Customizing Deep Learning Models for Code-Related Tasks

Alessandro Giagnorio

ICSME'24
Flagstaff

DROP ME A LINE

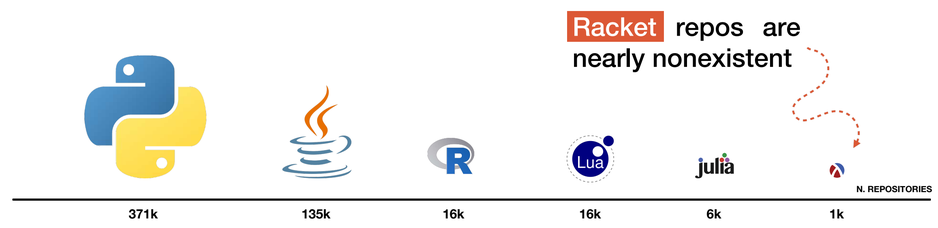
Current situation

LLMs have a GENERAL KNOWLEDGE of SE tasks

Modern LLMs are often evaluated on tasks that require a **general understanding** of programming. Although the performance of these models has improved over time, these results **do not extend** to their usage on **private projects**, where these models frequently fail to provide accurate recommendations. Indeed, LLMs are not tailored to the specific consumer who uses them and thus **lack knowledge** of the **developers' expertise** and the **projects** on which they are working.

LLMs works better on HIGH-RESOURCE languages

LLMs are typically trained on **publicly available** code, such as GitHub repositories, therefore favoring the **most popular programming** languages. The following graph compares the distribution of GitHub repositories with at least **10 stars** across different programming languages.

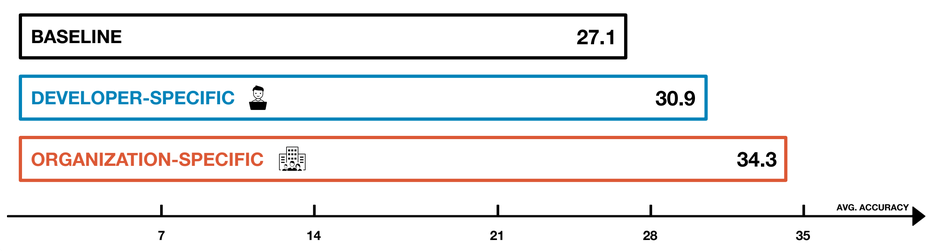


Our mission

DEVELOPER-DRIVEN CUSTOMIZATION

We **customized** LLMs on **code changes** written by a **single developer** or **all developers** of a given organization. To avoid data contamination, we first pre-trained and fine-tuned a deep learning model on general code completion samples. Then, we further trained the base model on **developer-specific** and **organization-specific** contributions to their projects.

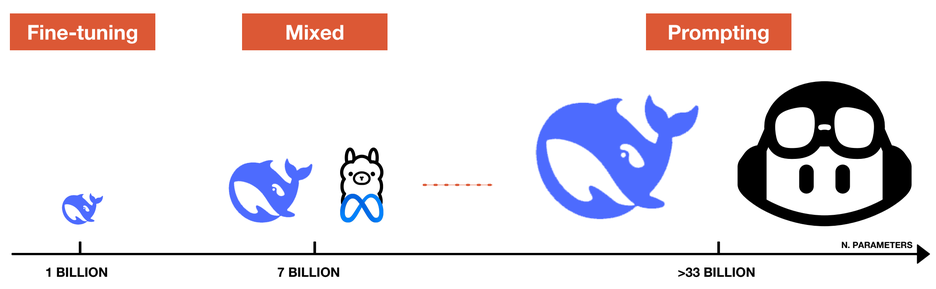
Our findings reveal that fine-tuning these models on **personal contributions** can significantly **improve** the accuracy of suggestions provided to the developers. Indeed, personalized models can infer recurring **implementation patterns** and **project-specific APIs** that general pre-trained model would not know.



CONTEXT-DRIVEN CUSTOMIZATION

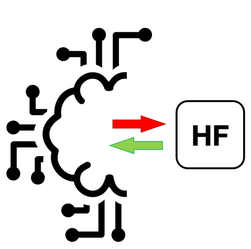
We started with investigating the **performance gap** between **high-** and **low-resource** languages on state-of-the-art code models. We observed that some niche languages, like R and Racket, perform significantly worse on the same task than high-resource languages.

Therefore, we evaluated several **fine-tuning** and **prompting techniques** to improve LLMs' code generation capabilities on these low-resource languages. Our findings reveal that **fine-tuning smaller models** (~1B parameters) on additional high-quality data **improves** code generation performance. Conversely, **larger models** (≥33B parameters) benefit more from including **extra features** into the instruction **prompt**.



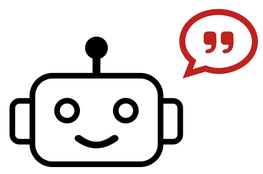
What's next

Aligning LLMs to developers through RLHF



Although fine-tuning can help to personalize LLMs, more **effective** and efficient techniques for **aligning** code models with developer knowledge and expertise should be investigated. To this end, we intend to explore **reinforcement learning** techniques, which are commonly used for aligning models in NLP tasks.

Better prompting techniques for improving LLMs



We observed that **in-context learning** techniques can significantly enhance the performance of LLMs when applied to **low-resource** programming languages. Therefore, we intend to pursue this path by developing **novel prompting techniques** for LLMs and investigating state-of-the-art methodologies, such as **Chain-of-Thought** prompting.

Università della Svizzera italiana
Software Institute
SEART
Funded by:
Schweizerischer Nationalfonds