Generating Narrative Text

Objectives

For our flagship project, we attempted to build a computer model that can generate a coherent passage of narrative text. Narrative is one of the important challenges of artificial intelligence. Although this project is not cutting-edge, it represented a unique opportunity for us: we had our own body of text to use for fine tuning due to our company's background as a publisher. We had two objectives:

> **1)** To gain a clearer theoretical understanding of how a computer model learns from a body text
>
> **2)** As a way of examining the nuts and bolts of the code to accomplish text generation

GPT-2

We built our model using the pretrained neural network GPT-2 from OpenAI. This is a 12-layer Decoder only Transformer-based model that can be fine-tuned for multiple NLP tasks. GPT-2 has been pretrained on approximately 40 GB of data.

The core of GPT-2 consists of twelve Decoder Blocks, with every Decoder Block comprised of:

> **1)** Twelve Attention Layers, each of which computes an output embedding (the attention value) as the weighted sum of the value vectors of the current *and* previous words; the word output embeddings of all the layers are then concatenated together into a single output embedding for that word.
>
> **2)** A Feed Forward Layer, which transforms the output embeddings from the Attention Layers into the final output word embeddings for that Decoder Block.

How is the probability calculated?

Rather than simply picking the next word with the highest probability, we use sampling to make sure we get a new sentence with each generation (i.e., no degenerate repetitions of common phrases).

The model article is configured to use **Nucleus Sampling** with $p = 0.95$. In calculating each successive word, we take a random sampling of all possible next words, then repeat the process for the next generation. However, at each stage, we take a random sample of the possible next words, but ignoring all words whose probability is below a threshold of .05 ($1 - p$).

Fine-Tuning Dataset

We fine-tune the model by feeding the pretrained model with our own dataset and comparing the model's estimated probability distributions for the dataset to the target probability distributions, computing the cross entropy loss, and then using the loss to update the weights.

For fine-tuning, we used a dataset built from books and magazines that we had published over the past 13 years. We provided data as ePubs, Word files, and even old InDesign files on CD-Roms. After preprocessing the data, there were a total of 1,511,329 words in the dataset. We split the dataset in a ratio of roughly 9:1 training set and testing set.

Baseline Training

As a baseline, the model was trained for one epoch (processing all the fine-tuning data a single time), which took 30 minutes on Google Cloud Platform. Some examples of the output are below.

> Seed sentence**:** There is a blue curtain.
>
> **=== GENERATED SEQUENCE 1 ===**
> There is a blue curtain. This to me appears like a piano, except with two strings. It is so unlike a prison or an upper court that we couldn't bear hearing it—I swear I did not notice.
> "What?" asked Peridot timidly.
>
> **=== GENERATED SEQUENCE 2 ===**
> There is a blue curtain. A shrine to departed warriors," she says. "They are told, if they don't repent, that these are crimes committed by people who wish to remain." Unlike fellow musicians, she still attempts to write about pop culture. "That's the attitude
>
> **=== GENERATED SEQUENCE 3 ===**
> There is a blue curtain. It goes up to the window, and leads to a forest on the outskirts of town. People are turning their heads to look at the horizon.
> "We come here from a foreign city. This is probably their location," Edward said. "As

Fine-Tuning with Additional Training

As we undertake additional fine-tuning, it is hard to judge whether a generated result is "good" or not, so we can only fine-tune against the perplexity.

Perplexity is a metric for measuring a model's performance in language modeling; the lower the score the better. Perplexity is an intrinsic evaluation, measuring within the model itself how well it predicts the next word. When we give the model an unseen test set and assign a probability to each word, the best model is the one that assigns the highest probability to the word that actually comes next ("Oh, I knew that word was coming, so I assigned it a very high probability!") Perplexity is normalized by the number of words, which enables us to compare test sets of different lengths – the longer the sentence, the less probable it is going to be. After training on the test set for 10 epochs, the perplexity was 25.9.

Fine-Tuning on Hyperparameters

As we are using a pre-trained model, there are not many hyperparameters we can adjust for fine-tuning, otherwise we'd have to do the training from scratch.

These are the hyperparameters we could try:

- num of epochs
- learning rate
- weight decay
- adam epsilon
- base model (right now we are using gpt2-small, we could try gpt2-medium as well)

  small model: 124 million parameters

  medium model: 355 million parameters

  large model: 774 million parameters

  largest model: 1.5 **billion** parameters

We then did the fine-tuning script (this is just the Python code—because the python interpreter runs against a text file containing the python code, and the python code can run anywhere there is a python interpreter, python is known as a scripting language).

We tried different combinations to fine-tune the GPT-2 small model, using the following parameters:

1. **learning_rate**: loguniform(5e-6, 5e-2)
   Deep learning neural networks are trained using the stochastic gradient descent algorithm, an optimization algorithm that estimates the error gradient for the current state of the model, then updates the weights of the model using backpropagation. The amount that the weights are updated during training is referred to as the step size or the learning rate. Generally, a large learning rate allows the model to learn faster, but at the cost of arriving on a sub-optimal final set of weights. In fine-tuning, we try using a smaller learning rate to train the network. Since we expect the pretrained weights to be quite good already as compared to randomly initialized weights, we do not want to distort them too quickly and too much.

2. **weight_decay**: uniform(scale=0.3)
   When training neural networks, it is common to use weight decay, where after each update, the weights are multiplied by a factor slightly less than 1. This prevents the weights from growing too large, and can be seen as gradient descent on a quadratic regularization term. Weight decay helps reduce the overfitting of the model on the training data and improve its performance on new data.

3. **adam_epsilon**: loguniform(1e-10, 1e-2)
   Adam (Adaptive Moment Estimation) is an “adaptive” learning rate. It uses smaller updates (a lower learning rate) for parameters associated with frequently occurring features and larger updates for parameters associated with infrequent features. Adam averages the current gradient with two other extensions of stochastic gradient descent: 1) an exponentially decaying average of all past gradients; and 2) an exponentially decaying average of past squared results. It behaves like a ball rolling down a slope but with some friction.

   As a step-size method, Adam has a stability parameter, **epsilon**, that increases the numerical stability of the method by ensuring the estimate of the variance is always above a certain level and avoiding a “divide by zero” error, while updating the variable when the gradient is almost zero. So ideally epsilon should be a small value and by default, the epsilon value is set to 1e-8. But having a small epsilon in the denominator will make larger weight updates. When you train with an epsilon that is too small, the optimizer will become unstable, with subsequent normalization larger weights will always be normalized to 1. Epsilon has a regulatory effect on the variance of the learning rate and a well-tuned epsilon can help in many settings where the learning trajectory is unstable.

4. **max_grad_norm**: uniform(loc=0.9, scale=0.2)
   During pre-training the input sentences are sampled from large-scale raw corpora, so some dimensions may have non-zero means or very large range of values. In the fine-tuning stage, these large embedding values may dominate the training process or lead to unstable gradients of model parameters in backpropagation. These can cause two widely known issues with training recurrent neural networks: the vanishing and the exploding gradient problems.

   One simple mechanism to deal with a sudden increase in the norm of the gradients is to rescale them whenever they go over a threshold. Gradient scaling involves normalizing the error gradient vector such that vector norm (magnitude) equals a defined value, such as 1.0. Gradient norm scaling involves changing the derivatives of the loss function to have a given vector norm when the L2 vector norm (sum of the squared values) of the gradient vector exceeds a threshold value. For example, we could specify a norm of 1.0, meaning that if the vector norm for a gradient exceeds 1.0, then the values in the vector will be rescaled so that the norm of the vector equals 1.0.

5. **num_train_epochs** with option [1,2]
   How many times the model processes the entire fine-tuning dataset.

In each iteration, we followed the following process:

1. Sample values for each parameter to make a config and run the fine-tuning. In other words, to take a random sample value from the range indicated for each fine-tuning feature. A config is a configuration file used to store configuration parameters, meaning initial parameters and settings for a model.

2. record the perplexity of each fine-tuning result

3. keep the best **n** fine-tuned models

As the fine-tuning script was running, it would upload to the testing website every model that registered a perplexity of less than 30. There were 500 iterations, each of which took between 3-6 minutes. Of the 500 iterations, 123 had a perplexity score of less than 30. The five models with the best perplexity scores were:

1_22: perplexity 19.6959
adam_epsilon: 8.65285266628266e-06
learning_rate: 0.00010128604458393196
max_grad_norm: 1.0054116204515218
num_train_epochs: 2
weight_decay: 0.21271275659785363

1_68: perplexity 19.7474
adam_epsilon: 5.772665718332016e-08
learning_rate: 3.973103106132757e-05
max_grad_norm: 1.0185023737531593

num_train_epochs: 2
weight_decay: 0.041784015110818015

1_9: perplexity 19.7593
adam_epsilon: 1.927905942121954e-05
learning_rate: 7.81539951047547e-05
max_grad_norm: 0.9892269012172212
num_train_epochs: 2
weight_decay: 0.030967801973292606

1_13: perplexity 19.7968
adam_epsilon: 1.2241421076749222e-08
learning_rate: 4.197896175488769e-05
max_grad_norm: 1.0066896981210351
num_train_epochs: 2
weight_decay: 0.20832004731832351

1_4: perplexity 19.8100
adam_epsilon: 1.3276864065059191e-09
learning_rate: 3.1001018385823595e-05
max_grad_norm: 1.0601489137351074
num_train_epochs: 2
weight_decay: 0.24082725118120946

The model naming format is: {seed}_{index}. The seed is what we used for the random sample so that the fine-tuning can be reproduced with the same seed. The index here just tells the *i*-th sample being drawn, i.e. it is the model fine-tuned in the *i*-th generation (in other words, our top performing model was the 22nd fine-tuning that used 1 as the random number seed).

Explainability

How is the model using self-attention to make its predictions? There are two visualization tools that can help us understand the process that the model uses to predict the next word.

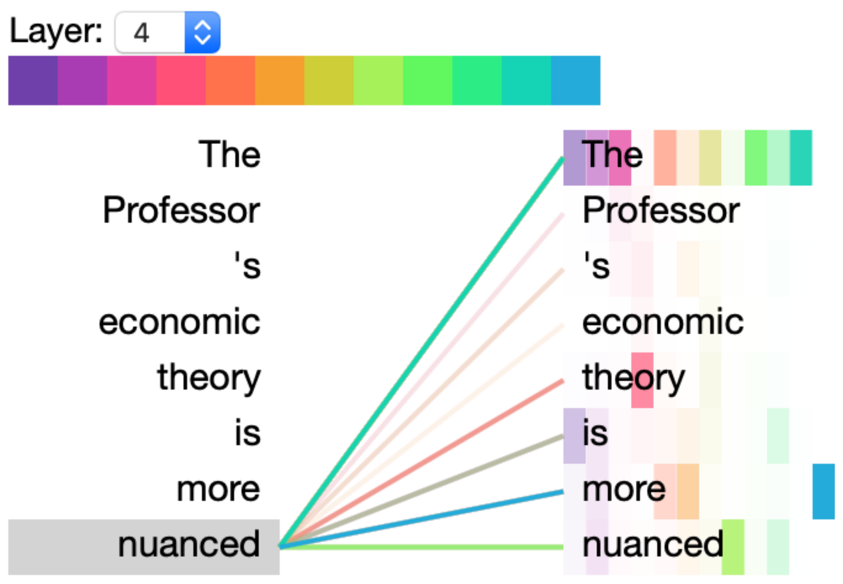
1) BertViz is a tool for visualizing attention in NLP models including GPT-2. Lines connect the tokens that are attending (left) with the tokens being attended to (right). Colors identify the corresponding attention heads, while line weight reflects the attention score. This enables us to examine the attention both by layer and by head within each layer. However, merely understanding the attention paid to each word may not produce a clear picture of how the input sentence shapes the model's prediction of the next word. We should keep in mind that *after* the Attention Layer, the output must still pass through the Feed Forward Layer, whose weights will further transform the output.

2) We therefore use another tool, **Word Importance Visualization**, to measure the relative importance of each word to the ultimate output. It does this by measuring the extent to which each word could be changed without altering the final output. A low score means that even a small amount of "noise" around that word would alter the output. So the more important a word is to the final output, the lower its score.

The Professor's economic theory is more nuanced (100)

BertViz

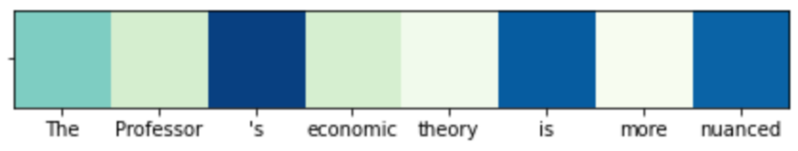
```
In [10]: head_view(attention, tokens)
```



Word Importance Visualization

```
In [4]: interpreter.get_sigma()
Out[4]: array([1.0557312 , 1.2707514 , 0.6834349 , 1.2773333 , 1.393605  ,
               0.75084466, 1.4114597 , 0.7645188 ], dtype=float32)

In [5]: interpreter.visualize()
```



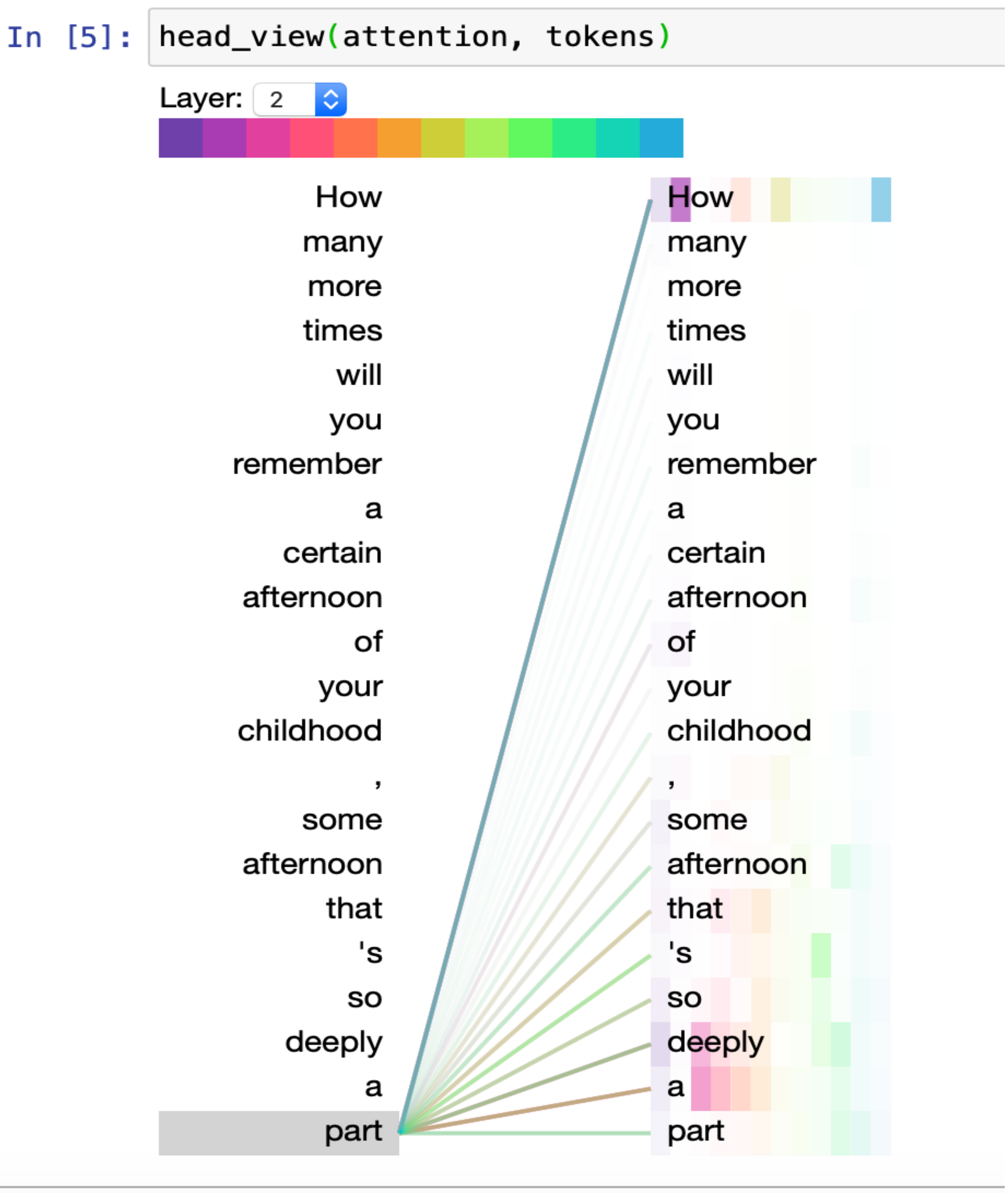
Model: 1_22 output

The Professor’s economic theory is more nuanced than that of anyone else on our list, he seems to be saying very different things. The difficulty lies in its not so obvious scope of concerns. There are those who feel that the political situation distorts learning in Hong Kong and that it betrays the marginalisation of those with less money. Yet Professor Bonnyman comes across as more than eager to talk about issues that concern even the privileged classes in our society. As one former classmate of Bonnyman

The Professor’s economic theory is more nuanced, though, and we mustn’t allow academic discourse to get stuck in “argumentative mode.” It is necessary for the day-to-day functioning of a university to have its dialogue with real life in practice. Failing to speak up today will undoubtedly make the whole semester seem too boring. (Yow Yuk-kuen) received her MA and PhD from Harvard University. She has also taught at the University of Hong Kong, and the University of Chicago.

How many more times will you remember a certain afternoon of your childhood, some afternoon that’s so deeply a part (200)

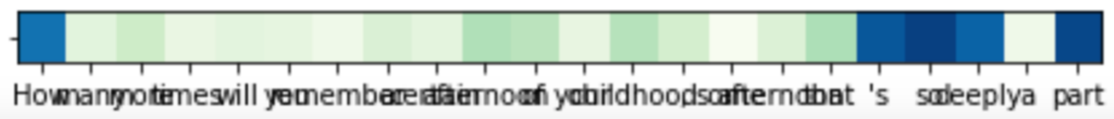
BertViz



```
In [5]: head_view(attention, tokens)
```

```
Out[4]: array([0.541841  , 1.2821257 , 1.1601733 , 1.3226551 , 1.2867883
        ,
               1.3029702 , 1.3542947 , 1.2331558 , 1.2909375 , 1.0450093
        ,
               1.0812416 , 1.3084031 , 1.0633148 , 1.1958865 , 1.3965175
        ,
               1.2421476 , 1.0326487 , 0.45266917, 0.3744312 , 0.48842564
        ,
               1.3544699 , 0.40084958], dtype=float32)
```

```
In [5]: interpreter.visualize()
```



Model: 1_22

How many more times will you remember a certain afternoon of your childhood, some afternoon that's so deeply a part of your heart? The first time I saw the likes of Antonio Vivi and Herbie Hancock in red-and-white clothes onstage at the Golden Glory, I shouted over my shoulder "we are people too! "We are welcome!" the audience chanted. The atmosphere became less and less festive. Few people expressed their excitement, and lots were patting me on the back. It was hard to tell if Vivi and Hancock had received any particular encouragement from the audience. If this turned out to be true, what an amazing story. It's impossible to know whether this show will have much impact on their lives. Perhaps some might begin to get into their Hollywood dreams, or perhaps they'll keep close watch on the latest in Hong Kong art. But a few who were there might just wonder: "Really? Does this show give them hope that there are other ways in which to work?" Looking

How many more times will you remember a certain afternoon of your childhood, some afternoon that's so deeply a part of you, one that will forever be an indelible memory?" It was a challenge worthy of the most magnificent wisdom of Edgar Allan Poe. I remember that scene perfectly, even if I haven't had a chance to see it. The heroine, (Chung Ying), was in her late teens, and at a certain point the camera was cut to show her sitting in a chair covered with photos. Wong tried to take that scene out of her mind, but it just kept haunting me. We know that Chan, at age 29, is a comedian in the best sense of the word. (She won Best Actor at the Hong Kong Academy of Performing Arts' Golden Globes last year for a role she played in (Red Chinese Adoption). She is also known for her well-known hit comedy show, The Powerless Show. Well, I imagine (Chan)'s problem right now is quite simple. She is a lot like Wong: a (