Using The Sequence GAN To Backpropagate Through Discrete Data

Sumit Minocha, Ryan Mui, and Matt Linker
Stanford Computer Science
{sminocha, ryanmui, mslinker}@stanford.edu

Abstract

General Adversarial Networks (GANs) have shown significant success in the domain of images, however, the same achievement has not come to fruition in the realm of text, mainly due to GANs' difficulties with handling discrete data. In our paper we examine the Sequence General Adversarial Network (SeqGAN), one of the most promising workarounds to this problem presented by Yu et al. (2017). Implementing this GAN variation, we then attempt to generate text by piping in the Stanford Sentiment Treebank (SST) dataset and analyzing the sentiment of the generated output. In an attempt to train the SeqGAN on datasets with different sentiment skews, we discover that partitioning the SST significantly reduced the performance of our model as it vastly reduced the size of the vocabulary. We also encounter evolving levels of sentence coherence (sampling from the generated text at various times) as the model trains, and we explore potential reasoning for this being the case.

1 Introduction

In 2014, Ian Goodfellow and his team introduced a method for training generative models called Generative Adverserial Networks (GANs) (2014). GANs have since garnered quite a bit of attention in the computer vision community as they have demonstrated much promise applied to image generation; however, the same cannot be said for text generation.

A typical GAN structure involves two neural networks, a discriminator D and a generator G. D is trained to decide whether input data is fake or real, while G is simultaneously trained to generate data that fools D into thinking that fake data is real. The propagation of gradients back from the discriminator through the generated samples to G (a step which is critical to the proper functioning of a GAN) relies on the generated data being continuous (M. Kusner, 2016). However, text data exists as sequences of discrete items and therefore is inherently not continuous. The backpropagated gradients can't move through the non differentiable discrete data, so they zero out. Ultimately, the reason that GANs have difficulty generating sequences of discrete tokens is as follows. The generator is initialized with a random sample, and then is deterministically transformed according to the model parameters (I. Goodfellow and Bengio, 2014). In the case where the generated data is continuous, the discriminator loss informs updates to the generator's parameters; however, in the case where the generated data is discrete, these updates from the discriminator will point to a word that most likely doesn't exist in vocabulary space (Lantao Yu, 2017).

In this paper, we aim to reimplement and explore the inner workings of the Sequence GAN with policy gradient, which is Lantao Yu and his team's solution to the above problem (2017). This model represents a sequence generation procedure as a sequential decision making process, where Monte Carlo (MC) search is incorporated into the policy gradient to approximate and directly train the generative model's policy, thus avoiding the differentiation difficulty for discrete data mentioned previously. The SeqGAN architecture as well as the role this updated policy gradient approach plays in the model is elaborated on in Section 3.

Our inputs include text samples extracted from the Stanford Sentiment Treebank dataset (SST) and a couple subsets of the dataset. We compare our outputs generated by the SeqGAN to a Maximum Likelihood Estimate baseline model qualitatively using human evaluations. Our experiments show that while the MLE implementation outputs words that contain more sentiment meaning, it still lacks the ability to learn the nuances of text phrases. Our SeqGAN model on the other hand is capable of generating some fluid phrases. Additionally, we notice that when training on the dataset subsets, our implementation produces significantly worse results as training iterations past 6,000 increase.

2 Related Work

Our work is originally inspired by the research of Ian Goodfellow et al., in which he demonstrates the success of the GAN learning architecture on image input types (2014). From his novel paper, many subsequent extensions of the GAN have been explored, including conditional GANs (cGANs) (M. Mirza, 2014) and inverted conditional GANs (IcGANs) (G. Perarnau, 2016). Additionally, others have diverged from image generation, focusing on the problem of representing discrete data as differentiable in order to apply GANs to text (Xiao, 2017; Lantao Yu, 2017; M. Kusner, 2016).

As alluded to in the Introduction, while Goodfellow's work on GANs kicked the adversarial wave off, the work of GANs for text generation has been less prolific. The difficulty of GANs operating on discrete data come from the discriminator only being able to assess a complete sequence instead of continuously updating discrete tokens. This problem is particularly critical for text generation, as words are obviously discrete. This issue has represented perhaps one of the most significant barriers in recent years to successful GANs on text models.

Yu et al (2017) and Kusner et al (2016) have come up with two of the most popular solutions that address this problem. On the other hand, Bowman et al (2016), and Li et al (2018) move outside the realm of GAN-based text generation implementations and try to achieve comparable results.

Our project implementation is primarily based on Yu et al.'s Sequence GAN (2017). Instead of looking only at existing generated tokens, Yu uses Monte Carlo simulation to consider the tokens that will still be generated. This allows for estimation of intermediate states, from which the team is able to form gradients and successfully backpropagate through discrete data. Kusner et al. propose another promising workaround that instead uses the Gumbel-softmax distribution, which is a continuous approximation to a multinomial distribution parameterized in terms of the softmax function (2016). A differentiable approximation of the discrete data is attained by sampling from this distribution, which is then used to train GANs on sequences of discrete tokens. The proposed LSTM is sampled through, and takes as input a sample pair, which effectively replaces the initial cell and hidden states. From this sample the generator constructs a sequence by successively feeding its predictions as input to the following LSTM unit.

Bowman et al. (2016) introduce a highly cited non-GAN rnn-based variational autoencoder generative model that incorporates distributed latent representations of entire sentences, which allows for explicit modeling of holistic properties of sentences such as style, topic, and high-level syntactic features. Recurrent neural network language models (rnnlms) represent some very state of the art research in the field of unsupervised generative modeling for natural language sentences. Specifically, the model that is outlined is an extension of the rnnlm that, while effective, is not able to learn a vector representation of the full sentence and thus experiences mixed success in the performed experiments (2016).

In Li et al. (2018), the authors introduce a non-GAN system that does not require a priori knowledge of positive and negative attribute words, requiring only a tagged corpus of positive and negative reviews. Under this methodology, sentiment-leading words are identified and weighted by comparing their frequency in positive and negative reviews. From there, reviews are analyzed and attributes with a strong sentiment are extracted. Next, the opposite-sided sentiment corpus is analyzed with an RNN to identify and generate opposite sentiment attributes found in similar contexts, which are then inserted into the original sentence. This method significantly outperformed previous systems of sentiment transformation for text, including existing GAN-based methods, but required quite a robust dataset with corresponding, roughly oppositely attributed samples. Finally, there are several open-source implementations of GAN variants online (Lantao Yu) and (codekansas). We utilize and make modifications to the latter implementation.

3 Data and Methods

Our goal with this work was to generate sample sentences of text via Lantao Yu et al.'s SeqGAN implementation (2017). Along these lines, we also wanted to test a couple hypotheses under the umbrella of this text generation SeqGAN approach. The first hypothesis is, point blank, can we use a SeqGAN architecture to interpret discrete data and output generated text? Second, how would changing the training dataset's distribution of sentiment affect the performance and output of our SeqGAN in its task of generating text?

3.1 Dataset

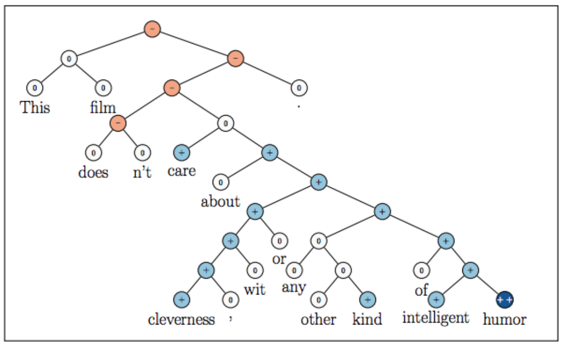


Figure 1: Example parse tree sentence in the Stanford Sentiment Treebank

Sentiment Score	Meaning
0	Very Negative
1	Negative
2	Neutral
3	Positive
4	Very Positive

Table 1: Sentiment Ratings

Dataset	Sentiment	Examples
Pos	0-1	3611
Neg	3-4	3311
Full	0-4	8545

Table 2: SST Data Splits

We chose to test our implementation on the Stanford Sentiment Treebank dataset (SST)(Richard Socher and Potts, 2013), which consists of 11,855 sentences labeled with a sentiment score between 0 and 4 (corresponding meaning of each sentiment rating is described in Table 1). In addition to using the full dataset, we also constructed derivative datasets containing just positive sentiment sentences (scores of either 3 or 4) and just negative sentiment sentences (scores of either 0 or 1). These splits were made in order to test our hypotheses discussed.

3.2 SeqGAN Architecture

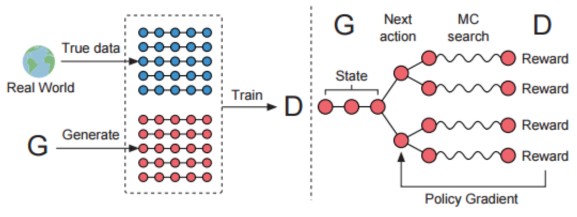
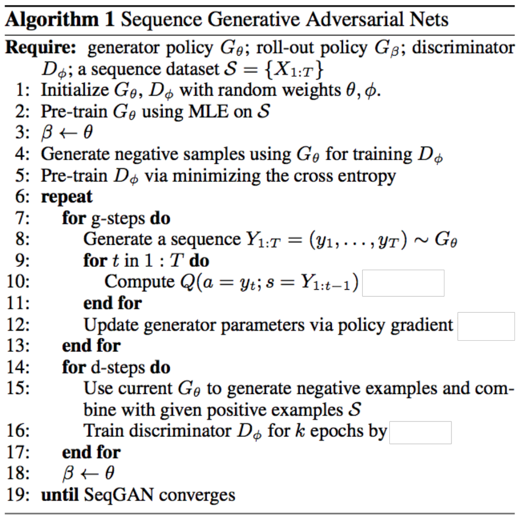


Figure 2: SeqGAN illustration. Left: D is trained over both the real data as well as the generated data from G. Right: G is trained by policy gradient where the final reward signal is provided by D and is passed back to the intermediate action value via Monte Carlo search (Lantao Yu, 2017)



Algorithm 1 Sequence Generative Adversarial Nets

```
Require: generator policy $G_\theta$; roll-out policy $G_\beta$; discriminator
    $D_\phi$; a sequence dataset $\mathcal{S} = \{X_{1:T}\}$
 1: Initialize $G_\theta$, $D_\phi$ with random weights $\theta, \phi$.
 2: Pre-train $G_\theta$ using MLE on $\mathcal{S}$
 3: $\beta \leftarrow \theta$
 4: Generate negative samples using $G_\theta$ for training $D_\phi$
 5: Pre-train $D_\phi$ via minimizing the cross entropy
 6: repeat
 7:   for g-steps do
 8:     Generate a sequence $Y_{1:T} = (y_1, \dots, y_T) \sim G_\theta$
 9:     for $t$ in $1 : T$ do
10:       Compute $Q(a = y_t; s = Y_{1:t-1})$
11:     end for
12:     Update generator parameters via policy gradient
13:   end for
14:   for d-steps do
15:     Use current $G_\theta$ to generate negative examples and com-
        bine with given positive examples $\mathcal{S}$
16:     Train discriminator $D_\phi$ for $k$ epochs by
17:   end for
18:   $\beta \leftarrow \theta$
19: until SeqGAN converges
```

Figure 3: SeqGAN Algorithm

Following (Lantao Yu, 2017), we implement a SeqGAN to generate output text. Given a dataset of structured sequences, we train a generative model G_θ, parameterized by θ and a discriminative model D_ϕ, parameterized by ϕ.

G_θ is trained to produce a sequence $Y_1 : T = (y_1, ..., y_t, ..., y_T), y_t \in Y$, where Y is the vocabulary of candidate tokens. This problem is approached as a reinforcement learning problem where the state s is the current produced tokens $(y_1, ..., y_{t1})$ at timestep t, and the action a is the next token y_t to select. This results in a policy model $G(y_t|Y_{1:t1})$ which is stochastic, and a state transition that is deterministic after an action has been chosen.

D_ϕ is trained by providing positive examples from real sequence data and negative examples from sequences generated from the generative model G_θ. At the same time, the generative model G_θ is updated by employing a policy gradient and MC search on the basis of the expected end reward received from the discriminative model D_ϕ. The reward is estimated by the likelihood that the discriminative model D_ϕ is tricked.

Once fully trained, we can use G_θ to sample output text.

3.3 Evaluation Metric

To evaluate our results, we compare against a baseline Maximum Likelihood Estimate model that assigns a sentiment weight to each target word based on labeled training data. Namely, this baseline is built around a Laplacian estimate of relative sentiment frequency for a given word. We first preprocess the input data by building a mapping of sentiment to relative word frequency, where each word's "score" is a Laplace-smoothed measure of the portion of times the word is observed with a given sentiment. In the baseline implementation, the model simply returns the set of n words with the highest score with a given sentiment flag (on a 0-4 scale), making the output of the baseline deterministic.

This creates an underlying weighted vocabulary, from which sentences would be generated. Each sentence is generated to maximize a particular log likelihood function, using the earlier defined weights. This baseline was chosen because this was the same baseline used by Yu's team, and was explained by the team to be a strong baseline choice (Lantao Yu, 2017).

Since we weren't performing a language translation task in the same way that Yu et al. (2017) did, we didn't use the same bilingual evaluation understudy (BLEU) metric that allowed Yu and his team to quantify the quality of the generated text. For our project, we leaned more toward using a human-based evaluation metric, in that we roughly intuited the performance of our model given its generated samples. We discuss drawbacks of this evaluation metric as well as potential alternatives in sections 5 and 6.

3.4 Experimental Setup

The first step in our approach of implementing Lantao Yu et al.'s SeqGAN was splitting the SST dataset as described in Section 3.1. Theoretically, these splits would allow us to gain insight into our second hypothesis of whether changing the training dataset's sentiment distribution would influence the SeqGAN's output.

Our experimental approach then involved testing the SeqGAN implementation on the full SST dataset, the positive subset, and the negative subset, and comparing these results to the MLE baseline. In order to get our SeqGAN to properly compile for each of these datasets however, we needed to build a dictionary with each subset's unique characters (and pipe that into the model), so the SeqGAN could understand what character distribution to sample from.

Pending the results from each of these three major experiments, we would also get a sense for the ability of SeqGANs to generate textual output (our first hypothesis).

4 Results

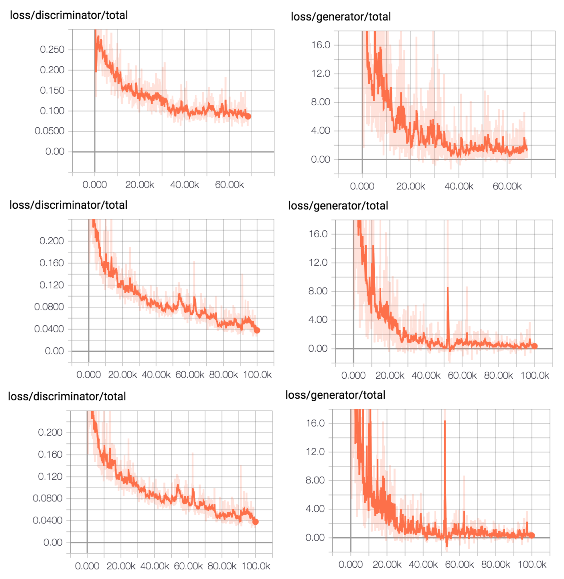


Figure 4: Moving from top to bottom: the discriminator and generator loss from training on the positive, negative, and full SST datasets, respectively.

Generated Text	Dataset	Iters.	Model
captivating splendid	-	-	MLE
exceptional	-	-	MLE
is a comedy	Pos	6k	Seq
in and and and th	Pos	60k	Seq
movie is a film	Neg	6k	Seq
ine it 's not th	Neg	60k	Seq
a movie is a movie	Full	6k	Seq
intelligent and a movie is a success	Full	60k	Seq

Table 3: Example sampled text, with corresponding training dataset as well as time of sampling, in terms of iterations.

Notes on implementation: Each of the three experiments (reference Section 3.4) took about a day and a half to train up to around $60,000$ iterations. To get a sense for how the generated output evolved over the course of training, we sampled at $6,000$ as well as at $60,000$ iterations.

First, training our SeqGAN on the positive subset of the SST, we saw the model output generated text comprised of seemingly coherent sounding phrases about $6,000$ iterations in, but comprised of mostly articles and incomplete words $60,000$ iterations in. The positive sentiment seemed missing (samples were neutral instead of having any particular sentiment skew) during the sample from $6,000$ iterations, and entirely destroyed during the sample from $60,000$ iterations. See Table 3 for example text samples.

Training our SeqGAN on the negative subset of the SST, we saw a similar outcome, where the model output seemingly coherent fragments of text $6,000$ iterations in, but output mostly articles and incomplete words $60,000$ iterations in. The negative sentiment seemed largely missing (samples were neutral instead of having any particular sentiment skew) during the sample from $6,000$ iterations, and again entirely destroyed during the sample from $60,000$ iterations. See Table 3 for example text samples.

Training our model on the full dataset, however, yielded different but related results. Specifically, the model output seemingly coherent fragments of text both $6,000$ and $60,000$ iterations in. While any particular sentiment was absent at $6,000$ iterations, the samples from $60,000$ iterations seemed skewed in the positive direction.

Our baseline, which during this particular run we configured to output generated text of length 10 with an extremely positive sentiment (sentiment score of 4), generated the following output: 'breathtaking universal thoughtful kinnear first-class breathtakingly dazzling tears'.

5 Analysis

As a quick observation, we first saw that both the discriminator and generator appeared to be training at the same rate (reference the discriminator and generator loss curves from Figure 4). This conveyed to us that the GAN was operating as desired during training; namely, both players were evenly matched and none was overpowering the other.

Looking at Table 3 and reading into some of the differences between the baseline generated text versus the generated text from our various SeqGAN experiments, we notice that our MLE implementation appears to do a better job at generating significant words, while our SeqGAN appears to be learning some of the nuances of generating phrases (including for example punctuation and grammar). In other words, the baseline model delivers strong results with respect to true sentiment of a given string, but performs very poorly in terms of actual sentence coherence and grammar. Reasoning a bit further, this is likely because our MLE model does not take into consideration the previously occurring word (the SeqGAN model does in fact do so) and instead explicitly counts up instances of occurrence of particular word-sentiment taggings.

Another major takeaway, alluded to in Section 4, is that while training our SeqGAN on both our positive and negative SST subsets, the generated text after $6,000$ iterations is more coherent than the text generated after $60,000$ iterations. This is most likely due to an overfitting problem, especially given that GANs are known to have limited variance in the range of outputs, which might explain the repeated occurrence of the same articles and incomplete words in the text generated after $60,000$ iterations. Considering the results of the positive and negative experiments in the context of the results from the full SST experiment, we see that the datasets for the experiments where we see this coherence problem are derivations (importantly, subsets) of the full SST, which almost certainly further compounded the issue. Specifically, splitting the data in this way ultimately reduced the total vocabulary available during training. With a fraction of the vocabulary of the full SST dataset, the negative and positive experiments converged to nonsense by $60,000$ iterations while the full experiment did not.

The last main discussion point involves the drawbacks of the evaluation mechanism we chose for this project (additionally elaborated on in the following 6 section). While human evaluation might be ideal if we wanted to get as close as possible to a human oracle, the human evaluation metric is very subjective as it is not standardized. For example, asking people to assign sentiment or asses the coherence of a phrase they see is neither practical nor guaranteed to be accurate. Lantao Yu et al., in order to simulate real-world structured sequences, considered a language model to capture the dependency of the tokens (2017). They use a randomly initialized LSTM as the true model (i.e. the oracle) to generate the real data distribution. However, this was not quite feasible for our experimental approach given our various time and compute constraints.

6 Conclusion and Future Work

To conclude, using our SeqGAN implementation, we were able to generate phrases of varying length by sampling sequences from the trained SeqGAN model, thus confirming our first hypothesis. Regarding our second hypothesis about changing the training dataset sentiment distribution, we saw that changing the training dataset definitely influenced the sentiment of the generated output text. In fact, revisiting the functionality of a conditional GAN from section 2, one can reason that this performance closely mirrored that of a conditional GAN (M. Mirza, 2014).

There are abundant opportunities for future work on this problem. We outline 3: Ablation, Evaluation mechanisms (Synthetic Data, BLEU), and Conditional GANs.

A robust ablation study might be helpful in order to test what features of our model or algorithm most optimize for high performance (in this case high quality generated text). Ablation options could include adding or subtracting various layers in the generator or discriminator nets. Alternatively, one could even experiment with different word vectorization techniques like GloVe or Gumbel Softmax (M. Kusner, 2016) that solve the discrete data backpropagation problem in a manner different from Yu et al.

Referencing the discussion in section 5 on the drawbacks of our evaluation mechanism, one could swap the mechanism for either a Synthetic Data or BLEU metric. Yu et al. implemented a synthetic data approach, where they performed extensive experiments based on synthetic and real data, which were conducted to investigate the efficacy and properties of the proposed SeqGAN. In their synthetic data environment the generative model was partially fed with its own synthetic data as a prefix (observed tokens) rather than the true data when deciding the next token in the training stage (Lantao Yu, 2017). As mentioned in the previous section, this evaluation mechanism closely approximated real-world structured sequences and was considered by Yu and his team to be most representative of a human oracle that manually labels the data (2017).

Lantao Yu et al. also used the BLEU comparison mechanism (frequently used in machine translation applications) to quantitatively assess how accurate their text generation was at translating between two languages . While our implementation didnt involve such an inter-language translation, some efforts could have been made to adapt BLEU to our task. BLEU, however, would assume that the goal of the text generation was to as closely as possible replicate a particular phrase, which might not always be the case (Lantao Yu, 2017).

Our last suggested trajectory for future work involves adding functionality to the SeqGAN that would allow it to generate text conditioned on sentiment. This functionality could be added according to Mirza and Osindero's paper published in 2014 (2014), and would be accomplished by feeding an additional input into the discriminator and generator as an additional input layer.

It is quite clear that there is vast potential for GANs to make an impact in improving text generation techniques, whose applications can be further extrapolated to improving language models, machine translation, summarization, and captioning (Xiao, 2017). High level motivations for research pursuits in this area are: 1) GANs in general are an extremely recent learning architecture that allows for representation and manipulation of high dimensional probability distributions, and 2) success in this area has potential to assist in semi-supervised learning and to develop systems, which can communicate as fluently as humans (S. Ramasundaram, 2016).

Code Repository

https://github.com/sminocha/text-generation-GAN

References

codekansas. https://github.com/codekansas/seqgan-text-tensorflow.

B. Raducanu J. Alvarez G. Perarnau, J. van de Weijer. 2016. Invertible conditional gans for image editing.

M. Mirza B. Xu D. Warde-Farley S. Ozair A. Courville I. Goodfellow, J. Pouget-Abadie and Y. Bengio. 2014. Generative adverserial nets. 1.

He He Percy Liang Juncen Li, Robin Jia. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer.

Jun Wang Yong Yu Lantao Yu, Weinan Zhang. https://github.com/lantaoyu/seqgan.

Jun Wang Yong Yu Lantao Yu, Weinan Zhang. 2017. Seqgan: Sequence generative adversarial nets with policy gradient.

J. Hernandez-Lobato M. Kusner. 2016. Seqgan: Sequence generative adversarial nets with policy gradient.

S. Osindero M. Mirza. 2014. Conditional generative adversarial nets.

Jean Y. Wu Jason Chuang Christopher D. Manning Andrew Y. Ng Richard Socher, Alex Perelygin and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank.

O. Vinyals A. M. Dai R. Jozefowicz S. R. Bowman, L. Vilnis and S. Bengio. 2016. Generating sentences from a continuous space.

S.P. Victor S. Ramasundaram. 2016. Gans for sequences of discrete elements with the gumbel-softmax distribution.

X. Xiao. 2017. Text generation using adverserial training.