Fine-Tuning of Transformer Models for High Quality Screenplay Generation

Stanford CS224N Custom Project No external collaborators, Mentor: Yuyan Yang, project is not being shared.
We use 1 late day for this assignment.

Jordan Byrd
Department of Computer Science
Stanford University
jbyrd23@stanford.edu

Anthony Le
Department of Computer Science
Stanford University
antle1@stanford.edu

Anchit Narain
Department of Electrical Engineering
Stanford University
anchitn@stanford.edu

Abstract

Screenplays contain semantically and structurally rich text as the average movie screenplay is thousands of words (tokens) long and contains long range dependencies of entity relations and contextual plot elements throughout. Large-scale pre-trained language models (like GPT-2) perform very well in open-domain text generation when the generated outputs are only ten to low hundreds of tokens long. This project aims to test how well current large transformer models perform at producing long, coherent texts for the task of movie screenplay generation. We compared the outputs of several different models such as GPT-2, GPT-2 finetuned for 1 epoch, GPT-2 finteuned for 3-epochs, and a recently published non-monotonic, progressive generation approach [ProGeT] to see which model and architecture could best support high quality screenplay generation. We found that non-monotonic generation approach performed best on a set of automated evaluation metrics, including the contextual embedding similarity approach known as BERTScore. Looking at the example model outputs [appendix], we can see that the non-montonic approach, ProGeT, has outputs that read most similarly to human written movie screenplays.

1 Introduction

We have seen multiple examples of neural text generation throughout the quarter. It is remarkable that large, transformer based language models (LMs) can be fine-tuned for various downstream text generation tasks such as generating prose [1], poetry [2], rap lyrics [3] [4], and more. However, long form text generation (100s to 1000s of generated tokens) is still difficult for such models due issues like exposure bias (explained below), and their best performance is on short form generation. Seeing that large pre-trained, transformer based LM's were able to generate high quality texts in a variety of short-form artistic styles, we were motivated to apply them to the task of screenplay generation because of a screenplay's unique artistic structure (i.e. spacing, indentations, capitalization), variety of text content (i.e. stage directions, dialogue, setting), and extensive length (> 1000 tokens per scene) to test the limits of state of the art neural methods for long text generation.

When doing research for this project, we found no academic papers focused on applying large, transformer based language models for the specific task of screenplay text generation. As such, we focused on understanding recent or often cited papers on LMs for language generation (i.e. GPT-2, BART) in other task domains, how to evaluate the quality of generated texts, and then we applied those techniques to the screenplay generation task using methods like those described in this blog post: [5].

We evaluate several GPT-2 baselines against the recently proposed ProGeT architecture from the paper, "Progressive Generation of Long Texts", to find a better method of long (> 1000 tokens) screenplay generation. As expected, we saw noticeable improvements when comparing the synthetic text produced by the non-finetuned GPT-2 and fine-tuned GPT-2 on a collection of movie screenplays. We saw even more significant gains with the ProGeT model architecture, as some of the synthetic texts produced by a GPT-2 $\rightarrow$ BART $\rightarrow$ BART non-monotonic model were almost indistinguishable from movie screenplays. The ProGeT outputs saw significant improvements in overall text coherency, structure, and across all automated evaluation metrics. We compared the outputs of all three of these models with automated statistical scores (MS-Jaccard, Frechet BERT Distance, TF-IDF Distance, Harmonic BLEU), and include a new LM based evaluation approach, BERTScore, which uses the encoder to measure similarity through contextual embeddings.

2 Related Work

One of the most commonly used transformer based language models for natural language generation is GPT-2 [6], a language model that generates the next token in an output text sequence by picking the most likely token to follow the model's input text sequence based on a conditional probability distribution learned during training. After a token is generated, it is added as the rightmost token in the model's input text sequence and the model generates the next word in the output sequence in the same auto-regressive fashion. It is important to note that the GPT-2 generation scheme is decidedly left to right as it generates tokens to the right of the last token in the input sequence (i.e. GPT-2 has a left-to-right decoder).

A common problem many LMs face during long form text generation is exposure bias, where the LMs generate text based on their learned training data distributions and not the model input distribution [7]. As such, current large pre-trained model outputs contain repetitive text and incoherence between utterances that are far apart in generated passages of > 1000 tokens.

A lot of research has been done in experimenting with the best way to generate human level text. The idea of discourse planning, a method that involves arranging parts of the generated utterances beforehand, is not a novel idea and has been explored by papers as early as the late 1990's [8]. In more recent development, the aid of neural networks has replaced the once manual or heuristic based approaches of filling in the pre-planned structures for the output text. We can see this in approaches that use non-monotonic and non auto-regressive multi layer approaches to produce natural text. Non-monotonic models do not produce text output in left-to-right fashion, rather they have their own methods of building out utterances. Some architectures including selecting a random word and building a binary tree from outputting the next or previous word for any generation. Model architectures like these have shown promising results in downstream tasks such as translation [9], so we would like to test it on a strictly domain specific generation task.

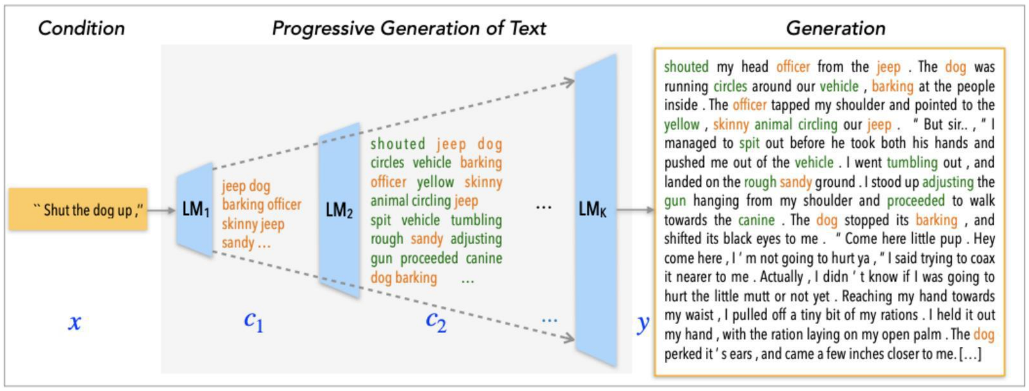
3 Approach

Our main approach uses the model architecture proposed in the paper, "Progressive Generation of Long Text", where instead of generating a full output screenplay y from input x directly, y is constructed in multiple intermediate stages. So, the generation process looks like: $x \rightarrow c_1 \rightarrow ... \rightarrow c_K \rightarrow y$, where for each stage $k \in \{1, ..., K\}$, c_k is an intermediate generated sequence that contains information about y at a given granularity. We thought the non-monotonic generation would lend itself extremely well to a downstream task that involves such complicated structure and entity dependencies.

ProGeT's Progressive Generation: The model's first layers of LM's produce more informative tokens that serve as the base of the output such as key entities. For each successive stage k, we build our output with finer-grained detail by adding less informative words based on thse in c_{k-1}. The generation can be described as a modification of the conditional probabilities at each stage k: $P(y, \{c_k\}|x) = P(c_1|x) \prod_{k=2}^{K} P(c_k|c_{k-1}, x)P(y|c_k, x)$.

ProGeT Model Architecture: Pre-trained language models like GPT-2 serve as the building blocks for the ProGeT model. ProGeT is built atop LM's that can produce outputs, y, in the form of $y = [y_1, y_2, ..., y_T]$ based on conditional/unconditional probabilities represented by $P_\theta(y|x) = \prod_T P_\theta(y_t|y_{<t}, x)$. Where $y_{<t} = [y_1, ..., y_{t-1}]$ and $P_\theta(y_t|y_{<t}, x) = softmax(h_{t-1}W)$. These LM models use nucleus sampling to avoid common decoder issues, which truncates the conditional probability distributions to their top-p probability mass.

ProGeT produces an output, y, in non-monotonic fashion by leveraging a series of k LM building blocks that pass in their outputs to the next model. Illustrated below is the model architecture.


Condition
Progressive Generation of Text
Generation
`` Shut the dog up ,"
LM$_1$
jeep dog barking officer skinny jeep sandy ...
LM$_2$
shouted jeep dog circles vehicle barking officer yellow skinny animal circling jeep spit vehicle tumbling rough sandy adjusting gun proceeded canine dog barking ...
...
LM$_K$
x
c_1
c_2
...
y

shouted my head officer from the jeep . The dog was running circles around our vehicle , barking at the people inside . The officer tapped my shoulder and pointed to the yellow , skinny animal circling our jeep . " But sir.. , " I managed to spit out before he took both his hands and pushed me out of the vehicle . I went tumbling out , and landed on the rough sandy ground . I stood up adjusting the gun hanging from my shoulder and proceeded to walk towards the canine . The dog stopped its barking , and shifted its black eyes to me . " Come here little pup . Hey come here , I ' m not going to hurt ya , " I said trying to coax it nearer to me . Actually , I didn ' t know if I was going to hurt the little mutt or not yet . Reaching my hand towards my waist , I pulled off a tiny bit of my rations . I held it out my hand , with the ration laying on my open palm . The dog perked it ' s ears , and came a few inches closer to me. [...]

Figure 2: Progressive generation of long text **y** given any condition **x**. Each stage refines the results from the previous stage by adding finer-grained details. Added content at each stage is highlighted in different colors.

In order for each stage LM to produce the intermediate sequence c_k, we must develop stage-wise vocabularies for each k stage. For each stage k we have a vocabulary V_k that contains a certain amount of words within the same range of stage-wise importance. Stage-wise importance for a word is measured as the average TF-IDF measure across all documents containing that word.

ProGeT Training: For each document, the sub-sequences c^*_{k-1} and c^*_k are extracted from stage-wise vocabularies V_{k-1} and V_k respectively. The sub-sequences contain all tokens from the document belonging to their respective stage vocabularies. c^*_{k-1} is the input and V_k is the ground truth output for training the LM at stage k via max-likelihood learning. This approach allows LMs at different stages to all train in parallel once their respective, independent stage-wise vocabularies are extracted from a document in the training set. In such a training scheme, each LM's outputs are conditioning on ground-truth input sequences from a training document. But during generation, an LM takes imperfect sequences produced at the previous stage as inputs, so the LM's outputs may have mistakes if it doesn't see noisy data during training (this is the exposure bias problem). To combat this, ProGeT injects noise into each ground truth input at every stage during training by randomly selecting an n-gram in the input and replacing it with another randomly sampled n-gram. This noising process forces the LMs to learn richer language representations that identify errors in the input data, yielding less mistakes in the generated outputs.

GPT-2 as Baseline: We chose GPT-2 as our baseline because it was publicly available via Huggingface and is one of the field's standards for language generation comparison. auto-regressive language models like GPT-2 generate the next token in an output sequence based on the product of conditional probabilities up to the point of generation. While this allows for good short text generation, longer texts tend to become incoherent as the conditional probabilities for the most appropriate next token get too low. GPT-2 can be used as an LM at any stage in the ProGeT architecture, so our final results can compare the efficacy of progressive generation using GPT-2 at all layers in ProGeT vs. single shot generation using a baseline fine-tuned GPT-2. Below are our results on from the stock Huggingface pre-trained GPT-2 and GPT-2 finetuned on our training set of screenplays for 1 epoch. Here, we evaluated on 50 generated texts against 50 reference texts. As expected, even after 1 epoch of finetuning, the model performs better on virtually all harmonic BLEU, TID, and MS-Jaccard metrics compared to the pre-train only model. We used these results along with human evaluation of the generated outputs to quickly debug our generation and evaluation code. We did not evaluate BERTScore or FBD for this preliminary testing as those metrics takes non-negligible compute time on the VM.

Preliminary Evaluation		
Evaluation Metric	pre-trained GPT-2	GPT-2 (1 epoch)
HA BLEU 2	45.3434	**48.858**
HA BLEU 3	27.3122	**29.744**
HA BLEU 4	17.22644	**18.76797**
HA BLEU 5	12.4198	**12.87**
TID	7.2120	**2.171988**
MS-Jaccard 2	14.849	**17.69978**
MS-Jaccard 3	8.796	**9.8546**
MS-Jaccard 4	5.887989	**6.0162444**
MS-Jaccard 5	**4.4141**	4.1456

Code: We adapted the IMSDB scraper from [5] and used their screenplayData class to tokenize each scraped screenplay in order to feed it into our language models. We adapted the following notebook [10] to fine-tune Huggingface pre-trained implementations of GPT-2 and BART on our screenplay dataset. We then generated synthetic screenplay-like passages of token length 1024 to evaluate against an unseen test set of truncated screenplays of token length 1024. We adapted the evaluate function from the ProGeT repo [11] to work with our generation screenplay and get baseline metrics (Harmonic BLEU, FD-IDF Distance, Frechet BERT Distance, and MS-Jaccard) for pre-trained and fine-tuned GPT-2 outputs. We added BERTScore [12] as an additional evaluation metric to compute generated output quality without using explicit word similarity with the reference texts and using embedding similarity instead. Our repo is linked here: **https://github.com/antle1/cs224n_FinalProject_ProGeT**

4 Experiments

4.1 Data

We generate training, dev, and test sets using the Internet Movie screenplay Database (IMSDb) [13]. Though this database only contains $\sim$ 1300 screenplays, each has an average of 30,000 words, giving the models $\sim$ 39 million sequences of words work with. The screenplays on IMSDb are all in plain text format, so they are easily scraped by URL using ScraPy, and the scraped screenplays maintain their original structure (i.e. indentation, stage directions, and other screenplay artifacts that specifically denote scene action, location, and dialogue). We want our trained models' outputs to maintain such structure. Additionally, each screenplay on IMSDb has user generated genre labels, but for the purposes of this project, we did not include these features.

Since each screenplay averages 30,000 words in length, we cut the screenplays into smaller pieces before feeding them as inputs to our models. The model inputs were screenplay chunks of size 1024 tokens (1 word = 1 token) and model outputs were also generated in chunks of size 1024. This made the inputs much faster to train with and the outputs much more efficient to generate. Note that the generated outputs of length 1024 tokens are still considerably longer than most generated outputs for other finetuned applications, so the methods here are still testing quality of long text generation. Additionally, to use the data in the ProGeT model, we had to build stage wise vocabularies using

TF-IDF measures. We created 3 sets for each of the 3 datasets: one with 20% of the total vocabulary, one with 25% of the total vocabulary, and one with the full vocabulary.

4.2 Evaluation method

We used 4 of the 5 metrics used in the original ProGeT paper: MS-Jaccard 2-5, Frechet BERT Distance (FBD), TF-IDF distance (TID), and Harmonic BLEU 2-5. MS-Jaccard measures the similarity of n-gram frequencies between two sets of texts with the Jaccard Index. The numerical value given at the end is for the "n" in n-gram. FBD uses BERT as a feature provider and computes the distance with features in each of BERT-Large's layers. The sum of layers 1-8, 9-16, 17-24 are denoted as FBD-S (shallow), FBD-M (medium), FBD-D (deep) respectively. For FBD, the lower the value, the better the model's performance. TID depends on TF-IDF. The term frequency (TF) is just the frequency of words in the document, and the inverse document frequency (IDF) tells us the importance of a word to a document. TID finds the distance between the average TF-IDF features of two passage sets, so the lower this metric, the better the score. Harmonic BLEU is just the F1 score of the Forward and Backward BLEUs. This allows us to measure domain specific output generation quality and is more reliable than either Forward or Backward BLEU by itself.

Note that these 4 methods are computed using relaitvely simple statistics such as exact n-gram similarity across the generated and ground truth (test) sets of test. As such, they are incredibly sensitive to the size of the sets being measured, as n-gram similarity increases with the number of compared texts. We can see this in BLEU, which only measures the n-gram overlap between output and reference texts. These methods don't account for semantic meaning and lexical/compositional diversity, resulting in incomplete evaluation scores. As such, we decided to also use BERTScore to evaluate our generated screenplays. BERTScore uses the contextualized embeddings from the BERT Encoder to compare semantic similarity. In doing this, the common pitfalls of statistical evaluation measures such as penalizing paraphrasing, inability to hold distant dependencies, and not penalizing semantically critical ordering changes - all things that are necessary in screenplay generation - are mitigated. BERTScore is computed by matching each x token to a token $\hat{x}$ to compute recall, and each token $\hat{x}$ to a token in x to compute precision. The algorithm then uses greedy matching to maximize the similarity score of each x token. Below, the method is described.

$$R_{\text{BERT}} = \frac{1}{|x|} \sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} \mathbf{x}_i^\top \hat{\mathbf{x}}_j, \quad P_{\text{BERT}} = \frac{1}{|\hat{x}|} \sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} \mathbf{x}_i^\top \hat{\mathbf{x}}_j, \quad F_{\text{BERT}} = 2\frac{P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}} \ .$$

In addition to automated evaluation scores, we examined the output of each of the model's as human evaluators and saw noticeable differences between each of the model's outputs. Knowing that human evaluation is still the best way to see if the output text, we went through most of the output texts to note down any common mistakes the model made in generating screenplays.

4.3 Experimental details

We ran several experiments throughout the course of this project. As seen previously, we first output 50 generated texts from the stock implementation of GPT-2 from Huggingface and finetuned that same GPT-2 on our training set. After seeing evaluation metrics for those preliminary experiments as well as manually examining the output screenplays, we decided to train GPT-2 for more epochs to see if we could get a significant increase in screenplay-like structure and overall text coherency. Additionally, we trained our ProGeT model on our full training screenplay to compare against the GPT-2 trained for more epochs. Below describes our experimentation in more detail.

Initially, we used the HuggingFace implementation of a pre-trained GPT-2 model. This model was pre-trained on wiki text, and we finetuned it on our movie screenplay dataset. We used the following parameters:

- Batch size = 4
- Epochs = 1
- Learning Rate = 0.00002

- Warmup Steps = 10000

We also used an AdamW optimizer that was initialized with this learning rate. This optimizer varies the learning rate for different parameters, as some parameters may need to be updated more than others (as we learned in assignment 3). We then trained the model on our screenplays, which took somewhere around 6 hours on the Azure virtual machine, and once this was done, we were able to generate text. The text was generated using the following parameters:

- Number of beams = 5
- Max Length = 1024
- Top Probability = 0.85

Number of beams denotes the number of beams for beam search, the max length denotes the number of tokens we generated, and the top probability is used to limit variance in the pool of generated tokens.

In addition to using a pre-trained GPT-2 model, we used the ProGeT model from [14]. We finetuned a ProGeT model whose first stage used GPT-2 and the following 2 stages used BART. The GPT-2 stage was trained on the top 20% of the total training set vocab (i.e. the most important/least common identifying words in the training set which would include names, locations, and other key screenplay artifacts). For the intermediate stage BART, the LM was trained on the top 25% of the total training set vocab. For the final stage BART, the LM was trained on the full training set vocab. For both the GPT-2 and BART stages, we used the following parameters:

- Batch size = 5
- Epochs = 3
- Learning Rate = 4e-5
- Adam Epsilon = 1e-8
- Warmup Proportion = 0.1
- Weight Decay = 0

4.4 Results

Below, we present the evaluation scores for each of the following models: GPT-2 (finetuned for 1 epoch), GPT-2 (finetuned for 3 epochs), and ProGeT (GPT-2 $\rightarrow$ BART $\rightarrow$ BART). Each model generated 500 screenplays, all of token length 1024, and compared those to a set of 500 actual screenplay segments of token length 1024 from our test set. We see that the non-monotonic model, ProGeT, marginally outperforms the baselines on all but one of the automated evaluation metrics, including BERTScore [12]. As expected, ProGeT's data noising in LM training aided in its achieving better evaluation scores, just like we saw with non-monotonic neural translation in [9]. This was not surprising as the aim of building out the texts in a sequential way helped eliminate problems such as long range dependencies of entities. The only automated statistical metric where ProGeT did not outperform the baselines was TID and the only BERTScore metric where it did not outperform the baselines was Recall, but in both cases, it was only marginally behind the better performing baseline model. Furthermore, the best performing TID model was GPT-2 finetuned for 1 epoch and the best performing BERTScore Recall model was GPT-2 finetuned for 3 epochs, but we initially expected GPT-2 model performance to increase with increasing training epochs (since the 3 epoch variant outperforms the 1 epoch variant on most other metrics). We also expected the ProGeT model to outperform the GPT-2 baselines by a more significant margin across all evaluation metrics. We examined the generated outputs (some of which are shown in the appendix below) using human evaluation and found that the ProGeT generated screenplays were noticeably better than the baselines. We discuss this discrepancy between the quantitative and human evaluation in more detail in the following Analysis section.

Evaluation Results			
Evaluation Metric	GPT-2 (1 epoch)	GPT-2 (3 epochs)	ProGeT
HA BLEU 2	61.39	60.66	**63.19**
HA BLEU 3	36.69	36.87	**38.88**
HA BLEU 4	19.22	19.82	**21.15**
HA BLEU 5	9.37	9.83	**10.63**
TID	**1.498**	1.556	1.524
MS-Jaccard 2	35.387	36.25	**37.79**
MS-Jaccard 3	19.306	19.85	**21.20**
MS-Jaccard 4	9.85	10.26	**11.37**
MS-Jaccard 5	4.894	5.20	**5.92**
FBD 1-8	5.489	6.8665	**3.4051**
FBD 9-16	28.264	31.0739	**22.5328**
FBD 17-24	50.84	57.9144	**41.1473**

BERTScore			
Evaluation Metric	GPT-2 (1 epoch)	GPT-2 (3 epochs)	ProGeT
Precision	62.4	61.8	**64.3**
Recall	65.3	**65.7**	64.8
F1	63.8	63.7	**64.5**

5 Analysis

- **GPT-2 (1 epoch) compared to GPT-2 (3 epochs):**
  When comparing GPT-2 (1 epoch) to GPT-2 (3 epochs), we saw an incredible increase in screenplay like structure. The 1 epoch model did not output texts in the canonical screenplay format and did not include artifacts such as scene descreenplayions and the titles of the person who is talking. Instead, most of the outputs were blocks of text that sometimes contained one screenplay specific element such as a capitalized name or setting. It output screenplay artifacts rather infrequently and also generated garbage text a few times. We have attached an example of the GPT-2 (1 epoch) generated output as the first appendix item below. In the output we see what looks to be a YouTube channel's biography descreenplayion, something not found in the finetuning data, but likely part of the Huggingface pre-training dataset. This leads us to believe that 1 epoch of finetuning on our screenplay dataset is not enough to teach the model to generate screenplay like texts as the model is still strongly affected by exposure bias and generates outputs according to its pre-training distribution moreso than the finetuning distribution. With GPT-2 finetuned for 3 epochs, we saw basic screenplay structure elements like that of a two character dialogue with the titles above each character's lines in most of the output texts. This was a significant improvement over the text generated by GPT-2 (1 epoch). But upon further investigation of the 3 epoch finetuned model, we saw that most of the dialogue was incoherent and did not read like an actual conversation, including utterances that either contradicted each other or semantically made no sense (example of this in second entry in appendix).
- **GPT-2 (3 epochs) compared to ProGeT:**
  From the evaluation scores above, we did not think we would see a huge increase in coherency when comparing the outputs of the GPT-2 model finetuned for 3 epochs and the ProGeT model side-by-side since the scores only nominally increased. However, when looking at the generated texts, we saw that ProGeT managed to generate text with screenplay like structure, artifacts, while also generating more coherent dialogue. This structure was similarly present in the majority of GPT-2 (3 epochs) outputs, but there were far more screenplay artifacts such as camera angles and movement, changing settings, more topically coherent dialogue, that were much more prevalent in the ProGeT model outputs. Though we set ProGeT to generate outputs of 1024 tokens, the generated screenplays are consistently less than 1024 tokens in length. That being said, the outputs are much more semantically coherent than the GPT-2 model outputs and more representative of actual, human generated screenplays when evaluating the outputs side-by-side. The third entry in the appendix shows two examples of the ProGeT generated screenplay. Note that each of the 3 language model stages are printed. In the first stage (denoted as STEP_0), we see the output listing the most

important words from the for the screenplay as the character names. Then, the intermediate stage (STEP_1) generally includes another character name or setting identifier, before the final stage (STEP_2) generates a full coherent scene. This construction adheres with our aforementioned model setup as the first and intermediate stage vocabs only contain 20% and 25% of the total training vocab whereas the final stage vocab is the entire training vocab.

6 Conclusion

Our experiments found that the non-monotonic ProGeT model performed better than the auto-regressive GPT-2 baselines on the evaluation metrics (simple statistical metrics and BERTScore) as expected. Still, we were hoping to see more significant increases in the evaluation scores. Perhaps if we were able to generate a larger number of synthetic tests in the future, we could see more promising evaluation metric increases from the baseline models to the the proposed non-monotonic architecture. Even though the scores were only marginally better, we saw much more concrete evidence of improvement when evaluating the output texts side by side as human evaluators. We were able to generate over 1500 generated texts during the span of the project and were able to train over 4 iterations of pre-trained LM models. We learned that GPT-2 and BART are great building blocks for text generation, and can provide comprehensible neural generation with only a few fine-tuning examples. We also learned that a non-monotonic structure can combat exposure bias in long text generation, and that evaluation metrics might not be the best representation of how well the model outputs compare to human written examples. This leads us to believe that task specific evaluation metrics may provide a more nuanced and complete analysis of generated output quality.

The primary limitations of our work were the GPU requirements and the time allotted for experimentation. Training the ProGeT model whose results are shown here took between 2-3 days on an NC12 (dual GPU) machine on the VM. If we were able to, we would have liked to train ProGeT and our baselines for more epochs to optimize their performance, as well as generate texts that were longer than 1024 tokens in order to really test the benefit of stage wise vocabularies trained in a multi-staged non-monotonic LM model architecture. If the ProGeT model could carry on long range entity dependencies for a couple of pages of text, that would be incredible progress in the field of long text generation. Another limitation is that this model only trains on a specific subset of all the movie screenplays in the world, and could include foreign texts and a non monotonic approach similar to ProGeT but using different models for specific languages/integrating a translation component.

If time and compute power permitted, we would have liked to evaluate ProGeT's generated screenplays with more intermediate stages as well as our modifications to our GPT-2 implementation as the first stage in ProGeT for conditional screenplay generation (i.e. continuing a screenplay given an input screenplay sample). Finally, as mentioned above, we did not use the genre labels in the IMSDB dataset while training our models, but would like to test our models at generating unconditional text given genre labels as input features.

References

[1] Abigail See, Aneesh Pappu, Rohun Saxena, Akhila Yerukola, and Christopher D. Manning. Do Massively Pretrained Language Models Make Better Storytellers? In *arXiv*, 2019.

[2] Dustin Palea, Hongwei (Henry) Zhou, and Kapil Gupta. Transformer Bard: Music and Poem Generation Using Transformer Models. 2020.

[3] Neil Sinclair. Teaching BART to Rap: Fine-tuning Hugging Face's BART Model. 2020.

[4] Alessandro Calmanovici. Generating Rap Lyrics with AI. 2020.

[5] Charles Pierse. Film Script Generation With GPT-2. 2020.

[6] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. 2019.

[7] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence Level Training With Recurrent Neural Networks. In *arXiv*, 2016.

[8] Ehud Reiter and Robert F Dale. Building Applied Natural Language Generation Systems. 1997.

[9] Sean Welleck, Kiante Brantley, Hal Daume III, and Kyunghyun Cho. Non-Monotonic Sequential Text Generation. In *arXiv*, 2019.

[10] Martins Frolovs. Teaching gpt2 a sense of humor. https://github.com/mf1024/Transformers/blob/master/Teaching%20GPT-2%20a%20sense%20of%20humor.ipynb, 2019.

[11] Bowen Tan and Zhiting Hu. Progressive-generation. https://github.com/tanyuqian/progressive-generation, 2020.

[12] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. BERTScore: Evaluating Text Generation With BERT. In *ICLR*, 2020.

[13] The internet movie script database IMSDb. https://imsdb.com/. Accessed: 2021-03-05.

[14] Bowen Tan, Zichao Yang, Maruan Al-Shedivat, Eric P. Xing, and Zhiting Hu. Progressive Generation of Long Text. In *arXiv*, 2020.

A Appendix

GPT-2: 1 epoch (1024 tokens)

```
      EXT. BEDROOM - NIGHT


We're in the middle of the night, and we hear a knock at the door.
The door opens and a man comes in. He's dressed in a black
T-shirt and jeans, with a pair of dark sunglasses on.
His face is covered in blood, his hands are tied behind his back and he's
holding a gun to his head.


The man looks at us, then back at him. We see the blood on his face, but we
don't know what it is. It's not blood at all, it's blood from a gunshot wound
to the back of his neck, a bullet in his left arm, an artery that runs through
his body. He looks up and sees that the man has been shot twice, once by the
same man, the other time by a different man. This is the first time we've ever
seen this man in our lives, we can't help but think of him as a normal person,
not a serial killer. But we're not sure what to make of this guy, because he
looks like he was shot by someone who had no idea what happened to him or what
he had done to deserve this kind of treatment. And then, suddenly, he starts
to walk away from the scene, leaving us to wonder if this is just a coincidence,
or if he just wanted to get out of here and get some sleep, so that he could
get back to work. Then he walks away again, this time with blood all over his
shirt and pants, blood dripping from his mouth and nose. Finally, after a moment
of silence, someone comes up behind him, grabs his arm and pulls him away,
screaming,"Get out! Get out!" The man grabs the gun and shoots him once more,
killing him instantly. As we watch this, our eyes widen in shock, as we realize
that this isn't just another case of serial killers being killed by people who
didn't even know who they were. There's a lot more to this story than just the
fact that it happened in front of us. I'm not going to go into the details of
how this happened,just to give you an idea of what's going on in this case. If
you want to read more about it, you can find it here: http://www.youtube.com
/watch?v=hVhXV-X-YoA


And if you'd like to see more of my work, check out my website:

http://bit.ly/1XnXlXg
```

```
If you have any questions or comments, please feel free to leave a comment below,
I'd love to hear from you.

 -------------------------------------------------
```

GPT-2: 3 epochs (1024 tokens)

```
STEVE
     We all have Grandparents here at 0,000 feet
     altitude.

             SEP
     What went wrong -???


STEVE turns his head back to reveal MAGGIE with a funnel on her wrist that
would make us a little nervous.
She whirls a quarter and carries the grenade out.

            MAGGIE
     I don't know... It was a trick to
     rescue her and sort out the tragedy here
     for herself...


The shocked reporter crosses the room and rushes into the bureau.
He grabs Ellie's wrist - she holds on for dear life - and throws

it up on her mother's lap.

The journalist hurries back.

             STEVE
     Maggie.. What are you doing for a living?

            MAGGIE
     For nothing.
He awkwardly impales her head in her cuffs as he raises his gun.

          MAGGIE
    It's okay. Take care.

He places it and Ellie wiggles a stiff finger.

          STEVE
    Remember the double-murder of the two
    White Preppy?


          MAGGIE
    'Course I remember.


          STEVE

    Put down the truck.

    She runs. Behind the actress's back she can faintly
    lose her eye.


           MAGGIE
    Keep going, Maggie - try your best.


He works off the grenade until he sees Ellie's wrist at the
heel of her neck - undressing and kissing the light-headed
girl.

          MAGGIE

    All right, Maggie. It's okay.

She sweeps off the fire. IN AN AUCTION DAY

SIGENTINA watches ELLIE go to work making a giant commission.

SECOND SCREEN ON
```

ProGeT (1024 tokens)

```
STEP_0:

doorman

doorman



doorman

doorman


doorman

doorman

doorman doorman



moran


moran
```

```
STEP_1:

doorman

doorman
```

doorman

doorman

doorman

doorman

doorman doorman
stranger

moran

moran

STEP_2:

into the courtyard of a residential building.

INT. DOORMAN'S BUILDING, POLICE STATION

DOORMAN and WITNESSES stand in a silent area. The woman
is holding the purse.

INT. POLICE STATION, DOORMAN'S BUILDING

The Doorman speaks to them.

INT. DOORMAN'S BUILDING, POLICE STATION

The DOORMAN stands listening to the Woman speak to the
Police. The Woman talks directly to the police.
The POLICE doorman is shooed out of his job by a DOORMAN, who
approaches the stranger with a bundle.

INT. MORAN'S OFFICE

The Woman listens as though she are hearing through a glass.
She calls out "FOOD!" and when Moran hears the demand,
he looks over to the Woman as she carries the tray to her
husband, who is standing in the doorway.

He notices the surprise, then the shock, the embarrassment, that
he is watching the strange woman as if she is very frightening.

INT. POLICE ST

STEP_0:

letour

grady
lophakin

letour

lophakin
morocco

letour

STEP_1:

letour

grady
lophakin

letour

lophakin
glen

letour

```
STEP_2:
                                 (listening in)
                         Is this the same guy from the pictures?


                                     LETOUR
                         You can say that.

                                     GRADY
                         Yes, the same guy, the same Lophakin --
                         the man...

               He hears an opening and stops to listen. He stays silent.

               INT. REAR OFFICE

               CLOSE SHOT -- LETOUR

               He is trying to speak to a man. Lophakin is standing before him
               but the words are slow. Glen hesitates a second.

                                    CUT TO:


                              CLOSE SHOT --  GUN

               That is the same revolver that LETOUR saw in the first scene.
```

Actual screenplay from Test Set (1024 tokens):

```
     EXT. DEEP SPACE

          A dark screen is lit up by twinkling stars .

                          SON
           Baba?

                          FATHER
           Yes, my son?

                          SON
           Tell me a story .

                          FATHER
           Which one?

                          SON
           The story of home .
          A meteorite drifts into frame , heading towards tiny Earth off
          in the distance.

                          FATHER
           Millions of years ago , a meteorite
           made of vibranium, the strongest
           substance in the universe struck
           the continent of Africa affecting
           the plant life around it.
```

The meteorite hits Africa and we see plant life and animals affected by vibranium.

FATHER (CONT'D)
And when the time of man came , five tribes settled on it and called it Wakanda. The tribes lived in constant war with each other until a warrior shaman received a vision from the Panther goddess Bast who led him to the Heart Shaped Herb, a plant that granted him super human strength , speed, and instincts.

A visual representation of the five tribes emerges as hands from the sand animation, and we see them unite , and then break apart as conflict arises . Bashenga rises above the conflict and eats the Heart Shaped Herb, proceeding to unite the tribes .

FATHER (CONT'D)
The warrior became King and the first Black Panther , the protector of Wakanda .