
textgenrnn

```
[maxs-mbp:tweet-generator maxwoolf$ python3  
Python 3.6.4 (default, Jan 6 2018, 11:51:59)  
[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.39.2)] on darwin  
Type "help", "copyright", "credits" or "license" for more information.  
>>> █
```

Easily train your own text-generating neural network of any size and complexity on any text dataset with a few lines of code, or quickly train on a text using a pretrained model.

textgenrnn is a Python 3 module on top of Keras/TensorFlow for creating char-rnns, with many cool features:

- A modern neural network architecture which utilizes new techniques as attention-weighting and skip-embedding to accelerate training and improve model quality.
- Train on and generate text at either the character-level or word-level.
- Configure RNN size, the number of RNN layers, and whether to use bidirectional RNNs.
- Train on any generic input text file, including large files.
- Train models on a GPU and then use them to generate text with a CPU.
- Utilize a powerful CuDNN implementation of RNNs when trained on the GPU, which massively speeds up training time as opposed to typical LSTM implementations.
- Train the model using contextual labels, allowing it to learn faster and produce better results in some cases.

You can play with textgenrnn and train any text file with a GPU *for free* in this Colaboratory Notebook!

Read this blog post or watch this video for more information!

Examples

```
1 from textgenrnn import textgenrnn
2
3 textgen = textgenrnn()
4 textgen.generate()
```

```
1 [Spoiler] Anyone else find this post and their person that was a little
   more than I really like the Star Wars in the fire or health and
   posting a personal house of the 2016 Letter for the game in a report
   of my backyard.
```

The included model can easily be trained on new texts, and can generate appropriate text *even after a single pass of the input data*.

```
1 textgen.train_from_file('hacker_news_2000.txt', num_epochs=1)
2 textgen.generate()
```

```
1 Project State Project Firefox
```

The model weights are relatively small (2 MB on disk), and they can easily be saved and loaded into a new textgenrnn instance. As a result, you can play with models which have been trained on hundreds of passes through the data. (in fact, textgenrnn learns *so well* that you have to increase the temperature significantly for creative output!)

```
1 textgen_2 = textgenrnn('/weights/hacker_news.hdf5')
2 textgen_2.generate(3, temperature=1.0)
```

```
1 Why we got money "regular "alter
2
3 Urburg to Firefox acquires Nelf Multi Shamn
4
5 Kubernetes by 'Googles Bern
```

You can also train a new model, with support for word level embeddings and bidirectional RNN layers by adding `new_model=True` to any train function.

Interactive Mode

It's also possible to get involved in how the output unfolds, step by step. Interactive mode will suggest you the *top N* options for the next char/word, and allows you to pick one.

When running `textgenrnn` in the terminal, pass `interactive=True` and `top=N` to `generate`. `N` defaults to 3.

```
1 from textgenrnn import textgenrnn
2
3 textgen = textgenrnn()
4 textgen.generate(interactive=True, top_n=5)
```

```
>>> textgen.generate(interactive=True, top_n=5)
```

This can add a *human touch* to the output; it feels like you're the writer! (reference)

Usage

`textgenrnn` can be installed from pypi via `pip`:

```
1 pip3 install textgenrnn
```

For the latest `textgenrnn`, you must have a minimum TensorFlow version of 2.1.0.

You can view a demo of common features and model configuration options in this Jupyter Notebook.

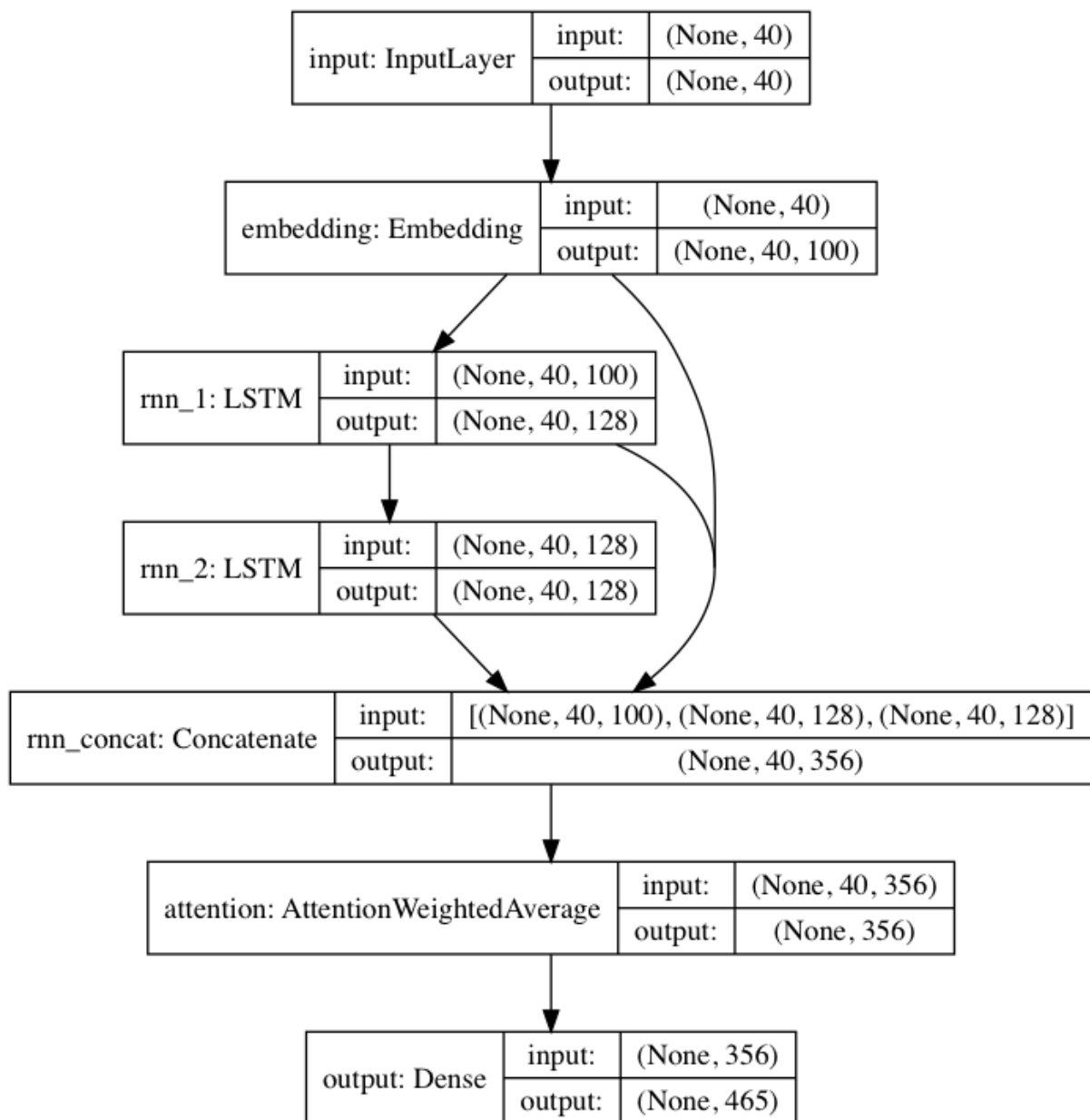
`/datasets` contains example datasets using Hacker News/Reddit data for training `textgenrnn`.

[/weights](#) contains further-pretrained models on the aforementioned datasets which can be loaded into textgenrnn.

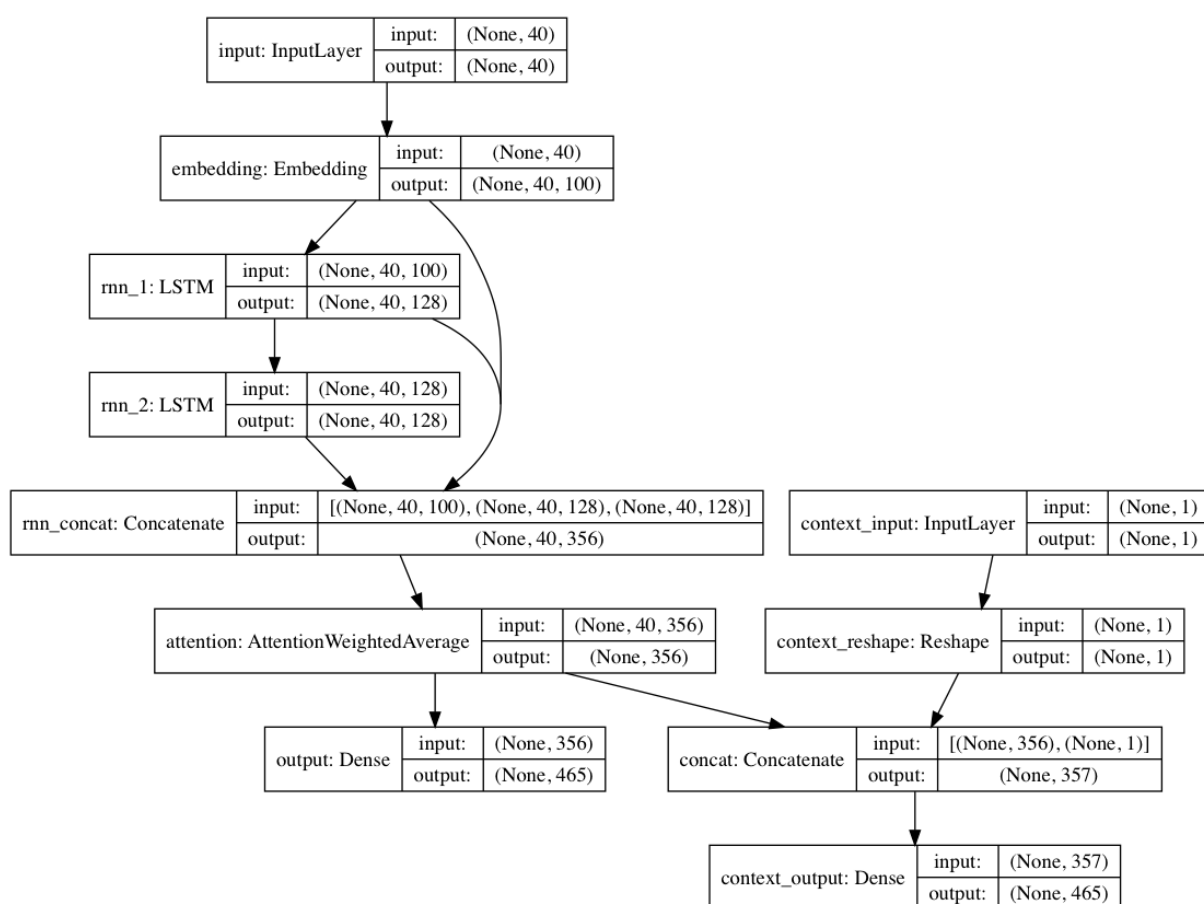
[/outputs](#) contains examples of text generated from the above pretrained models.

Neural Network Architecture and Implementation

textgenrnn is based off of the char-rnn project by Andrej Karpathy with a few modern optimizations, such as the ability to work with very small text sequences.



The included pretrained-model follows a neural network architecture inspired by DeepMoji. For the default model, textgenrnn takes in an input of up to 40 characters, converts each character to a 100-D character embedding vector, and feeds those into a 128-cell long-short-term-memory (LSTM) recurrent layer. Those outputs are then fed into *another* 128-cell LSTM. All three layers are then fed into an Attention layer to weight the most important temporal features and average them together (and since the embeddings + 1st LSTM are skip-connected into the attention layer, the model updates can backpropagate to them more easily and prevent vanishing gradients). That output is mapped to probabilities for up to 394 different characters that they are the next character in the sequence, including uppercase characters, lowercase, punctuation, and emoji. (if training a new model on a new dataset, all of the numeric parameters above can be configured)



Alternatively, if context labels are provided with each text document, the model can be trained in a contextual mode, where the model learns the text *given the context* so the recurrent layers learn the *decontextualized* language. The text-only path can piggy-back off the decontextualized layers; in all, this results in much faster training and better quantitative and qualitative model performance than just training the model given the text alone.

The model weights included with the package are trained on hundreds of thousands of text docu-

ments from Reddit submissions (via BigQuery), from a very *diverse* variety of subreddits. The network was also trained using the decontextual approach noted above in order to both improve training performance and mitigate authorial bias.

When fine-tuning the model on a new dataset of texts using `textgenrnn`, all layers are retrained. However, since the original pretrained network has a much more robust “knowledge” initially, the new `textgenrnn` trains faster and more accurately in the end, and can potentially learn new relationships not present in the original dataset (e.g. the pretrained character embeddings include the context for the character for all possible types of modern internet grammar).

Additionally, the retraining is done with a momentum-based optimizer and a linearly decaying learning rate, both of which prevent exploding gradients and makes it much less likely that the model diverges after training for a long time.

Notes

- **You will not get quality generated text 100% of the time**, even with a heavily-trained neural network. That’s the primary reason viral blog posts/Twitter tweets utilizing NN text generation often generate lots of texts and curate/edit the best ones afterward.
- **Results will vary greatly between datasets.** Because the pretrained neural network is relatively small, it cannot store as much data as RNNs typically flaunted in blog posts. For best results, use a dataset with at least 2,000-5,000 documents. If a dataset is smaller, you’ll need to train it for longer by setting `num_epochs` higher when calling a training method and/or training a new model from scratch. Even then, there is currently no good heuristic for determining a “good” model.
- A GPU is not required to retrain `textgenrnn`, but it will take much longer to train on a CPU. If you do use a GPU, I recommend increasing the `batch_size` parameter for better hardware utilization.

Future Plans for `textgenrnn`

- More formal documentation
- A web-based implementation using `tensorflow.js` (works especially well due to the network’s small size)
- A way to visualize the attention-layer outputs to see how the network “learns.”
- A mode to allow the model architecture to be used for chatbot conversations (may be released as a separate project)

-
- More depth toward context (positional context + allowing multiple context labels)
 - A larger pretrained network which can accommodate longer character sequences and a more indepth understanding of language, creating better generated sentences.
 - Hierarchical softmax activation for word-level models (once Keras has good support for it).
 - FP16 for superfast training on Volta/TPUs (once Keras has good support for it).

Articles/Projects using textgenrnn

Articles

- Lifehacker: How to Train Your Own Neural Network by Beth Skwarecki
- New York Times: Let Our Algorithm Choose Your Halloween Costume by Janelle Shane
- CNN Business: This quirky experiment highlights AI's biggest challenges by Rachel Metz

Projects

- Tweet Generator — Train a neural network optimized for generating tweets based off of any number of Twitter users
- Hacker News Simulator — Twitter bot trained on 300,000+ Hacker News submissions using textgenrnn.
- SubredditRNN — Reddit Subreddit where all submitted content is from textgenrnn bots.
- Human-AI Collaborated Pizzas — Pizza recepies generated with textgenrnn and made in real life.
- Board Game Titles
- Video Game Discussion Forum Titles
- A.I Created Cakes
- AI Created Cookies
- AI Generated Songs

Tweets

- BuzzFeed YouTube Videos
- AWS Services
- Recipes + D&D Spells + Heavy Metal Names
- RPG Adventure Names
- The Onion + Cosmopolitan
- Google Conference Room Names
- Sith Lords

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.

Credits

Andrej Karpathy for the original proposal of the char-rnn via the blog post The Unreasonable Effectiveness of Recurrent Neural Networks.

Daniel Grijalva for contributing an interactive mode.

License

MIT

Attention-layer code used from DeepMojji (MIT Licensed)