A SYSTEMATIC EVALUATION OF LARGE LANGUAGE MODELS OF CODE

Frank F. Xu, Uri Alon, Graham Neubig, Vincent J. Hellendoorn School of Computer Science Carnegie Mellon University

Year of publication: May 2022
Number of citations:6

Overview

1. Introduction
2. Related Work
3. Evaluation Settings
4. Compared Models
5. Results
6. Conclusion

Introduction

Problem definition: current state-of-the-art code Large language models (LMs) are not publicly available

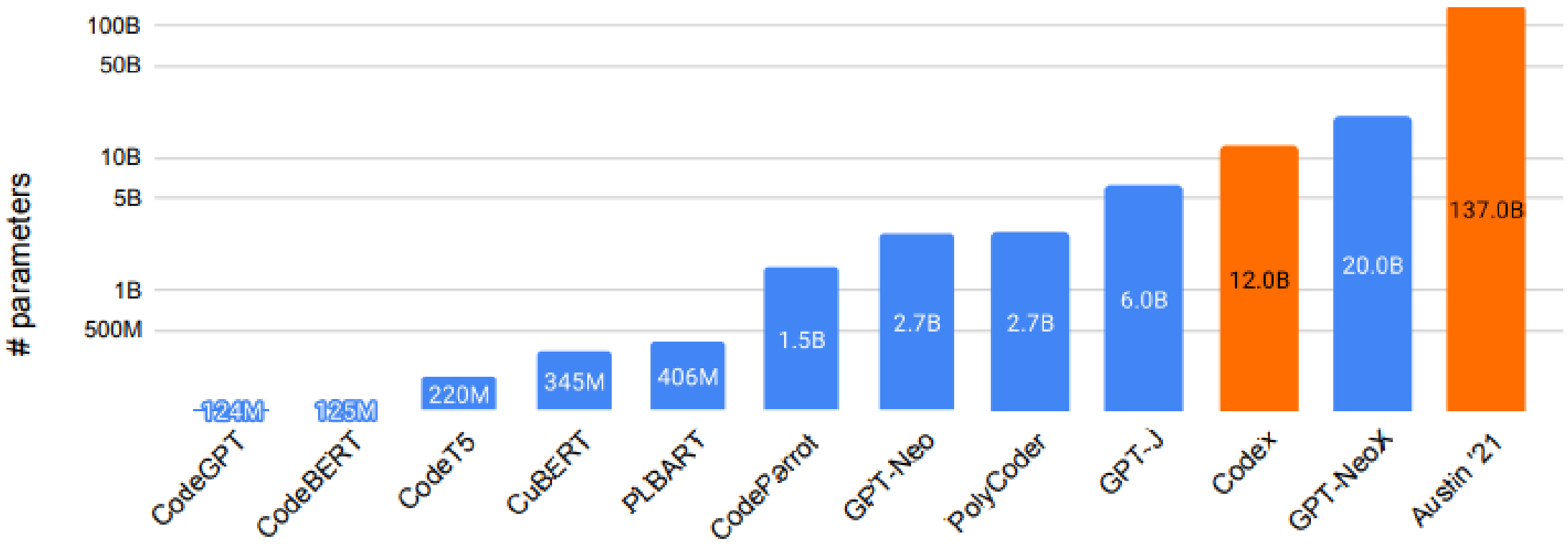
e.g: Codex provides non-free access to the model's output through black-box API calls, but the model's weights and training data are unavailable.

- Limits applications and research
- Incorporating additional mechanisms. E.g: Retrieval

PolyCoder:

- 160M - 2.7B parameters
- GPT-2 architecture
- 12 programming languages
- Outperforms all models for C programming language

Existing language models of code, their sizes and availability



Related work

- Pretraining Data

Model	Pretraining code
CodeParrot	Google BigQuery's GitHub dataset/ CodeSearchNet
CodeT5	Google BigQuery's GitHub dataset/ CodeSearchNet
GPT-Neo	Pile
GPT-J	Pile
Cordex	Unknown

Evaluation settings

- Extrinsic Evaluation - evaluate all models on the HumanEval dataset

- The dataset contains 164 prompts with descriptions in the form of code comments and function definitions, including argument names and function names, and test cases to judge whether the generated code is correct

- Intrinsic Evaluation - compute the perplexity for each language on an unseen set of GitHub repositories

- prevent training-to-test data leakage for models such as GPT-Neo and GPT-J, removed repositories that appeared in the GitHub portion of the Pile training dataset

PolyCoder

- **Raw Code Corpus Collection :** cloned the most popular repositories for 12 popular programming languages from GitHub in October 2021
- **Data Preprocessing**

	PolyCoder	CodeParrot	Codex
Dedup	Exact	Exact	Unclear, mentions “unique”
Filtering	Files > 1 MB, < 100 tokens	Files > 1MB, max line length > 1000, mean line length > 100, fraction of alphanumeric characters < 0.25, containing the word ”auto-generated” or similar in the first 5 lines	Files > 1MB, max line length > 1000, mean line length > 100, auto-generated (details unclear), contained small percentage of alphanumeric characters (details unclear)
Tokenization	Trained GPT-2 tokenizer on a random 5% subset (all languages)	Trained GPT-2 tokenizer on train split	GPT-3 tokenizer, add multi-whitespace tokens to reduce redundant whitespace tokens

- **Deduplication and Filtering:** Deduplication and Filtering Similarly to Codex and CodeParrot, very large (>1MB) and very short (<100 tokens) files were filtered out reducing dataset by 33%
- **TRAINING:**

- GPT-2 as the model architecture

- Train 3 different sized models with 2.7 billion, 400 million and 160 million parameters, as the largest 2.7B model being on par with GPT-Neo for fair comparison.

	PolyCoder (2.7B)	CodeParrot (1.5B)	Codex (12B)
Model Initialization	From scratch	From scratch	Initialized from GPT-3
NL Knowledge	Learned from comments in the code	Learned from comments in the code	Natural language knowledge from GPT-3
Learning Rate	1.6e-4	2.0e-4	1e-4
Optimizer	AdamW	AdamW	AdamW
Adam betas	0.9, 0.999	0.9, 0.999	0.9, 0.95
Adam eps	1e-8	1e-8	1e-8
Weight Decay	-	0.1	0.1
Warmup Steps	1600	750	175
Learning Rate Decay	Cosine	Cosine	Cosine
Batch Size (#tokens)	262K	524K	2M
Training Steps	150K steps, 39B tokens	50K steps, 26B tokens	100B tokens
Context Window	2048	1024	4096

Results

Model	Pass@1	Pass@10	Pass@100	Tokens Trained	Code Tokens	Python Tokens
PolyCoder (160M)	2.13%	3.35%	4.88%	39B	39B	2.5B
PolyCoder (400M)	2.96%	5.29%	11.59%	39B	39B	2.5B
PolyCoder (2.7B)	5.59%	9.84%	17.68%	39B	39B	2.5B
CodeParrot (110M)	3.80%	6.57%	12.78%	26B	26B	26B
CodeParrot (1.5B)	3.58%	8.03%	14.96%	26B	26B	26B
GPT-Neo (125M)	0.75%	1.88%	2.97%	300B	22.8B	3.1B
GPT-Neo (1.3B)	4.79%	7.47%	16.30%	380B	28.8B	3.9B
GPT-Neo (2.7B)	6.41%	11.27%	21.37%	420B	31.9B	4.3B
GPT-J (6B)	11.62%	15.74%	27.74%	402B	30.5B	4.1B
Codex (300M)	13.17%	20.37%	36.27%	100B*	100B*	100B*
Codex (2.5B)	21.36%	35.42%	59.50%	100B*	100B*	100B*
Codex (12B)	28.81%	46.81%	72.31%	100B*	100B*	100B*

*Codex is initialized with another pretrained model, GPT-3.

Conclusion

Systematic evaluation of large language models for code.

- The performance generally benefits from larger models and longer training time.
- The better results of GPT-Neo over PolyCoder in some languages show that training on natural language text and code can benefit the modeling of code.

Reference

Xu, Frank F., et al. "A systematic evaluation of large language models of code." *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*. 2022.

Thank you!

Any Questions?

You can find me at:
dinuja.21@cse.mrt.ac.lk