facebookresearch /
codellama

Code Issues 79 Pull requests 5 Actions Projects Security Insights

Inference code for CodeLlama models

View license

Code of conduct

Security policy

13.6k stars 1.4k forks 154 watching 4 Branches 0 Tags Activity Custom properties

Public repository

main 4 Branches 0 Tags Go to file t Go to file + Add file

jgehring 3 weeks ago

.circleci	Initial commit	6 months ago
llama	Updates for 70b release	3 weeks ago
.gitignore	Updates for 70b release	3 weeks ago
CODE_OF_CONDUCT.md	Initial commit	6 months ago
CONTRIBUTING.md	Initial commit	6 months ago
LICENSE	Initial commit	6 months ago
MODEL_CARD.md	Updates for 70b release	3 weeks ago
README.md	Updates for 70b release	3 weeks ago
USE_POLICY.md	Initial commit	6 months ago
dev-requirements.txt	Initial commit	6 months ago
download.sh	Updates for 70b release	3 weeks ago
example_completion.py	Updates for 70b release	3 weeks ago
example_infilling.py	Updates for 70b release	3 weeks ago
	Updates for 70b release	3 weeks ago

example_instructions.py		
requirements.txt	Updates for 70b release	3 weeks ago
setup.py	Updates for 70b release	3 weeks ago

Introducing Code Llama

Code Llama is a family of large language models for code based on Llama 2 providing state-of-the-art performance among open models, infilling capabilities, support for large input contexts, and zero-shot instruction following ability for programming tasks. We provide multiple flavors to cover a wide range of applications: foundation models (Code Llama), Python specializations (Code Llama - Python), and instruction-following models (Code Llama - Instruct) with 7B, 13B and 34B

README Code of conduct License Security

infilling based on surrounding content. Code Llama was developed by fine-tuning Llama 2 using a higher sampling of code. As with Llama 2, we applied considerable safety mitigations to the fine-tuned versions of the model. For detailed information on model training, architecture and parameters, evaluations, responsible AI and safety refer to our research paper. Output generated by code generation features of the Llama Materials, including Code Llama, may be subject to third party licenses, including, without limitation, open source licenses.

We are unlocking the power of large language models and our latest version of Code Llama is now accessible to individuals, creators, researchers and businesses of all sizes so that they can experiment, innovate and scale their ideas responsibly. This release includes model weights and starting code for pretrained and fine-tuned Llama language models — ranging from 7B to 34B parameters.

This repository is intended as a minimal example to load Code Llama models and run inference.

Download

In order to download the model weights and tokenizers, please visit the Meta website and accept our License.

Once your request is approved, you will receive a signed URL over email. Then run the download.sh script, passing the URL provided when prompted to start the download. Make sure that you copy the URL text itself, **do not use the 'Copy link address' option** when you right click the URL. If the copied URL text starts with: https://download.llamameta.net, you copied it correctly. If the copied URL text starts with: https://l.facebook.com, you copied it the wrong way.

Pre-requisites: make sure you have `wget` and `md5sum` installed. Then to run the script: `bash download.sh`.

Keep in mind that the links expire after 24 hours and a certain amount of downloads. If you start seeing errors such as `403: Forbidden`, you can always re-request a link.

Model sizes

Model	Size
7B	~12.55GB
13B	24GB
34B	63GB
70B	131GB

Setup

In a conda environment with PyTorch / CUDA available, clone the repo and run in the top-level directory:

```
pip install -e .
```

Inference

Different models require different model-parallel (MP) values:

Model	MP
7B	1
13B	2
34B	4
70B	8

All models, except the 70B python and instruct versions, support sequence lengths up to 100,000 tokens, but we pre-allocate the cache according to `max_seq_len` and `max_batch_size` values. So set those according to your hardware and use-case.

Pretrained Code Models

The Code Llama and Code Llama - Python models are not fine-tuned to follow instructions. They should be prompted so that the expected answer is the natural continuation of the prompt.

See `example_completion.py` for some examples. To illustrate, see command below to run it with the `CodeLlama-7b` model (`nproc_per_node` needs to be set to the `MP` value):

```
torchrun --nproc_per_node 1 example_completion.py \
    --ckpt_dir CodeLlama-7b/ \
    --tokenizer_path CodeLlama-7b/tokenizer.model \
    --max_seq_len 128 --max_batch_size 4
```

Pretrained code models are: the Code Llama models `CodeLlama-7b` , `CodeLlama-13b` , `CodeLlama-34b` , `CodeLlama-70b` and the Code Llama - Python models `CodeLlama-7b-Python` , `CodeLlama-13b-Python` , `CodeLlama-34b-Python` , `CodeLlama-70b-Python` .

Code Infilling

Code Llama and Code Llama - Instruct 7B and 13B models are capable of filling in code given the surrounding context.

See `example_infilling.py` for some examples. The `CodeLlama-7b` model can be run for infilling with the command below (`nproc_per_node` needs to be set to the `MP` value):

```
torchrun --nproc_per_node 1 example_infilling.py \
    --ckpt_dir CodeLlama-7b/ \
    --tokenizer_path CodeLlama-7b/tokenizer.model \
    --max_seq_len 192 --max_batch_size 4
```

Pretrained infilling models are: the Code Llama models `CodeLlama-7b` and `CodeLlama-13b` and the Code Llama - Instruct models `CodeLlama-7b-Instruct` , `CodeLlama-13b-Instruct` .

Fine-tuned Instruction Models

Code Llama - Instruct models are fine-tuned to follow instructions. To get the expected features and performance for the 7B, 13B and 34B variants, a specific formatting defined in `chat_completion()` needs to be followed, including the `INST` and `<<SYS>>` tags, `BOS` and `EOS` tokens, and the whitespaces and linebreaks in between (we recommend calling `strip()` on inputs to avoid double-spaces). `CodeLlama-70b-Instruct` requires a separate turn-based prompt format defined in `dialog_prompt_tokens()` . You can use `chat_completion()` directly to generate answers with all instruct models; it will automatically perform the required formatting.

You can also deploy additional classifiers for filtering out inputs and outputs that are deemed unsafe. See the llama-recipes repo for an example of how to add a safety checker to the inputs and outputs of your inference code.

Examples using `CodeLlama-7b-Instruct`:

```
torchrun --nproc_per_node 1 example_instructions.py \
    --ckpt_dir CodeLlama-7b-Instruct/ \
    --tokenizer_path CodeLlama-7b-Instruct/tokenizer.model \
    --max_seq_len 512 --max_batch_size 4
```

Fine-tuned instruction-following models are: the Code Llama - Instruct models `CodeLlama-7b-Instruct`, `CodeLlama-13b-Instruct`, `CodeLlama-34b-Instruct`, `CodeLlama-70b-Instruct`.

Code Llama is a new technology that carries potential risks with use. Testing conducted to date has not — and could not — cover all scenarios. In order to help developers address these risks, we have created the Responsible Use Guide. More details can be found in our research papers as well.

Issues

Please report any software “bug”, or other problems with the models through one of the following means:

- Reporting issues with the model: github.com/facebookresearch/codellama
- Reporting risky content generated by the model: developers.facebook.com/llama_output_feedback
- Reporting bugs and security concerns: facebook.com/whitehat/info

Model Card

See MODEL_CARD.md for the model card of Code Llama.

License

Our model and weights are licensed for both researchers and commercial entities, upholding the principles of openness. Our mission is to empower individuals, and industry through this opportunity, while fostering an environment of discovery and ethical AI advancements.

See the LICENSE file, as well as our accompanying Acceptable Use Policy

References

1. Code Llama Research Paper
2. Code Llama Blog Post