Never Ending Stories

Juhwan Lee
juhwan@pdx.edu

Graham Seamans
gseamans@pdx.edu

Tareq Jallad
tajallad@pdx.edu

Abstract

Our original project plan was to use Shakespeare literature as training data and train it on a language model to generate fake Shakespeare plays. We used the PG-19 dataset evaluated with single word perplexity as proposed in *Compressive Transformers for Long-Range Sequence Modeling* by Rae et al. (2019). Although this is not our original task, it was close enough for us to switch over. The PG-19 dataset includes a set of books extracted from the Project Gutenberg books project that were published before 1919. For text generation, we chose the GPT2 language model. In hopes of improving the text generation performance of GPT2, we created an input pipeline which summarized preceding text using the T5 transformer as a Summarizer. The pipeline creates a summary of the text preceding where we generate from in hopes of giving the language model a greater context understanding. Finally, we then compared GPT2 without T5 and GPT2 with T5 on single word perplexity.

1 Introduction

Never Ending Stories is the project about implementing a language model by combining a language modeling transformer and a summarizing transformer to create one language generator that is, unfortunately, likely worse than just a normal language modeling transformer. We originally set out to use transformers to generate text in the style of authors with the dream of creating a transformer than can use a very large input to infinitely extend an already written piece of literature. After researching papers in NLG we realized that this was maybe beyond the scope of this assignment, and so we focused in on a possibly new way to give pretrained NLG's context using pretrained summarizing models. If this story continuation language model is successfully implemented and shows reliable performance, it can be used as a guidance or inspiration for writers who are writing literary works.

2 Related Work

There are many papers such as Beltagy et al. (2020) and Rae et al. (2019) that create models capable of using large input sequences. Creating something similar to these would be great, but due to lack of time, compute, and skill we realized that we were best off using a system which combines pretrained transformer architectures. We found that *Pretrained Language Models for Text Generation: A Survey*Li et al. (2021) is the most similar paper to ours as it covers many uses of pretrained language models.

2.1 Pretrained Language Models for Text Generation

Text generation has evolved into one of the most important, yet difficult, tasks in natural language processing (NLP). The rise of deep learning by neural generation models, particularly the paradigm of pretrained language models, has greatly advanced this field (PLMs). The paper takes on the task of being the first work presenting a comprehensive overview of major advances achieved in the topic of PLMs for text generation, and to provide text generation researchers a synthesis and pointers to related research.

The paper presents general task definitions and outlines the mainstream architecture of PLMs for text generation. The paper explains how different inputs, from noise, to structured text can be combined with different model architectures to achieve diverse outputs. It explains how decoders are useful to auto-regressing tasks like NLG and and encoder-decoder can be used for conditional generation such

as summarizing. The former is used by the models GPT and CTRL, and latter is used by models MASS, T5 and BART. The paper shows how to adapt an existing PLM to model a variety of input data and meet special properties of the generated text, such as relevance, faithfulness, and order-preservation and then covers fine-tuning strategies. One fine-tuning strategy that was extremely helpful was to thoroughly train the parameters of PLMs using task-specific data, so that PLMs can capture the semantic characteristics that are unique to the generation task.

2.2 Long Input Transformer Models

The transformer has been limited by the quadratic scaling of its self attention layers since its inception in Vaswani et al. (2017). To get around this transformers such as Longformer from Beltagy et al. (2020) have been proposed which use sliding window attention and other tricks to reduce the run time of the model. Compressive transformers proposed by Rae et al. (2019) use a transformer that "maps past hidden activations (memories) to a smaller set of compressed representations (compressed memories)".

Longformer and the Compressive Transformer are inspiring, but outside of what we can do given our compute and time budget. Because of this we decided to combine PLM's.

2.3 Evaluation and Datasets in Related Work

We were uncertain on how to evaluate our project task and in looking for standards we found the PG-19 dataset and proposal to use single word perplexity as a benchmark from Rae et al. (2019). Although it was different from our original task of extending prewritten books, it was close enough for us to switch over and therefore we decided to use this paper as a baseline for our model. This paper was published in 2019 by Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. This paper summarizes the findings of a new Compressive Transformer. This paper uses the PG-19 dataset, and proposes it as the new language modelling benchmark, not only for its data containing larger contexts, but also for its size, greater than BookCorpus and Billion Word Benchmark.

3 Methodology

The Pretrained Language Model For Text Generation: A Survey research paper gave us a good understanding of the pretrained language model for text generation and the Compressive Transformers for Long-Range Sequence Modeling research paper gave us clear guidelines for datasets and evaluation. Because of this, we were able to get the project off to a smooth start.

3.1 PG-19 Dataset

The PG-19 dataset is a language modeling benchmark proposed in Rae et al. (2019) to evaluate long format text generation. It includes a collection of books published before 1919 that were extracted from the Project Gutenberg books library. It also includes metadata such as book titles and publication dates that will be ignored in this project. The PG-19 benchmark is more than twice as large as the Billion Word benchmark, with documents that are 20X longer on average than the WikiText long-range language modelling benchmark. The dated linguistic style of old texts and the inherent biases present in historical writing in this dataset is a perfect fit for our project. Because our work environment was Google Colab, this dataset was difficult to handle (being 11 gigabytes), but we found a way to download and use it quickly which made us move on to the next step.

3.2 GPT-2 Language Model

Because of its size and output quality, GPT2 became our model of choice. We had hoped to train with larger newer models like the GPT-J, but we were unable to do so because they required too much V-ram. Although it appears that we could have trained the models using TPUs, we decided against it because it would have pushed back the already tight project timeline. When deciding between models that could fit on the GPUs we had access to, such as GPT-2 and Reformer, we did not run any quantitative tests, but we found that GPT-2's generated text was the most 'human' of the transformers we tested.

3.3 T-5 Transformer

In hopes of improving the text generation performance of GPT2, we created an input pipeline which summarized preceding text using the T5 transformer as a Summarizer. The pipeline creates a summary of the text preceding where we gener-

ate from in hopes of giving the language model a greater context understanding. T5 is a new Google transformer model that is trained in an end-to-end manner using text as input and modified text as output. It uses a text-to-text transformer trained on a large text corpus to achieve state-of-the-art results on multiple NLP tasks such as summarization, question answering, machine translation, and so on. For this reason, we decided to use the T5 Transformer as our Summarizer.

3.4 Overall Pipeline

To properly check the performance of our unique language model, which fine-tuned GPT2 with the PG-19 dataset and created the input pipeline using the T5 Transformer as a Summarizer, we should try and compare three experiments. The first is to generate text using GPT2 that has not been fine-tuned, the second is to generate text using GPT2 that has been fine-tuned, and the third is to fine-tune GPT2 using T5 Transformer and then generate text using summarized text. And we will come to a final conclusion by comparing and evaluating the single word perplexity score of fine-tuning without T5 Transformer and fine-tuning with T5 Transformer.

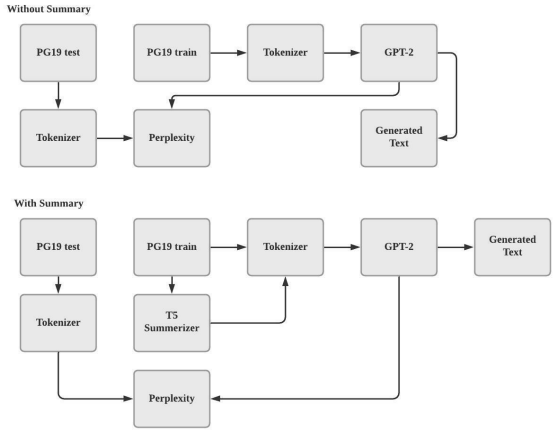

Figure 1: Overall Pipeline

3.5 Single Word Perplexity

As proposed by Rae et al. (2019) we use the single word perplexity as a metric for evaluation. Perplexity (PPL) is one of the most common metrics for evaluating language models. Perplexity is defined as:

$$PPL = e^{-\frac{1}{t}\sum_{i=0}^{t} logp(x_i|x_{<x})}$$

Intuitively, it can be thought of as an evaluation of the model's ability to predict uniformly among the set of specified tokens in a corpus. Importantly, this means that the tokenization procedure has a direct impact on a model's perplexity which should always be taken into consideration when comparing different models.

The hiccup for us in using perplexity is in how we have modified our input. As seen above, perplexity is normally found starting from the first character in a sequence as we sum from $i = 0$ to t. our input including summarizing has a summary for the first half of the sequence. Because of this we decided to modify perplexity to be:

$$PPL = e^{-\frac{1}{t}\sum_{i=\frac{t}{2}}^{t} logp(x_i|x_{<x})}$$

With this we start evaluating perplexity half way into the example. This has the benefit if no longer expecting the model to be able to predict a summary, but now the model without the summary has a large advantage. The non-summary model has the whole first half of the example to 'warm-up' to get better prediction, while the model with summary only has a summary to work with.

One work around for this is to make testing data which only evaluates perplexity on tokens predicted after the model input length, and another would be to make loss prediction begin at $3t/4$, to give the summarizing model a bit of warm up on raw text.

It turned out to be a little more difficult than we anticipated but we found an open source code that was extremely helpful and that is Calculating perplexity with GPT2 in Transformers from Hugging Face.

4 Experiments

To begin with, generating text before fine-tuning GPT2 was not difficult. However, due to the 11 gigabytes of data, fine-tuning GPT2 with and without T5 took a long time to complete and sometimes we experienced runtime disconnect while training. However, we were able to complete fine tuning by finding some code to prevent Google Colab runtime disconnect from the website using **?** How to stop Colab from disconnecting? — 2021 Solution. For fair evaluation and comparison, both the model without fine-tuning and the two models with fine-tuning have the same input text.

4.1 Before Fine-Tuning GPT2

First, we tried to run text generation from GPT2, which was not fine-tuned. The result is as below.

4.1.1 Input Text

The king marvelled much at her silence, and presently repeated his questions, adding, ”And what do you carry so carefully in those two sacks, which seem over-heavy for your delicate shoulders?” Still holding her eyes downcast, the princess took a ruby from one bag, and a sapphire from the other, and in silence handed them to the king, for she willed that he should know she was no beggar, even though her shoes were dusty. Thereat all the nobles were filled with amazement, for no such gems had ever been seen in that country. But the king looked steadfastly at the princess, and said, ”Rubies are fine, and sapphires are fair; but, maiden, if I could but see those eyes of yours, I warrant that the gems would look pale and dull beside them.” At that the princess raised her clear dark eyes, and looked at the king and smiled; and the glance of her eyes pierced straight to his heart, so that he fell on his knees and cried: ”Ah! sweet princess, now do I know that thou art the love

4.1.2 Generated Text

of the King. I am so glad that thou art so dear to me. I wish to be king until thou canst be queen. Now, my sweet heart, if I could only see the crown with my own eyes, I could not be mad; for I would have never asked thee to be my wife to you. I would have made thee a queen; and I would have made thee a queen if thou were my own wife.” The king and the princess were delighted to hear the king’s furious cry, and when they saw this, they both rose. ”Princess!” exclaimed the king, ”what did thou say, and how will thou do me no harm?” ”I never said that,” replied the princess, ”for thou made me such a great hope and such a great desire that I could never see thee again. I would have never tried to see thee again, and yet thou were so far from seeing me that I could not see thee any more of thee. If thou didst seek the King to give me my crown, I would have loved thee well

4.1.3 Observation

When we examine the results, we can see that the text flow continues uninterrupted. However, it is still not the writing style we desire. We were able to observe the performance of the GPT2 that had not been tuned.

4.2 Fine-Tuning GPT2 without T5

Then we ran text generation using fine-tuned GPT2 without T5. Input text will be skipped since it is the same as before.

4.2.1 Generated Text

of man, and I believe that thou art the love of the kingdom, and, with me, that thou mayest bear me! I have only one love, that is, that which shall be always with thee.” When the King heard this, he said, ”Oh, sweet princess, we have sworn to keep thee for ever, and we shall never forget that thou art my beloved.” And the princess, in her tears, said, ”I will never forget that thou art my love. Thou art my love, and I love thee. Thy love is the love of the kingdom.” Then she said, ”Yes, I promise to keep thee. Thy love is the love of the kingdom. Do thou not believe in it myself, and believe in every one that has it, that I have ever given it to thee.” And the king smiled as if he was happy. And, as he spoke, his eyes grew wide, and he said: ”And now, O king, I will never forget that thou art my love, and that thou art my wicked love. Thou art my love, and I love thee. Thou art my love, and I love thee

4.2.2 Observation

The result shows a clear difference from the previously generated text. The text generated by this model follows the flow of writing naturally while at the same time showing the old literature style of writing. Clearly, the text generated by a model trained on the PG-19 dataset is closer to the style of writing we are aiming for.

4.3 Fine-Tuning GPT2 with T5

Finally we tried text generation using fine-tuned GPT2 with T5. This time, the input text is different from previous attempts because the summary of the input text is also used as the input text.

4.3.1 Input Text

by dale and by down, through moss and through meadow, and by-and-by a fair high palace, built all of marble and shining jasper, and with two bags of gems hung upon her shoulders, fared forward with a light heart, by dale and by down, through moss and through meadow. The princess, standing alone at his palace-gate, took a ruby from one bag, and a sapphire from the other, and in silence handed them had ever been seen in that country. But the king looked steadfastly at the princess, and said,

”Rubies are fine, and sapphires are fair; but, maiden, if I could but see those eyes of yours, I warrant that the gems would look pale and dull beside them.” At that the princess raised her clear dark eyes, and looked at the king and smiled; and the glance of her eyes pierced straight to his heart, so that he fell on his knees and cried: ”Ah! sweet princess, now do I know that thou art the love

4.3.2 Generated Text

of the world?” Then, before her eyes she stretched out her hands, and said, ”I know that the gems will look pale and dull without me, and without all those eyes of mine, and without all those eyes of mine!” The king looked at her in amazement, and said, ”I know what I cannot do, but, and I know the gems will look pale and dull; and I know the gems will look pale and dull with me without thee, even without thee, and without all those eyes of mine.” And the princess opened her eyes and said, ”My heart never can begin to grow faint. But I must trust in thy love and love and the love of the world, because thou art the love and love of the princess. I am not weary, my heart, and I am not weary; but I fear that I cannot be weary and my heart with thee; and I fear that I cannot be weary and my heart with thee. The ruby and the sapphire are a fine gem, and I can see them, and it is very hard to see them, and I am not doubtful it will not be as she thinks; but the sapp

4.3.3 Observation

As a result of checking the generated text, unfortunately, there was no significant improvement from the text generated by the previous model. It shows the old literature writing style while continuing the flow of writing naturally, but there are some parts that feel a little unnatural.

5 Results

The possibility of creating a long input transformer without the need to train a very large language model like GPT-3 is exciting, however the results from our experiments seem to be largely inconclusive because our testing setup was rushed. As stated in section 3.5 we are unsure how to evaluate PPL correctly given that our inputs are modified. A second, and maybe stronger reason against our results is how we evaluated our model.

Ideally our pipeline runs through and creates a new input for each token generated, so that the generator has the same relative context provided by the summary and amount of preceding raw text. Instead we are evaluating loss on a section of raw text following a summary. Because of this, as we generate text the summary becomes less relevant and we have an uneven amount of raw text to use as context. All of these issues are fixable with more engineering and research effort, and are not inherent to the ideas proposed in this paper.

Input Pipeline	Single Word PPL
With Summary	29.83
Without Summary	25.04

Table 1: PPL results.

6 Future Directions

There are a few ways to take this idea, which include using larger transformers, and better engineering for evaluation and generation.

6.1 Different Summarizing Transformers

We chose T5 for this project, but ideally something like LED from Beltagy et al. (2020) would be used as our summarizer. T5 is limited to very small inputs which limits the amount of context we can feed to our generator.

Implementing LED as a summarizer with the 16k token input length would be great. One option is to use 16k input tokens, take that down to 512 of summary tokens, combine that with 512 of raw input and use the full 1024 input length of GPT-2. Another option is to use LED into Longformer also from Beltagy et al. (2020), which can also use an input length of 16k. This would allow us to summarize multiple 16k long preceding token sections and feed them all into a Longformer to give very long context to the generator. For extra long inputs the summarized sequences could be summarized giving an effectively infinite possible input length, although any bias held by the summarizer would be amplified by each summarizing pass.

6.2 Better Evaluation and Generation

The biggest design challenge for this project that we still have not understood is where summarizing transformer belongs. The summarizer could either go into the model itself, or into an input pipeline. If the summarizer is in the model, we have compute and V-RAM issues, as two medium transformers cannot fit onto V-RAM of any of the accelerators we are currently using, and we would need to run a

summary generation each time we want to predict a word. Although this is ideal, generation, especially with multiple beams, takes a good amount of compute. There are a few hybrid approaches to this that may work, but really just having more computation may be the answer.

Another exciting possibility from combining summarizer and auto-regressive models into one would be being able to train the summarizer and the generator at the same time. This is really exciting, but would mean that we would be training what is effectively a massive NN. Issues include: storing the massive gradient from all of the forward passes made during summary generation, keeping any amount of gradient in what would be a very tall unrolled net, and finding or creating a differentiable conditional generation algorithm.

7 Ethical Considerations

If our proposed model works as intended, then we will be able to generate stories based on any style, author, or previous written text. These stories would be never ending, and take into context previous events, characters and details. In theory, many people would be able to benefit. Authors would be able to use the model to gather inspiration and ideas while writing stories. This in turn benefits the consumer and the general public, as more literary works could be published. However there is a chance some authors, dead or alive could be harmed. If the model was trained on an authors work, without their permission, it could generate stories written in an authors style. While a "style" is not necessarily protected by law, it still takes something unique to an author, and provides other people the ability to recreate it, lowering the authors value and demand.

Our model is currently limited to only English text generation. This is in part because we fine tuned the model on the PG-19 dataset, which is comprised of English books written before 1919, and also because our model uses GPT-2, which was pretrained on English text. Due to the fine tuning on PG-19, the model was mostly trained on male authors.

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *CoRR*, abs/2004.05150, 2020. URL https://arxiv.org/abs/2004.05150.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, and Ji-Rong Wen. Pretrained language models for text generation: A survey. *CoRR*, abs/2105.10311, 2021. URL https://arxiv.org/abs/2105.10311.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, Chloe Hillier, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint*, 2019. URL https://arxiv.org/abs/1911.05507.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.