ReplitLM: using Open-source from Training to Production for a Code Completion LLM

Michele Catasta

https://twitter.com/pirroh
https://pirroh.fyi
Code Completion on Replit

```css
/* container with centered text and sans-serif font */
.

/* Style H1 with font size of 24 */
/* button add padding on top and box shadow */
/* .quotes add margin and padding */
/* .quote font size of 18 */
/* .author font size of 12 and bold text */
```
In early May 2023 we released **replit-code-v1-3b**, our bespoke Code Completion LLM serving a large number of Replit users.

**Model Description**

`replit-code-v1-3b` is a 2.7B Causal Language Model focused on **Code Completion**.

The model has been trained on a subset of the [Stack Dedup v1.2 dataset](https://github.com/replit/replit-code-v1-3b).
First Llama-style LLM for code
~195 tokens per parameter

Trained on 525B tokens of code
175B tokens over 3 epochs

20 languages
Markdown, Java, JavaScript, Python, TypeScript, PHP, SQL, JSX, reStructuredText, Rust, C, CSS, Go, C++, HTML, Vue, Ruby, Jupyter Notebook, R, Shell
Pretraining data mixture based on The Stack v1.2 (released in March 2023)

- Selected the top 20 languages used on Replit
- Large number of code quality heuristics to filter the dataset (e.g., Codex paper, stripping long content from HTML/CSS files, etc.)
- Data processing on Spark, vocabulary training with Google SentencePiece
Scaling Data-Constrained Language Models

Published coincidentally just a few weeks after we released our LLM

Highly recommended paper, confirming our ablation studies on repeated data

This intuition allowed us to train to completion using only permissively-licensed code, hence we could release our model under CC BY-SA-4.0

Figure 1: Return and Allocation when repeating data. (Left): Loss of LLMs (4.2B parameters) scaled on repeated data decays predictably (§6). (Right): To maximize performance when repeating, our data-constrained scaling laws and empirical data suggest training smaller models for more epochs in contrast to what assuming Chinchilla scaling laws [42] hold for repeated data would predict (§5).
<table>
<thead>
<tr>
<th>2.7B parameters</th>
<th>256 A100-40GB GPUs</th>
<th>LLM best practices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom 32k vocabulary focused on code</td>
<td>For ~3 days on the MosaicML platform</td>
<td>Flash Attention, AliBi positional embeddings, LionW optimizer, etc.</td>
</tr>
</tbody>
</table>
All training runs based on an early release of **LLM Foundry** by MosaicML.

Same library used to train larger open-source models like MPT-7B and MPT-30B.
<table>
<thead>
<tr>
<th>Language</th>
<th>Framework</th>
<th>Score pass@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>OpenAI HumanEval</td>
<td>22.56%</td>
</tr>
<tr>
<td>Python</td>
<td>MultiPL-E</td>
<td>20.49%</td>
</tr>
<tr>
<td>Java</td>
<td>MultiPL-E</td>
<td>20.25%</td>
</tr>
<tr>
<td>JavaScript</td>
<td>MultiPL-E</td>
<td>19.25%</td>
</tr>
<tr>
<td>C++</td>
<td>MultiPL-E</td>
<td>18.63%</td>
</tr>
<tr>
<td>Rust</td>
<td>MultiPL-E</td>
<td>16.02%</td>
</tr>
<tr>
<td>PHP</td>
<td>MultiPL-E</td>
<td>13.04%</td>
</tr>
</tbody>
</table>
- To navigate the latest Code LLM releases, **BigCode** created **Multilingual Code Models Evaluation**

- Based on **MultiPL-E**, an extension of the original OpenAI HumanEval benchmark to 18 languages

- **replit-code-v1-3b** was trained only on 10 languages out of the 18 supported by MultiPL-E
<table>
<thead>
<tr>
<th>Further pretraining on 111B tokens of code</th>
<th>Code authored by our users in public Repls</th>
<th>Same languages, same data filtering heuristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>37B tokens over 3 epochs</td>
<td>A lot of Python and Javascript</td>
<td></td>
</tr>
</tbody>
</table>
The problem

Nowadays everybody finetune / continue train LLaMA. A practical problem is learning rate re-warm: the pretraining learning rate schedule stops at 3e-5, naively increasing the continue train lr to 3e-4 typically causes double descent. Is there a good way to mitigate this issue? 😐

11:09 AM · Aug 15, 2023 · 46K Views

The solution?

- **Continual Pre-Training of Large Language Models: How to (re)warm your model?**

- A pragmatic hack explained by Shital Shah in this thread, inspired by the LR schedule from "Scaling Vision Transformers"
<table>
<thead>
<tr>
<th>Language (Implementation)</th>
<th>Score pass@1</th>
<th>Base model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python (OpenAI HumanEval)</td>
<td>30.48%</td>
<td>22.56%</td>
</tr>
<tr>
<td>Python (MultiPL-E)</td>
<td>29.81%</td>
<td>20.49%</td>
</tr>
<tr>
<td>Java (MultiPL-E)</td>
<td>19.62%</td>
<td>20.25%</td>
</tr>
<tr>
<td>JavaScript (MultiPL-E)</td>
<td>27.95%</td>
<td>19.25%</td>
</tr>
<tr>
<td>C++ (MultiPL-E)</td>
<td>26.08%</td>
<td>18.63%</td>
</tr>
<tr>
<td>Rust (MultiPL-E)</td>
<td>15.38%</td>
<td>16.02%</td>
</tr>
<tr>
<td>PHP (MultiPL-E)</td>
<td>23.60%</td>
<td>13.04%</td>
</tr>
</tbody>
</table>
replit-***-v1-3b / Inference

~ 200 tokens / s on a single A100-40G (no batching)

We made explicit architectural choices to support:
- [https://github.com/NVIDIA/FasterTransformer](https://github.com/NVIDIA/FasterTransformer)
- [https://github.com/triton-inference-server](https://github.com/triton-inference-server)
for optimized inference on NVIDIA GPUs

Reliable inference evaluation across model architectures is still really HARD
Since the open-source release, a lot of interesting projects spun up from `replit-code-v1-3b`

Instruct fine tuned on CodeAlpaca and GPTeacher Code-Instruct:
[https://huggingface.co/teknium/Replit-v2-CodeInstruct-3B](https://huggingface.co/teknium/Replit-v2-CodeInstruct-3B)

Quantization + ggml support to boost local inference for VSCode plugins

@morph_labs allows you to use the recently released Replit GPT4All model on Apple Metal to perform privacy aware
- Code completion (23 tok/second)
- Chatting and asking questions

all through the Rift VSCode extension.

Local LLMs power the future of software development.
Links
https://github.com/replit/ReplitLM
https://huggingface.co/replit/replit-code-v1-3b
https://blog.replit.com/llm-training

Acknowledgements
- Madhav Singhal, Juan Sigler Priego, Bradley Heilbrun, Samip Dahal, Giuseppe Burtini, Reza Shabani, Amjad Masad & the whole Replit team
- Jonathan Frankle, Hanling Tang, Abhinav Venigalla, Vitaliy Chiley, Alexander Trott, Daya Khudia, Scott Sovine, Barry Dauber, Naveen Rao & the whole MosaicML team