Text Generation Project

Team 7 Executive Summary

## presented by

Isaac Bojorquez

Ezekial Curran

Tonatiuh de Leon

1. Objective of the Model

The objective of the model is to create a large language model capable of generating text in the distinctive style of a particular author or group of authors. Three models were developed in this effort—two based on Recurrent Neural Networks (RNNs) and one using the Transformer architecture. The Transformer model, which is now the industry standard adopted by leading companies such as OpenAI, Anthropic, and Google, represents the cutting edge in text generation technology.

1. Transformer Model Overview

The training of this model was comprised of two main parts:

* **Data Pre-Processing**: which included the downloading, cleaning, tokenizing, sequencing, and shuffling of the data.
* **Model Development and Training**: which included the development of the model’s final architecture, training schedules, and final prompt testing.

This overall approach supports highly technical implementations. An overview of the details are provided in the following sections.

## Getting the Data

In the data pre-processing phase, a decent amount of textual data was gathered, cleaned, tokenized, sequenced, and shuffled to prepare it for effective learning.

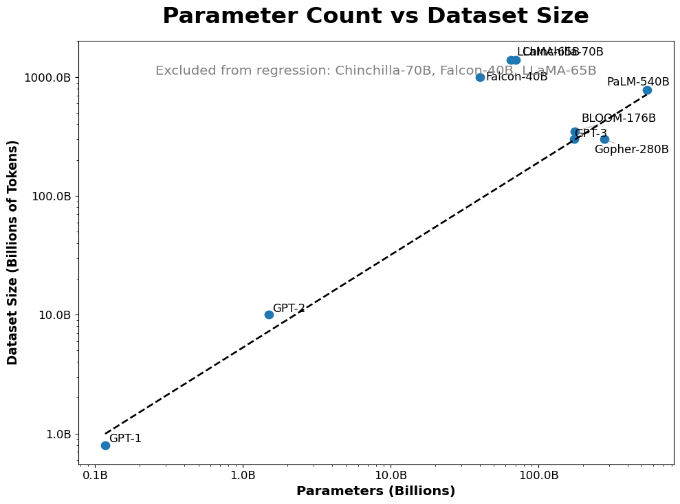
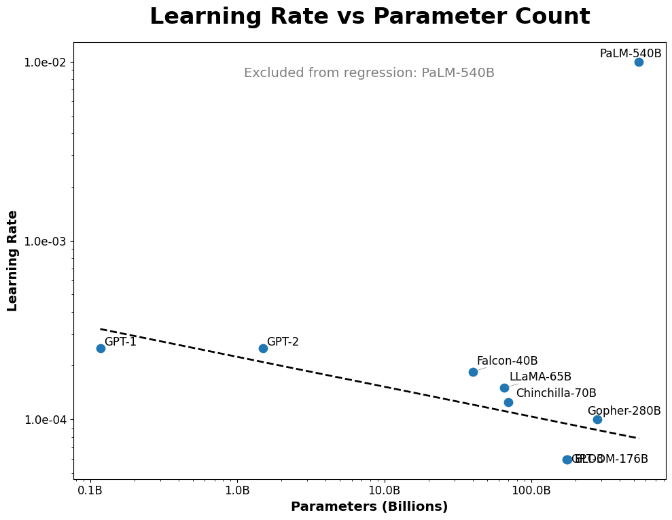
The text corpus was assembled from Project Gutenberg, which provided around 27,000 books, and supplemented with conversation datasets sourced from Hugging Face. The conversation data was included to enable the model to understand and generate responses that mimic user-assistant interactions.

Cleaning involved removing unnecessary line breaks and punctuation while ensuring that only primarily English text was retained—each sequence of 512 tokens was required to contain at least 95% English as determined by the fastText library. Furthermore, the text was tagged to distinguish between prompts and narrative content. A Byte Pair Encoding (BPE) decoder was applied to break down words into smaller sub-word units, creating tokens. In total, a vocabulary of 50,127 unique tokens was established, and approximately 2 billion tokens from the books and conversations were organized into sequences of 512 tokens each before being randomly shuffled to ensure diverse training sequences.

For industry standards this is a smaller-sized model which is great to be able to test train this beta version so that the company can make larger investments as needed. The next picture shows a couple random text sequences extracted from the pre-processed text files to show what the model was fed during training. The picture shows decoded (natural language) text but in training the model was fed encoded (numbers instead of words) sequences.

## 

## Model Development and Training

During the model development and training phase, significant attention was given to managing computational and financial constraints inherent to Transformer models. Research into industry practices for developing large-scale models provided insights for designing a smaller beta version. The focus was on optimizing the number of parameters, the learning rate schedule, the optimizer state, and other hyperparameters of the model. Analysis suggests that, given the dataset size, a model comprising around 125 million parameters would be sufficient to capture the underlying patterns. Additionally, a maximum learning rate of 2.54e-4 was determined to maintain numerical stability throughout later stages of training.

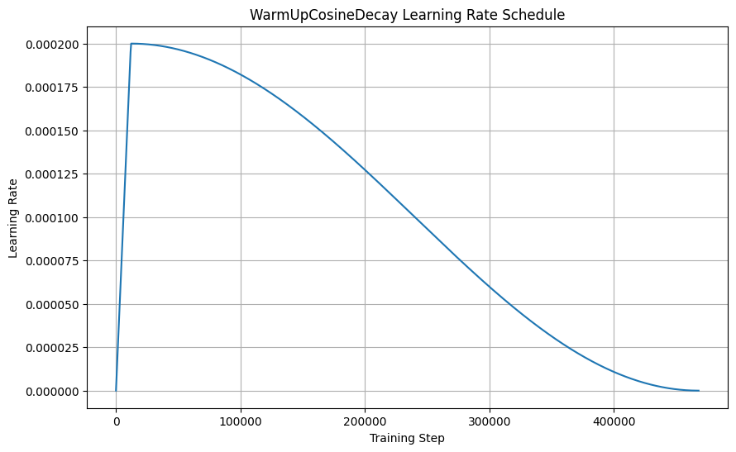
As an interesting note, further investigation on Google’s PALM model would be interesting to understand how it was able to sustain such a large learning rate while still being the largest model in parameter count.

Continuing with model architecture; Other hyperparameters, including embedding dimensions, the number of transformer layers, and sequence length, were similarly calibrated. The batch size was set at 32 sequences per step, constrained by the available computational resources. The AdamW optimizer was employed, configured with parameters such as an alpha of 0.9, beta of 0.995, a clipnorm of 0.6, and a weight decay of 0.01. These settings were chosen to ensure smooth learning progression, control the impact of extreme gradients, and reduce the risk of overfitting respectively.

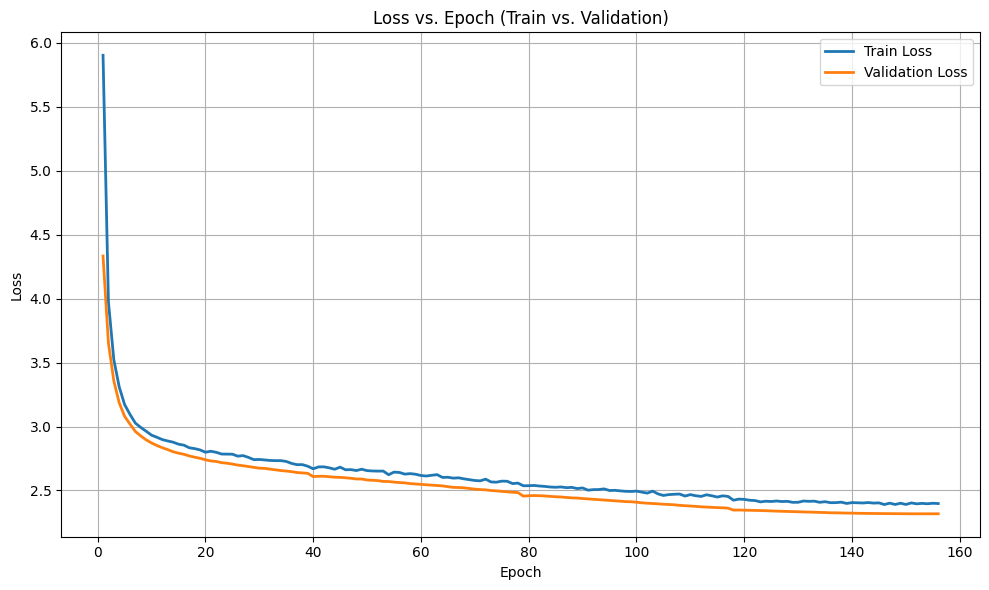
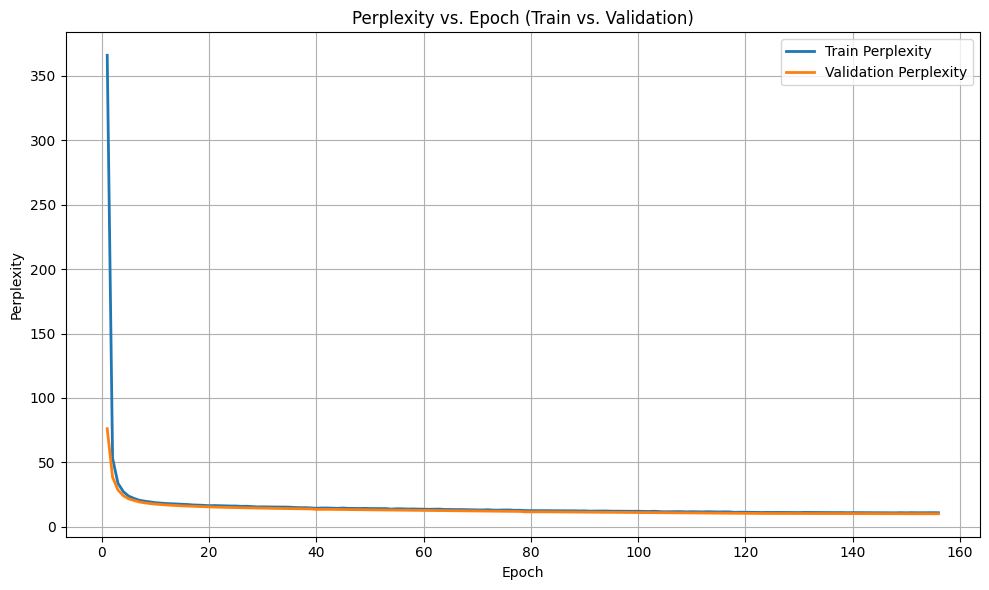
|  |  |
| --- | --- |
| Parameter | Value |
| Alpha | .9 |
| Beta | .995 |
| Clipnorm | .6 |
| Weigth Decay | .01 |

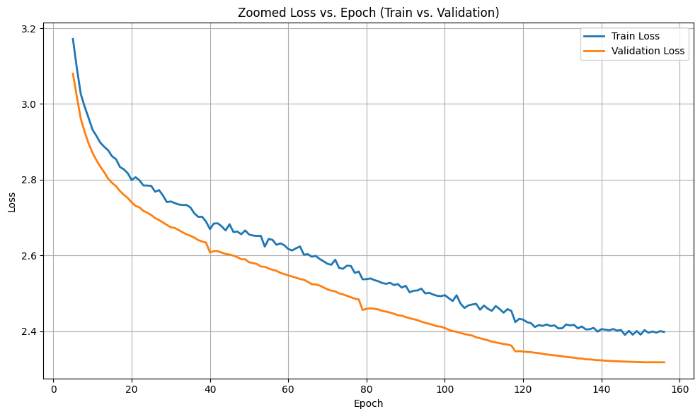
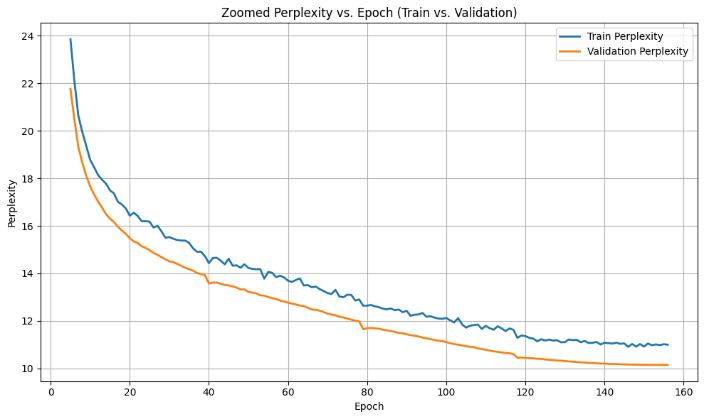
After the Optimizer was set with the best calculated hyperparameters (according to research) a learning rate schedule was generated to emulate natural human brain’s growth. A human brain develops 85% of its neural architecture by the age of 3. Assuming a person lives to the age of 85 that represents only 3.5% of their life.

In the case of this model, the neural architecture is 100% existent since the beginning of its training, but the learning rate schedule was modified to have a 3.5% warmup (percent off of all training steps) and then decay from there in a wave all the way to zero. As a recapitulation, when training deep-learning models, one epoch is a full training pass through the dataset. In this case, to be able to have more control on each training step, each epoch was divided into 40 mini-epochs allowing us to save the model every other 3 thousand steps to keep track of it. In a larger production version this would not be done, and training would be a full epoch at a time.

This is explained because for the first 3 mini-epochs the model will reach its peak of learning speed, and then it will start learning more slowly and making fewer changes here and there until training stops and it reaches zero. This ensures the model learns most of the language architecture earlier in its training, but all the smaller complexities of language are learned little by little as training steps move forward – this is human-like growth.

After getting all the architectural key details set, training began. The optimizer was set to reduce loss on each epoch. Perplexity – a metric that measures how well the model predicts for language complexity – was also recorded. Remember in this case a lower loss, and a lower perplexity is what we are aiming for, so in order to read the next graphs, the Y-axis represents the respective metric, and the X-axis represents the given mini-epoch it was recorded at.

These metrics are not comparable in a same-scale way. They are representations of how far off the model is predicting the next word from each training sequence. In other words, we are just looking for a decaying loss and perplexity to analyze if the model is still learning or not.

Both graphs show that the model is still learning. Let’s take a closer look the same training logs but zoomed-in to see more details.

Despite training being conducted for only 4 epochs (divided into 160 mini-epochs), the gradual decay observed in metrics such as loss and perplexity suggests that further training would yield significant improvements in grammatical accuracy, syntax, and stylistic consistency at inference time. Training would stop when these two lines fully converge because there would be more certainty that the model has learned everything it could before overfitting to its given dataset.

This model has the capacity to keep learning even more and to reach the capabilities of ChatGPT-1; This depends on the budget that the company allows this project to have.

1. Results

The results of multiple test prompts show that the model has already learned enough to be able to write mostly proper grammar. It can form good sentences and write in a poetical way. However, it lacks the ability to maintain structure or connect the dots when changing to a different topic or paragraph. This is not surprising given that training shows signs of only been just began. With the proper amount of full-epochs (estimated to need 40 epochs to be fully trained) this model has significant potential in text generation. Please read some of the sentences generated at different temperatures. Note that in this context, temperature is a parameter that allows the model to be flexible in its writing. Temperature goes from 0 and above, with a higher number allowing the model to write text that is more creative, and lower temperatures forcing the model to stick to what the prompt is actually asking – In this case the default is a temperature of 1.

**Test prompt:**

"The world seemed like such a peaceful place until the magic tree was discovered in London."

**Transformer Model Test Outputs:**

* Temperature 1: “Among the magnificent bygone remnants of the woods, the tree drew curious souls who sensed something extraordinary. Artists, sckepticals, curious and thieves all wanted to see its maginificence.”
* Temperature 2: “For a time, the thought of the tree was so much more than usual among the people of London. Then, as if to divert suspicion and create a real estate bubble from some new interest, it developed into an almost impossible to overcome economic downturn.”

The same prompt was tested against the actual ChatGPT-1 model. These are some of its outputs to compare what a fully trained model would be able to do.

**ChatGPT-1 Test Outputs:**

* Temperature 1: “the young lovers had fallen in love so often their lives were woven together, so easily forgotten . at first they were merely physical , their dreams merely a figment of their imaginations . but it was the tree 's cryptic warnings , the haunting whispers that haunted and made life seem a living...”
* Temperature 2: “One young girl spotted the magnificent wood where its limbs draped in lustrous vines . for the moment her young life was shattered beyond thought by her mother giving birth to twin boys named brian who was almost five at some tender age at this perfect new age .... a child ! -- brought their…”

It is notable that the transformer model (not ChatGPT-1) has already learned to make capital letters after every dot, take out blank white spaces before commas or dots, write proper English (grammatically speaking), and all of these with only four epochs. The transformer model is not too far away from ChatGPT-1. However, subtle differences make ChatGPT-1 be more enjoyable to read. It is also able to maintain longer context windows in its prompts. All of which is possible for the transformer model if given more refined training.

1. Conclusion

In summary, the development of the language model, particularly the Transformer version, demonstrates a robust approach to replicating specific authorial styles. The project integrated comprehensive data preparation, sophisticated tokenization techniques, and meticulous tuning of hyperparameters and learning schedules to optimize performance. Early training results suggest that even the preliminary model exhibits significant improvements in text quality, and with additional investment, it is anticipated to achieve a performance level comparable to established language models. This work represents a significant advancement in leveraging cutting-edge technology to create highly effective text generation systems.

Early training results indicate that the model is still in the early stages of learning, with performance metrics continuing to improve gradually. Even minor decreases in loss and perplexity have the potential to produce substantial enhancements in text quality. With an estimated additional investment of approximately $1,000 to support full-scale training—potentially involving around 40 epochs—the model is expected to reach a level of performance comparable to early iterations of well-known language models like ChatGPT-1.

Notebooks

Below is the code that was used to get the data, clean it, and train the model. The actual training of the model was not done in Google Colab but with a different cloud service provider that offered H100 GPU’s. The notebooks in their instances cannot be saved like in colab, but all the results from training were recorded and showed in respective placeholder Google Colab notebooks below.

**Research on tokenization**: [Tokenization.ipynb - Colab](https://colab.research.google.com/drive/1y0KnCFZvGVf_odSfcNAws6kcDD7HsI0L)  
**Dowloading all data**: [Download all Data.ipynb - Colab](https://colab.research.google.com/drive/1IgeKXothsxaYnAC61ILaE72liAl_SuF6)  
**Tokenization Process:** [Tokenizing Books (part AF) - IMPORTANT.ipynb - Colab](https://colab.research.google.com/drive/1asVR3HDsW1Yor8FsxzIzKzydai6NloYN)  
**Sequencing and Shuffling**: [Sequencing and Shuffling.ipynb - Colab](https://colab.research.google.com/drive/1FFpD5Ppomx55edgil1dH85k8XFACCoPE)  
**Small transformers testing**: [Early Transformer Tries.ipynb - Colab](https://colab.research.google.com/drive/1g3R_nrJkugWTJpeQquib1uXd3cN1zSOd)  
**Checking data pre training**: [Checking sequences randomness and learning rate schedule - Colab](https://colab.research.google.com/drive/1ceimmIbUCC4RVfGxOkqW88_MnhCeX3_8)  
**Final training results**: [the final notebook.ipynb - Colab](https://colab.research.google.com/drive/1Yspt8uR8zcPH9u_KmnZrGmlGtTE_jqmI)

1. Discussion Responses

For the 1000 test text a temperature of 1.2 was used alongside of a top k tokens of 40. Meaning that a higher than average creative writing was allowed at inference time but the amount of different words the model was allowed to output was shortened to not let it hallucinate largely.

**1 thousand tokens test**: [1k tokens – Google Documents](https://docs.google.com/document/d/1yo7JKk9udV7Vw0hH0qFZMIRh3ru_katqG5sMZ2SZ98A/edit?tab=t.0)