Comic Strip Generation using GPT2

Gayathri Pulagam, Rishitha Bandi, Surabhi Govil

SJSU Davidson College of Engineering Extended Studies, San Jose State University
San Jose, CA, United States

gayathri.pulagam@sjsu.edu
rishitha.bandi@sjsu.edu
surabhi.govil@sjsu.edu

***Abstract*—Text generation has become a topic of heavy research in the last decade. The aim of text generation is for the model to generate text which is indistinguishable to human-written text. In the early research of text generation, markov processes or deep generative LSTMs were used. More research in the area resulted in the development of more powerful text generation models like Open AI's GPT, BERT and other GAN-based models.**

***Keywords*— text generation, GPT2, transformers, language models, mlops, vertex ai**

I. Introduction

Transfer learning, particularly models like Allen AI's ELMO, OpenAI's Open-GPT, and Google's BERT allowed researchers to smash multiple benchmarks with minimal task-specific fine-tuning and provided the rest of the NLP community with pretrained models that could easily (with less data and less compute time) be fine-tuned and implemented to produce state of the art results.

GPT-2 is a state-of-the-art language model developed by Open-AI. Its main application is synthetic text generation. It has pushed the boundary for what is algorithmically possible which is exemplified in the table below. The text generated is coherent over a long horizon, and grammatical syntax and punctuation are near-perfect.

In the midst of what is truly a golden era in NLP, OpenAI's GPT-2 has remodeled the way we work with text data. Where ULMFiT and Google's BERT eased opened the doors for NLP enthusiasts, GPT-2 has smashed it in and made it so much easier to work on NLP tasks – primarily text generation. Fine tuning allows for quicker development. Requirement for less data and better results.

Most RNN's use character tokens or word tokens thus compressing the input and adding randomness to the final generated length.

GPT2 on the other hand uses what is called Byte ***Pair Encoding(BPE)*** which is sub word encoding. This is especially useful in text generation because it takes care of not treating different forms of word as different. (e.g. greatest will be treated as two tokens: 'great' and 'est' which is advantageous since it retains the similarity between great and greatest, while 'greatest' has another token 'est' added which makes it different). Also, it is not as low level as character-level encoding, which doesn't retain any value of a particular word.

The pre-trained GPT2 model can be fine-tuned on specific datasets, for example, to "acquire" the style of a dataset or learn to classify documents. This is done via Transfer Learning, which can be defined as "a means to extract knowledge from a source setting and apply it to a different target setting". Deep Daze is a library that can create large paragraphs of text, find interesting keywords from that and generate images from those pieces of text.

In this project, we used GPT-2 as it is better at maintaining context over its entire generation length, making it good for generating conversational text. For continuous monitoring and deployment code and to create a scalable environment we integrated MLOps using Vertex AI to train and serve our results.

II. Related Work

Deep neural networks and their training has become a real breakthrough in solving basic AI problems, including NLP [1].

The neural text generation problem is analyzed in many works, for example [1]. Authors describe the classic and recently proposed neural text generation models

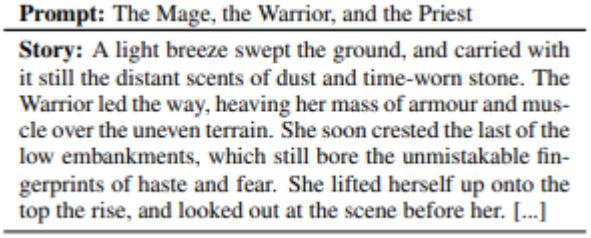
In 2017, a new simple neural network architecture was proposed, called transformer, based solely on the attention mechanism, without recurrence, i.e. sequential calculations [2]. Transfer learning technology allows you to retrain ready-made models. Today, this technology is the most promising in deep learning and is used in the most advanced neural models for the generation of natural language texts [3]. The next step was to demonstrate that language models begin to learn NLP tasks without any explicit supervision when trained on a new dataset of millions of web pages called WebText [4, 5]. Authors are researchers from OpenAI and they demonstrate their largest model GPT-2 (Generative Pre-Trained Transformer). This is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting but still underfits WebText.

Authors are researchers from OpenAI and they demonstrate their largest model GPT-2 (Generative Pre-Trained Transformer). This is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested languages modeling datasets in a zero-shot setting but still underfits WebText.

Unlike BERT, another popular generative pretrained transformer GPT-2 is created for generating samples of synthetic text with a completely logical narrative, if you give it any beginning [3]

III. Dataset

For our project, we leveraged the WritingPrompts dataset published by Facebook AI Research. WritingPrompts has a collection of 300K human-written stories paired with writing prompts collected from various online forums like Reddit. The WritingPrompts a dataset was published by Facebook AI Research intending to make machines able to compose a long and consistent story.



Prompt: The Mage, the Warrior, and the Priest

Story: A light breeze swept the ground, and carried with it still the distant scents of dust and time-worn stone. The Warrior led the way, heaving her mass of armour and muscle over the uneven terrain. She soon crested the last of the low embankments, which still bore the unmistakable fingerprints of haste and fear. She lifted herself up onto the top the rise, and looked out at the scene before her. [...]

Fig1: Example of writing prompt and the corresponding beginning of the story from dataset[8]

This dataset is of particular interest as it lets generate hierarchical stories that are first generating a premise and then transforming it into a paragraph of text.

Building upon the prompt or premise makes it easier to generate sequential stories because they provide base for the overall plot. It also reduces the tendency of standard sequence models to drift off topic.

IV. Methodologies

A. Fine Tune GPT2

Architecture of GPT2 is based on the very famous transformer model proposed by Google's paper "attention is all you need".

At each step, the model consumes the previously generated symbols as additional input when generating the next output.

"GPT-2 achieves state-of-the-art scores on a variety of domain-specific language modeling tasks. Our model is not trained on any of the data specific to any of these tasks and is only evaluated on them as a final test; this is known as the "zero-shot" setting. GPT-2 outperforms models trained on domain-specific data sets

(e.g. Wikipedia, news, books) when evaluated on those same data sets." – Open AI team.

GPT2 can generate large text from just prompts of input text also. For this project we used the GPT2 medium model with 345 million parameters.

B. Generate Images From Text

To generate images from the stories, we trained a model using Open AI's CLIP and Siren loosely inspired by an image-generating library called Deep Daze. Deep daze is capable of generating images for a given word. We trained an image generation model using CLIP and Siren and wrote a simple generate function to generate images from the generated story.

To generate the images for a particular story, the output from our fine-tuned GPT-2 model is fed into the image generator which is an imagine function based on the deep-daze library. By using a *create story* flag in the imagine function, images can be generated for longer sequence sentences.

V. Experimentation

A. *Story Generation:*

The fine tuning of GPT2 language model to generate story can be broken down into the following:

- Tokenize the test dataset with GPT-2 tokenizer.
- Load a pre-trained GPT-2 model.
- Evaluate the perplexity of the model on the validation dataset and record results.

We used the medium-sized 355M version of GPT-2 because it was large enough to handle the data set but small enough to run on Google Colab's cloud servers. We trained both models with 19,525 steps.

Along with the prompt, the temperature is one of the most important settings on how the model output is affected. The temperature controls how much randomness is in the output. In general, the lower the temperature, the more likely GPT-2 will choose words with a higher probability of occurrence. It is particularly useful when we want GPT-2 to generate ideas or complete a story as in our case. A higher temperature value brings more variety into the model output. Thus, we set a temperature of "1" here.

Another interesting thing used here is the **top p filtering** which helps the model sort the word probabilities in descending order. Then, it will sum those probabilities up to p while dropping the other words. This means the model not only keeps the best one but the most relevant word probabilities, as more than one word can be appropriate given a sequence.

According to this article on Perplexity by Wikipedia[9] the perplexity of a discrete distribution can be given as

$$2^{-\sum_x p(x) \log_2 p(x)}$$

Equation 1

which could also be written as:

$$\exp\left(\sum_x p(x) \log_e \frac{1}{p(x)}\right)$$

Equation 2

i.e. as a weighted geometric average of the inverses of the probabilities. For a continuous distribution, the sum would turn into an integral.

Thus, the metrics used to gauge the correctness of the generated sequence was the perplexity score. It indicates how well the model predicts a word. The lower the score the better is the model at predicting.

We were able to achieve a perplexity score of 24.03 after fine tuning the model.

```
***** Running training *****
  Total_num_training_step = 7810
  Num Epochs = 2
  Train_batch_size = 4
  Valid_batch_size = 4
Start epoch1 of 2
(batch loss=2.84406): 100%|██████████| 3905/3905 [23:09<00:00,  2.81it/s]
Average train loss per example=3.285960201112012 in epoch1
Starting evaluate after epoch 1
eval: 100%|██████████| 3785/3785 [07:34<00:00,  8.33it/s]
Average valid loss per example=3.1904303497178392 in epoch1
Perplextiy for valid dataset in epoch1 is 24.298882210433185
Start epoch2 of 2
(batch loss=2.75639): 100%|██████████| 3905/3905 [23:08<00:00,  2.81it/s]
Average train loss per example=3.101484777741182 in epoch2
Starting evaluate after epoch 2
eval: 100%|██████████| 3785/3785 [07:34<00:00,  8.33it/s]Average valid loss p
Perplextiy for valid dataset in epoch2 is 24.031594486782744
```

Fig 2: Perplexity scores after training the model for 2 epochs

B. *Image Generation*

For image generation we experimented with DALL-E pytorch implementation. We realized that we needed data which could associate with the story generated by our GPT-2 to produce relevant images.

We trained a model by using deep-daze as a reference using Open AI's CLIP and Siren. After training the model, we wrote a simple image generation function. The generated story, when passed into the *generate_images* function, generates images corresponding to the story.

We then experimented with deep-daze which is a library meant for generating animated images. We used their Imagine function with a *create_story* flag set to *True* to generate images for our generated stories. These generated images are saved in the folder and can be put together to create an animated story video

Fig3: An example of generated image

C. *MLOps Using Vertex AI*

Successful deployment and effective operations are the key to a successful AI model.

Vertex AI platform was used to manage the data, prototype, experiment, deploy, interpret models, and monitor it. Initial experiment with a smaller number of epochs was conducted in Google Colab environment and then at scale was done in Google Cloud platform.

Fig4: MLOps lifecycle [10]

As the implementation is pytorch and needs support libraries like transformers for GPT2, a pytorch jupyter colab is created and launched from the google console.

For training the GPT2 model on vertex AI pre-build pytorch container image is used with providing the required parameters like output directory, dataset bucket location, training code as python package etc.

On launching the training job from the jupyter notebook the model artifacts are saved in the cloud storage location provided while training. The model artifacts in the storage bucket have to be uploaded to the vertex AI to use it for the endpoint.

For uploading the model to the vertex AI a deploy container image is needed for inference.

But vertex AI doesn't have pre built container images without GPU. Torch model archiver is used to convert model artifacts to the mar extension file and serve the pytorch model using torchserve. Create a container image and push it to the container registry to use it for deploying the model. Docker is used to build the image and push to the container registry.

The model uploaded to the vertex AI is deployed to the endpoint created. With the endpoint the inference to the model is achieved.

VI. Results and Evaluations

Perplexity, a standard evaluation metric for language models, was used here to evaluate the text generation model. Perplexity is the inverse probability of a test set which is normalised by the number of words in the test set. Language models with lower perplexity are considered to be better than ones with higher perplexity.

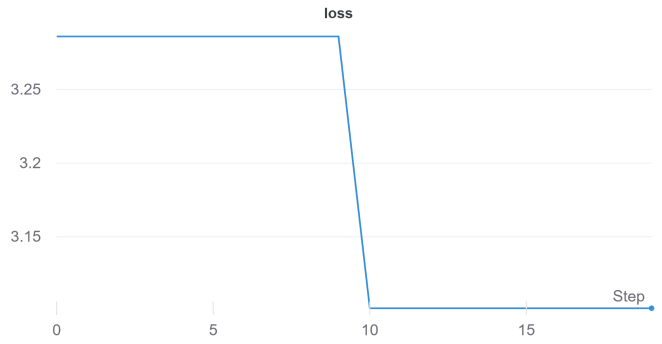


Fig 5: Wandb integration results

VII. Conclusions

We were able to quickly fine-tune a GPT2 model, prototype a fun application, and deploy it. The generated stories can further be improved by using more advanced pre-trained models, decoding methods, or even structured language prediction.

One of the major challenges in story generation is the inefficiency of standard recurrent architectures to model long documents with stories containing 734 words on average in a dataset. That is why we decided to use the 774M parameter GPT-2 language model to efficiently generate long form sentences.

Acknowledgment

We would like to express our gratitude to Professor Vijay Eranti for his guidance throughout the course of this project.

References

[1] D. Foster, Generative Deep Learning. Teaching Machines to Paint, Write, Compose and Play, O'Reilly Media, Inc., USA, 2019.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, I. Polosukhin. Attention Is All You Need, in: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA, pages 6000–6010.

[3] D. Rothman, Transformers for Natural Language Processing: Build innovative deep neural network architectures for NLP with Python, PyTorch. BERT, RoBERTa, T5, GPT-2, architecture of GPT-3, and much more, Packt Publishing Ltd, Birmingham, UK, 2021.

[4] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, Improving language understanding with unsupervised learning. Technical report, OpenAI, 2018.

[5] A. Radford, J. Wy, R. Child, D. Luan, D. Amodei, I. Sutkever. Language Models are Unsupervised Multitask Learners, Computer Science, 2019.

[6] S. Golovanov, R Kurbanov, S. Nikolenko, K. Truskovskyi, A. Tselousov, T. Wolf, Large Scale Transfer Learning for Narural Language Generation, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, july 28 – August 2, 2019, pp. 6053 – 6058.

[7] P. Budzianowski, I. Vulic, Hello, It's GPT-2 – How Can I Help You? Towards the Use of Pretrained Language Models for Task-Oriented Dialogue Systems, arXiv preprint arXiv: 1907.05774v2 [cs.CL] 4 Aug 2019.

[8] https://arxiv.org/pdf/1805.04833.pdf

[9] http://en.wikipedia.org/wiki/Perplexity

[10] https://cloud.google.com/blog/products/ai-machine-learning/google-cloud-launches-vertex-ai-unified-platform-for-mlops