# Classification Optimization via Adversarial Text Generation

Joshua Chung, Zain Khan, Anchit Sood
W266: Natural Language Processing with Deep Learning
Spring 2021

## Abstract

Text generation is a critical task in NLP that has been applied to many different problems. As of late, text generation capabilities have been greatly advanced by transformer architecture based models such as Open AI's GPT-2. Another popular form of neural generation has been utilizing generative adversarial networks (GAN). In this paper, we seek to apply a GAN inspired model framework as a means of improving on text generated by GPT-2. Specifically, we use a BERT model to filter text produced by our generator (GPT-2) as a discriminator of sorts. After doing so, data that is classified as "real" is fed back into the generator model to optimize generation. We apply generated text at different stages to the problem of text classification as a means of evaluation. We see that the addition of generated text slightly increases classification accuracy.

## Introduction

Our goal for this work is to utilize existing, state-of-the-art text generation models and apply a GAN-like framework to produce better text. We decided on applying the problem of text generation to TV show scripts. Televisions scripts come with an existing corpus of text, as well as a general tone which can be emulated in a generation task. In deciding on which television show to generate scripts for we looked for shows which had a cyclical plot structure, consistent characters, and no overarching plot that could cause generation complications. We decided on the kids show Spongebob Squarepants as it satisfied all our constraints, and had a relatively large corpus of episodes compared to other shows. While text generation work with this specific show is not new work, we seek to expand on the existing work done with our GAN-like structure[1].

While text generation in and of itself is interesting, we wanted to apply our text generation to a classification problem as a means of evaluation along with testing the application of generating fake data towards model training. To do this, we settled on

[1] https://www.reddit.com/r/learnmachinelearning/comments/jodejr/i_forced_a_bot_to_watch_over_1000_hours_of/

applying our text generation towards the classification problem of discerning an episode's rating from a script input. The application itself delves into using generated, fake data as a means to train a classifier. This aids in classification tasks where training data is sparse.

Benefits of a classification model such as this take shape in many forms. One example of such a benefit is providing script writers with a litmus test of how a newly written script may do critically. Thus, we aim to see if text generation is an effective method to optimize a rating classifier, and to what degree.

## Project Background

The idea of utilizing generated data to improve deep learning is not a new one[2]. As seen in the literature, utilizing GANs for synthetic data generation is a well documented approach in the field of computer vision. We hope to emulate this success in the field of natural language generation, specifically for the problem of rating classification.

As mentioned, GANs have shown good success in the computer vision domain, but have lagged behind in text[3]. Structured on two sub-models with cyclical updating, GANs introduce a technique of training a generative model through the use of the generator, which generates credible examples, and the discriminator, which classifies the examples produced by the generator[4].

Rather than implement a GAN that could potentially struggle with text, we looked to implement the adversarial nature of a GAN in a way that is better suited for text generation. Taking the inspiration of an adversarial approach, we hoped to implement GPT-2 to accomplish the task of text generation and BERT for text discrimination. GPT-2's diverse nature would allow for it to be trained on television transcripts and generate legible scripts, which could further be tuned through our adversarial approach[5]. Additionally, BERT is a pre-trained language model which can be used for classification tasks. There have been many works on improving BERT, leading to different variations of the bidirectional transformer[6]. Variations include transformers such as RoBERTa, BigBird, ALBERT, which are all created for their own respective objectives. Therefore, by implementing GPT-2 to generate scripts and BERT for classifying the scripts as real or fake, we were able to create this adversarial approach with a discriminator, which would filter out the scripts fed in.

[2] https://arxiv.org/pdf/1909.11512.pdf
[3] https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2737434.pdf
[4] https://arxiv.org/pdf/1406.2661.pdf
[5] https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
[6] https://arxiv.org/pdf/2004.03705.pdf

## Project Approach (Methods)

### Data

Initially, episode scripts were pulled from the Spongebob transcript wiki via a web scraping script. After producing some scripts with this data and self encoded tokens we opted to use an existing data source of every Spongebob script transcript found on kaggle[7].

We utilized IMDB's episode ratings and binarized the results into a measure of a "good" episode and a "bad" one. In order to split ratings into a binary measure we used the median of ratings as a threshold, values less or equal to the median were given a label of 0 whereas values greater than the median were given a label of 1. This was to ensure that we had a balanced distribution of labels.

The scripts consist of character dialogue as well as scene cues which are denoted by brackets. On top of the base scripts pulled from Kaggle, we added special tokens for the start of an episode, the end of an episode, as well as a rating token. The rating token serves to prompt GPT-2 to produce a rating token in its scripts, allowing us to generate scripts that come with a rating.

When using generated, synthetic data for training the BERT rating classifier, we initially planned to train our model on a purely synthetic dataset. Though after further literature review, we heeded the observations of work that has come before us [8] and opted for a hybrid dataset of real and synthetic data points. There are two versions of the synthetic dataset. The first version is the data that was first generated by GPT-2 and then filtered by our BERT filter classifier. The second version is data that is generated by GPT-2 after it had been trained on both real episodes and episodes that were false-positives when classified by our BERT filter model. The hope is that the second version of the synthetic dataset is higher quality than the first version as it is generated in a pseudo-GAN style.

### Models

In designing our pseudo-GAN, we track varying model stages in which we test the results of the IMDB ratings classifier using various training data. The model used for classification is BERT as distributed by Hugging Face, and our implementation of GPT-2 utilizes the python package gpt_2_simple.

[7] https://www.kaggle.com/mikhailgaerlan/spongebob-squarepants-completed-transcripts
[8] https://arxiv.org/pdf/1909.11512.pdf

Our model checkpoints are as follows: BERT training on real episode transcripts, BERT using GPT-2 generated transcripts that have been filtered via another BERT filtering model, and lastly BERT using GPT-2 generated scripts that is trained with real episode transcripts as well as generated scripts that were deemed “real” by the filtering BERT (in a pseudo-GAN fashion).

Before going into the various model checkpoints, we will delve into the specific architecture of our BERT models. The two being the BERT model for rating classification and the model used for generated script filtering. Both use a similar supervised learning based approach. The pre-trained BERT model, as distributed by Hugging Face, has a large set of training parameters which is why fine-tuning the model is infeasible due to GPU memory constraints. Instead of training the gradients for BERT, we took the BERT CLS token embedding and put that into a 2 layer fully-connected network for training. This aids in speeding up runtime as well as avoiding BERT taking up our entire allocated memory. Our filtering model employs a similar architecture. In the case of the filtering model, rather than use the regular 0.5 probability cutoff to assign class real, we opt for a cutoff of 0.7 so that filtering is completed at a higher level of scrutiny.

The initial classification model uses only the preexisting episode transcripts from the kaggle dataset in a train-validation split. The model serves as a baseline to compare our text generation aided classification models to. This model is tested on the entire dataset, which is not an issue since the model does not seem to even grasp the relation between the CLS token embedding and the IMDB rating. We also use the entire dataset for testing as we will use the entire real episode dataset when testing the other BERT rating classifiers. We will refer to this model schema as Naive BERT.

Our next model builds on the previous one by implementing the first stage of our pseudo-GAN. As mentioned in the background section, our adversarial approach uses GPT-2 as our generator and a filter BERT as our discriminator. Thus, in this stage we introduce our BERT discriminator. In this instance, the BERT model used for rating calculation is trained on a subset of generated scripts that is deemed as real text by the filter BERT (false positives). The filter BERT is trained on both real scripts and fake scripts and deems whether an inputted script is *real* or fake. We will refer to this model schema as BERT with Filtered Text Generation.

Our last iteration of our model is the final step in building our pseudo-GAN. The final model includes the steps from our previous model, but with an additional step. Instead of feeding the filtered data into the classification BERT model, GPT-2 is fine-tuned once again on data that is classified as *real*. This allows us to have a cyclical training process of sorts, similar to the indirect training by the discriminator in a true GAN[9]. In our approach, the filter BERT as our discriminator is cyclically fine-tuning the

[9] https://theaisummer.com/gan-computer-vision/#vanilla-gan-generative-adversarial-networks-2014

GPT-2 generator with more data. In this implementation we can conduct multiple cycles of generation and fine-tuning and then use the final text generated as an output.

Generally, neural network tasks that are complex require a high amount of data to work on. The issue with having such a low amount of data is that the model will never learn the relationship between the features and the target variable. There are only about 500 Spongebob episode transcripts available based on the amount of episodes that have been released. While this is a large amount of episodes for a television show to have, this is a relatively small source of fine-tuning data. Thus, we use our generated data that has been vetted by a discriminator model as a means of increasing the amount of fine-tuning data.

The goal here, similar to the previous model approach, is that with a better quality of generated scripts classification accuracy can increase. We will refer to this model schema as BERT with pseudo-GAN, *n* cycles run.

One note on model testing is that we don't necessarily expect every iteration of our model process to produce better classification results. We hope that at the very least text generation quality is improved. Though, this is an experiment on the efficacy of utilizing generated data to improve classification accuracy, so we observe which method ends up producing the best results.

## Approach Shortcomings

There are certain shortcomings in the approach as is that we must acknowledge. Firstly, a flaw in our filter model training is that we assign generated scripts a class 0. In doing so, we train the model on generated scripts with varying degrees of geneity. The goal of the training is for the filter model to discern between generated scripts that are obviously fake and ones that look real. It is possible that in the training process we assign a class 0 (fake data) to a script that would be classified as real. This shortcoming is mitigated by training on a wide set of generated data. The hope is that the potential “real” looking scripts that are in the training set make it such that the model has a higher threshold for deeming a script *real*, and end up producing a higher quality of generated text.

Another issue can be error propagation. If our BERT filter classifies a false positive example that contains errors that make rating classification infeasible (e.g. missing IMDB rating, malformed text), then with the second training of GPT-2 this error could be reinforced and be more apparent in the newly generated scripts. This is the reason why our pseudo-GAN approach failed.

Lastly, a big shortcoming of our implementation is that we utilize BERT for classification tasks. BERT has a maximum token limit of 512 so we effectively lose chunks of our data. We initially experimented with using BERT variations that can handle larger amounts of tokens such as Longformer, and BigBird. In initial testing, both variations showed poor performance results as compared to that of BERT, so despite the token limit we stick with BERT in favor of its optimal performance.

## Results

The table below shows the results of our models. For this project, the metric for evaluation is the performance of the BERT rating classifier. We measure this performance of the model via its precision, recall and F-1 score. As mentioned before, the objective is to measure if generated, fake data can aid in text classification. On top of that we see how varying methods of generating text in different stages of the pseudo-GAN implementation can affect classification performance. Thus, the precision, recall, and F-1 score provide a holistic measure of how each classification model is doing given varying generated data.

| **Model Schema** | **Precision** | **Recall** | **F-1 Score** |
|---|---|---|---|
| Naive BERT (Baseline) | 0.45 | 0.45 | 0.45 |
| BERT Filter | 0.48 | 0.48 | 0.48 |
| BERT with Filtered Text Generation (v2) | 0.50 | 0.50 | 0.49 |
| BERT with pseudo-GAN, 1 cycle run (v3) | n/a | n/a | n/a |

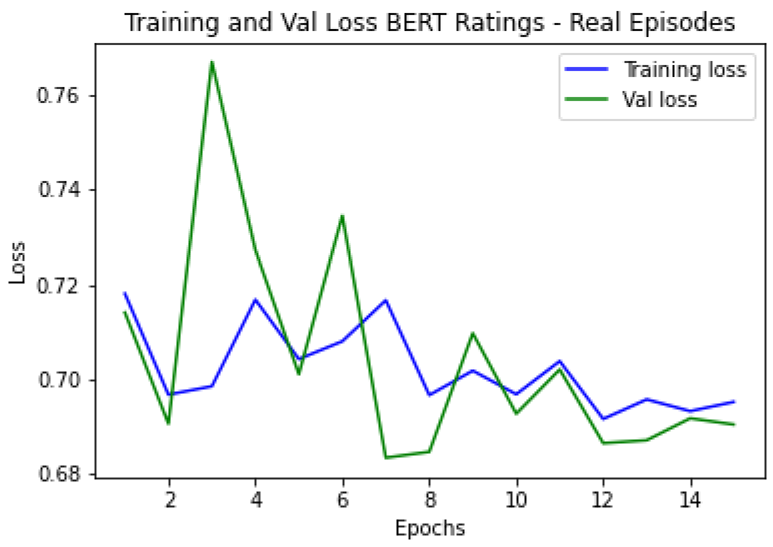
Based on the model performances above, we see that the addition of generated data that had been filtered slightly increased precision, recall, and the F-1 score. This indicates that the addition of generated data aids in the performance of classification, if only by a small amount.

Unfortunately, the fully realized pseudo-GAN did not overcome some of the shortcomings we foresaw. Upon fine-tuning the GPT-2 generator model with both real scripts and our generated scripts that had passed the filter, the quality of generated scripts showed a major decline. Scripts generated by the final GPT-2 model in v3 showed common issues relating to overfit generators. Issues with the generated scripts included repetitive dialogue, lack of a rating token, and incoherent text. These were issues we had seen before when originally using GPT-2 with poor data sources. Seeing as these issues were not present in the previous GPT-2 model, we can attribute them to the addition of generated scripts in the fine-tuning process. Our prior GPT-2 generator only used real episode transcripts and rarely showed any of these issues.

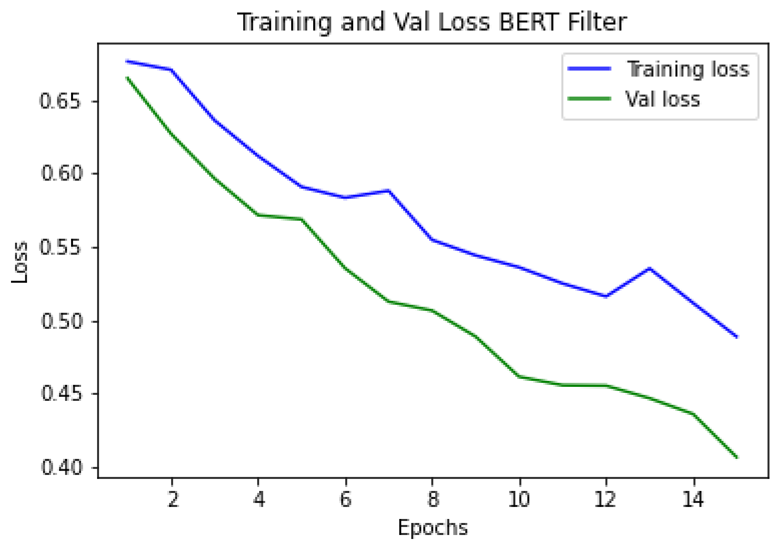
## Classification Model Loss Evaluation

To further evaluate the results of our project, it is important to take a look at the training and validation set loss for the different classification models.

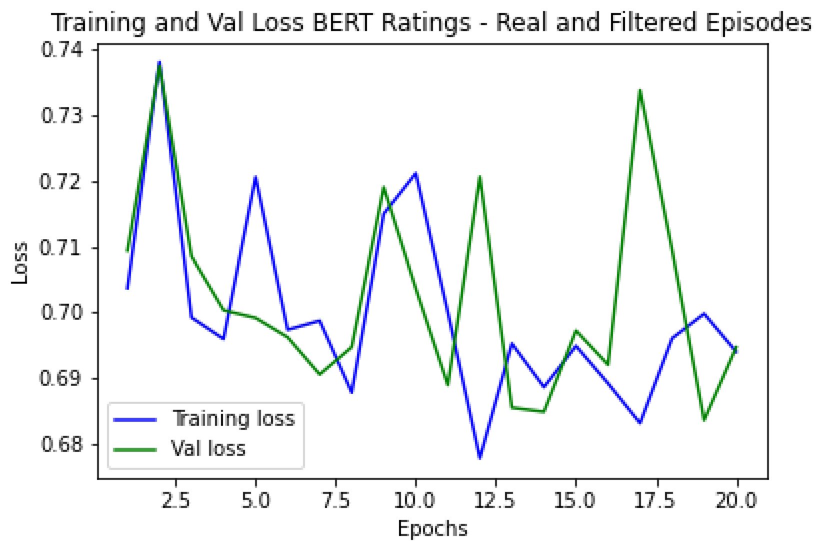
Below, we see the training and validation loss for the BERT ratings classification model on real episodes. This model utilizes the real Spongebob scripts from the Kaggle dataset as the input and outputs the predicted ratings of the scripts. Over the course of 15 epochs, we see a significant decrease in the training and validation loss. Looking specifically at the training data, it is evident that the model is able to predict the rating of the real episodes much more accurately by the 15th epoch. Looking at the validation dataset, we see it has a slight increase in loss around the 6th epoch and is negatively correlated with the training dataset loss from the 10th epoch onwards. However, the validation loss doesn't increase to the extent where we would believe our model is overfitting.



In the visualization below, we see the training and validation loss for the BERT filter classification model. The intention of this model was to input the real scripts & generated scripts and output whether the scripts were from real or fake episodes. Over the course of 5 epochs, we see that the training and validation loss decreased significantly. By the 5th epoch, we were able to attain a loss of roughly 0.2 for both the training dataset and validation dataset. Rather than doing 15 epochs here, it was important to do a lower amount to avoid the model overfitting. We see that there is correlation between the training loss and validation loss.



In the below visualization, we see that the training and validation loss were quite all over the place. This data comes from the BERT ratings classifier with the real and augmented data. We see an evident correlation between the training and validation loss of this classifier.



## Conclusion

Our struggles in fully implementing our intended pseudo-GAN show the errors which can arise in attempting to develop a new algorithmic process. We saw that the intended model architecture seemed perfectly sound in theory, but when it came to implementation, we could not overcome predicted shortcomings. Our work demonstrates an attempt to apply a researched application of adversarial networks that has worked well in computer vision to the field of natural language. Naturally, the work could not be laterally applied to the domain of text. Even with the measures we took, we fell into pitfalls that stopped us from applying our fully realized pseudo-GAN to a classification problem.

Despite the lack of a proper pseudo-GAN implementation we were able to show the benefits of utilizing generated data to improve classification performance. We see that in our v2 model which utilizes generated text, that recall, precision, and accuracy improve by 0.04. While these gains are minimal, they still show that generated text can aid in classification performance.

## References

- https://arxiv.org/abs/1803.07133
    - <Neural Text Generation: Past, Present and Beyond>
- https://arxiv.org/pdf/2004.03705.pdf
    - <Deep Learning Based Text Classification: A Comprehensive Review>
- https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
    - <Language Models are Unsupervised Multitask Learners>
- https://arxiv.org/pdf/1406.2661.pdf
    - <Generative Adversarial Nets>
- https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2737434.pdf
    - <Evaluating Generative Models for Text Generation>
- https://arxiv.org/pdf/1909.11512.pdf
    - <Synthetic Data for Deep Learning>
- https://www.reddit.com/r/learnmachinelearning/comments/jodejr/i_forced_a_bot_to_watch_over_1000_hours_of/
    - Reddit post with previous work in Spongebob Script Generation
- https://www.kaggle.com/mikhailgaerlan/spongebob-squarepants-completed-transcripts
    - Data Source
- https://theaisummer.com/gan-computer-vision/#vanilla-gan-generative-adversarial-networks-2014
    - Article on GANs