks and project initialisation.
- What do LLMs use when the technology is not specified?
- Do LLMs' natural language recommendations match what they actually use when generating code?



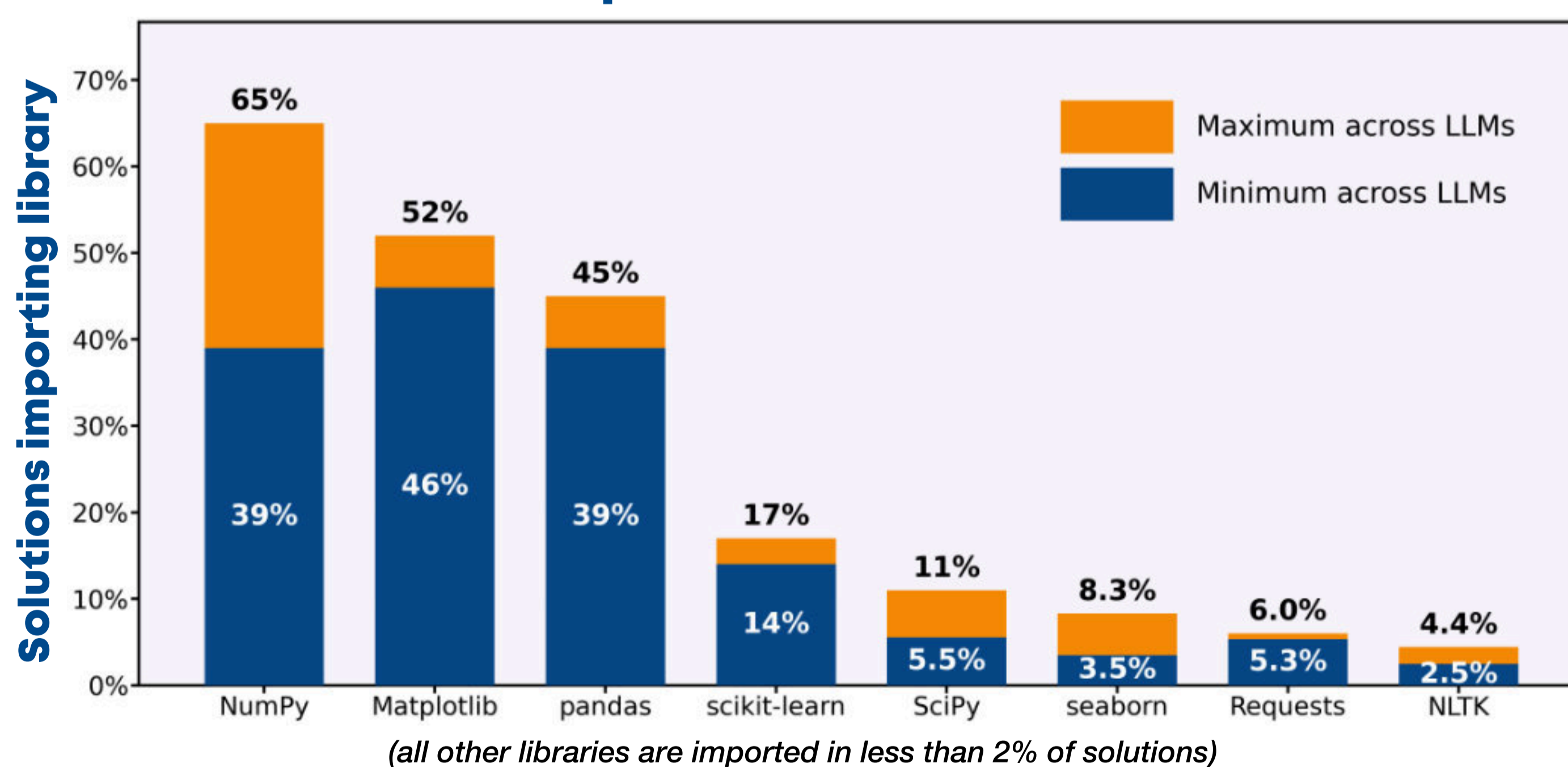
Library Findings

- The same few libraries dominate usage. Up to **45% of NumPy usage diverges** from ground-truth solutions.
- Each LLM used only 32–39 distinct libraries.
- Across tasks, *LLMs strongly favour older defaults over high-momentum alternatives*: pandas over Polars, Flask over FastAPI, Dask over Ray/Celery.

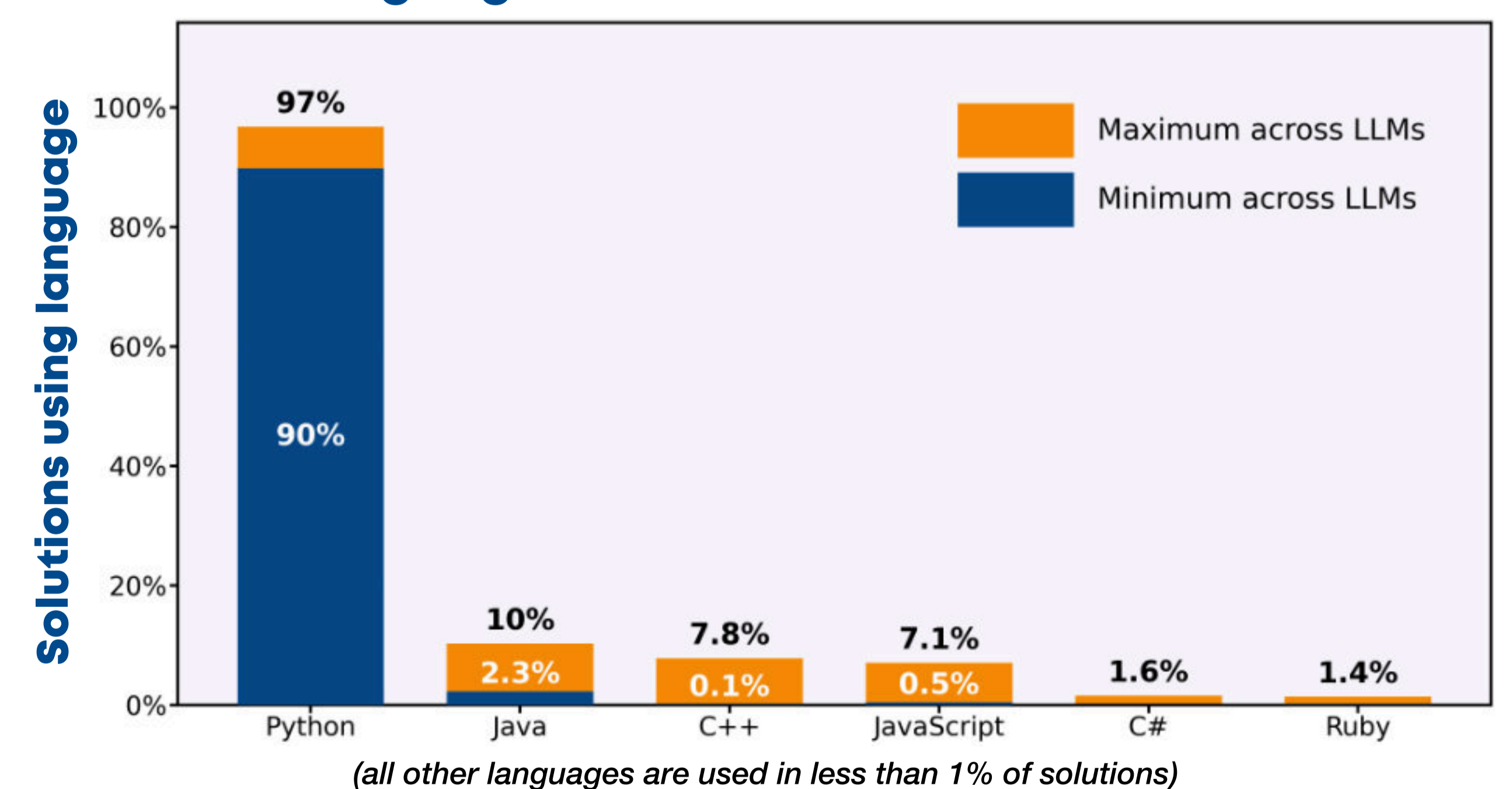
Language Findings

- *Python dominates* generated solutions across all models, accounting for 90–97% of benchmark task solutions.
- Even in performance-critical project prompts, Python remains the most-used language in 58% of cases.
- LLMs *consistently contradict their own recommendations* for new projects, rarely choosing C, C++, Go, or Rust.

Libraries imported for benchmark tasks



Languages used for benchmark tasks



1. Security threat.

Suboptimal defaults can steer production systems toward unsafe or inefficient implementation choices.



2. Ecosystem impact.

LLM preferences may reinforce dominant libraries, reducing visibility for emerging open-source tools.



3. Evaluation gap.

Code evaluation should measure whether technology choices fit the task, not just whether code runs.

