

---

## **gpt-2-simple**

```
gpt2.generate(sess.)
```

QUEEN ELIZABETH:

Uncle, thou hast made good supper; but I am  
Disgusted with thee. Go not home.

DUCHESS OF YORK:

Why, fie, fie, fie! Why, fie, fie!  
Thou hast made me requite my former hours of nap,  
In ordering the day, of all; fie, fie!  
Thou hast put on my gown, which I have untied up,  
With a little hoop; and, not to be seen,

A simple Python package that wraps existing model fine-tuning and generation scripts for OpenAI's GPT-2 text generation model (specifically the "small" 124M and "medium" 355M hyperparameter versions). Additionally, this package allows easier generation of text, generating to a file for easy curation, allowing for prefixes to force the text to start with a given phrase.

This package incorporates and makes minimal low-level changes to:

- Model management from OpenAI's official GPT-2 repo (MIT License)
- Model finetuning from Neil Shepperd's fork of GPT-2 (MIT License)
- Text generation output management from textgenrnn (MIT License / also created by me)

For finetuning, it is **strongly** recommended to use a GPU, although you can generate using a CPU (albeit much more slowly). If you are training in the cloud, using a Colaboratory notebook or a Google Compute Engine VM w/ the TensorFlow Deep Learning image is strongly recommended. (as the GPT-2 model is hosted on GCP)

You can use gpt-2-simple to retrain a model using a GPU **for free** in this Colaboratory notebook, which also demos additional features of the package.

Note: Development on gpt-2-simple has mostly been superceded by aitextgen, which has similar AI text generation capabilities with more efficient training time and resource usage. If you do not require using TensorFlow, I recommend using aitextgen instead. Checkpoints trained using gpt-2-simple can be loaded using aitextgen as well.

---

## Install

gpt-2-simple can be installed via PyPI:

```
1 pip3 install gpt-2-simple
```

You will also need to install the corresponding TensorFlow 2.X version (min 2.5.1) for your system (e.g. `tensorflow` or `tensorflow-gpu`).

## Usage

An example for downloading the model to the local system, finetuning it on a dataset. and generating some text.

Warning: the pretrained 124M model, and thus any finetuned model, is 500 MB! (the pretrained 355M model is 1.5 GB)

```
1 import gpt_2_simple as gpt2
2 import os
3 import requests
4
5 model_name = "124M"
6 if not os.path.isdir(os.path.join("models", model_name)):
7     print(f"Downloading {model_name} model...")
8     gpt2.download_gpt2(model_name=model_name)    # model is saved into
9         current directory under /models/124M/
10
11 file_name = "shakespeare.txt"
12 if not os.path.isfile(file_name):
13     url = "https://raw.githubusercontent.com/karpathy/char-rnn/master/
14         data/tinyshakespeare/input.txt"
15     data = requests.get(url)
16     with open(file_name, 'w') as f:
17         f.write(data.text)
18
19
20 sess = gpt2.start_tf_sess()
21 gpt2.finetune(sess,
22               file_name,
23               model_name=model_name,
24               steps=1000)    # steps is max number of training steps
25
26 gpt2.generate(sess)
```

The generated model checkpoints are by default in `/checkpoint/run1`. If you want to load a model from that folder and generate text from it:

---

```
1 import gpt_2_simple as gpt2
2
3 sess = gpt2.start_tf_sess()
4 gpt2.load_gpt2(sess)
5
6 gpt2.generate(sess)
```

As with `textgenrnn`, you can generate and save text for later use (e.g. an API or a bot) by using the `return_as_list` parameter.

```
1 single_text = gpt2.generate(sess, return_as_list=True)[0]
2 print(single_text)
```

You can pass a `run_name` parameter to `finetune` and `load_gpt2` if you want to store/load multiple models in a `checkpoint` folder.

There is also a command-line interface for both finetuning and generation with strong defaults for just running on a Cloud VM w/ GPU. For finetuning (which will also download the model if not present):

```
1 gpt_2_simple finetune shakespeare.txt
```

And for generation, which generates texts to files in a `gen` folder:

```
1 gpt_2_simple generate
```

Most of the same parameters available in the functions are available as CLI arguments, e.g.:

```
1 gpt_2_simple generate --temperature 1.0 --nsamples 20 --batch_size 20
  --length 50 --prefix "<|startoftext|>" --truncate "<|endoftext|>" --
  include_prefix False --nfiles 5
```

See below to see what some of the CLI arguments do.

NB: *Restart the Python session first* if you want to finetune on another dataset or load another model.

## Differences Between gpt-2-simple And Other Text Generation Utilities

The method GPT-2 uses to generate text is slightly different than those like other packages like `textgenrnn` (specifically, generating the full text sequence purely in the GPU and decoding it later), which cannot easily be fixed without hacking the underlying model code. As a result:

- In general, GPT-2 is better at maintaining context over its entire generation length, making it good for generating conversational text. The text is also generally grammatically correct, with proper capitalization and few typos.

- 
- The original GPT-2 model was trained on a very large variety of sources, allowing the model to incorporate idioms not seen in the input text.
  - GPT-2 can only generate a maximum of 1024 tokens per request (about 3-4 paragraphs of English text).
  - GPT-2 cannot stop early upon reaching a specific end token. (workaround: pass the `truncate` parameter to a `generate` function to only collect text until a specified end token. You may want to reduce `length` appropriately.)
  - Higher temperatures work better (e.g. 0.7 - 1.0) to generate more interesting text, while other frameworks work better between 0.2 - 0.5.
  - When finetuning GPT-2, it has no sense of the beginning or end of a document within a larger text. You'll need to use a bespoke character sequence to indicate the beginning and end of a document. Then while generating, you can specify a `prefix` targeting the beginning token sequences, and a `truncate` targeting the end token sequence. You can also set `include_prefix=False` to discard the prefix token while generating (e.g. if it's something unwanted like `<|startoftext|>`).
  - If you pass a single-column `.csv` file to `finetune()`, it will automatically parse the CSV into a format ideal for training with GPT-2 (including prepending `<|startoftext|>` and suffixing `<|endoftext|>` to every text document, so the `truncate` tricks above are helpful when generating output). This is necessary to handle both quotes and newlines in each text document correctly.
  - GPT-2 allows you to generate texts in parallel by setting a `batch_size` that is divisible into `nsamples`, resulting in much faster generation. Works very well with a GPU (can set `batch_size` up to 20 on Colaboratory's K80)!
  - Due to GPT-2's architecture, it scales up nicely with more powerful GPUs. For the 124M model, if you want to train for longer periods of time, GCP's P100 GPU is about 3x faster than a K80/T4 for only 3x the price, making it price-comparable (the V100 is about 1.5x faster than the P100 but about 2x the price). The P100 uses 100% of the GPU even with `batch_size=1`, and about 88% of the V100 GPU.
  - If you have a partially-trained GPT-2 model and want to continue finetuning it, you can set `overwrite=True` to `finetune`, which will continue training and remove the previous iteration of the model without creating a duplicate copy. This can be especially useful for transfer learning (e.g. heavily finetune GPT-2 on one dataset, then finetune on other dataset to get a "merging" of both datasets).
  - If your input text dataset is massive (>100 MB), you may want to preencode and compress the dataset using `gpt2.encode_dataset(file_path)`. The output is a compressed `.npz` file which will load much faster into the GPU for finetuning.
  - The 774M "large" model may support finetuning because it will cause modern GPUs to go out-of-memory (you may get lucky if you use a P100 GPU on Colaboratory). However, you can still

---

generate from the default pretrained model using `gpt2.load_gpt2(sess, model_name='774M')` and `gpt2.generate(sess, model_name='774M')`.

- The 1558M “extra large”, true model, may not work out-of-the-box with the GPU included with the Colaboratory Notebook. More testing is needed to identify optimal configurations for it.

## Interactive Apps Using gpt-2-simple

- gpt2-small — App using the default GPT-2 124M pretrained model
- gpt2-reddit — App to generate Reddit titles based on a specified subreddit and/or keyword(s)
- gpt2-mtg — App to generate Magic: The Gathering cards

## Text Generation Examples Using gpt-2-simple

- ResetEra — Generated video game forum discussions (GitHub w/ dumps)
- /r/legaladvice — Title generation (GitHub w/ dumps)
- Hacker News — Tens of thousands of generated Hacker News submission titles

## Maintainer/Creator

Max Woolf (@minimaxir)

*Max's open-source projects are supported by his Patreon. If you found this project helpful, any monetary contributions to the Patreon are appreciated and will be put to good creative use.*

## License

MIT

## Disclaimer

This repo has no affiliation or relationship with OpenAI.