TweetGAN: Natural Language Generation using Tweets

Connor Hainje and **Kathryn Leung**
Princeton University, Department of Computer Science
{chainje,kl22}@princeton.edu

Abstract

Today, the generative adversarial network is one of the most important and fervently studied networks for image generation. Beyond image generation, however, it has been extended to natural language generation, where its study is of great interest. One of the most famous models is SeqGAN, which we use throughout this study. In this study, we explore the performance of SeqGAN on a previously unstudied form of natural language: tweets. Tweets are a more informal format for written text, so the added "loose-ness" of the grammatical structures at play may allow GANs to produce more convincing generated samples. We found, however, that this "loose-ness" was typically correlated with wider-ranging subject matter, leaving the networks somewhat confused. We also found that our GAN performed poorly during adversarial training, meaning it did not perform much better than a maximum likelihood estimation-trained generative model. Even still, our results are promising, as the addition of larger datasets and further hyperparameter tweaking seem to be the keys to a more stable and much-improved model.

1 Background

Text generation has long been a heavily researched problem in the field of natural language processing, and its solution is far from trivial. One interesting approach is to use a generative adversarial network, a special kind of neural network architecture. In this paper, we seek to test the performance of a specific generative adversarial net, called SeqGAN, on natural language text from Twitter. In this section, we will first introduce Twitter, then generative adversarial networks, and lastly the SeqGAN model.

1.1 Twitter

Twitter is an internationally popular social media platform on which users share short posts called "tweets" (Twitter, Inc., 2020b). Tweets may in general contain photo, video, URLs, and more, but for the purposes of our study, we have stripped them down to contain only text. Tweets differ from the posts one may find on Facebook or other social media platforms as they are held to a 280 character limit, which makes tweets short and concise by design. The result of this is that tweets feature rather informal usage of English, comparable to the styles often used in text messaging. This informality is desirable, we propose, because the lack of strict grammar rules governing punctuation, run-on sentences, etc. means that a network which fails to learn these rules will still likely appear convincing.

The informality of tweeted text, however, is not necessarily universal. For example, many news agencies share breaking news and short snippets from articles on Twitter. These tweets generally use headline-style English, with proper spelling and grammar. It is still typically looser than what one might find on Wikipedia or in a journal paper, though, given that it is social media and is designed to be concise.

With this in mind, we desire to study how this relaxed language structure will impact a GAN's ability to generate convincing tweets. Because different kinds of Twitter accounts will tweet more or less formally, we will train GANs on the tweets from two different kinds of Twitter accounts, news accounts and universities, to determine if there is a relationship between GAN performance and the perceived formality of the training material.

1.2 Generative Adversarial Networks

Generative adversarial networks, or GANs, are fundamentally neural networks which generate data (*generative*) and are trained against themselves (*adversarial*) (Goodfellow and Pouget-Abadie et al., 2014). In particular, a GAN consists of two networks: a **generator** and a **discriminator**. The gen-

erator is trained to generate data, where the discriminator is trained to differentiate between real and generated data.

The trademark of the GAN is the process of **adversarial training**, a training method that pits the generator and discriminator against each other. In this process, the generator is fed random input and asked to generate new data samples. The generated samples are then fed, along with real data samples, into the discriminator, which is asked to determine which are real and which are generated. If the discriminator can successfully identify which is which, then the generator is updated more strongly. If not, the discriminator is updated more strongly. The two networks are thus trained in concert: as the generator generates more sophisticated data samples, the discriminator gets better at telling which is which. Ideally, the generator will eventually "win" this "battle" of the networks, as it gets sophisticated enough that its output cannot be told apart from real data.

In the realm of image generation, this network architecture has led to the creation of highly sophisticated networks capable of fooling even humans. The most popular examples are generated images of human faces: in most cases, humans cannot tell whether an image was a photograph of a real person or was generated by a GAN. The most famous of these implementations is StyleGAN (Karras et al., 2018). One can see examples of these generated faces at `thispersondoesnotexist.com`, a website which uses StyleGAN to generate a new face every time you refresh the page.

Our goal, then, is to use these methods with tweets. Our hope is to identify whether GANs are able to successfully fool humans into believing that GAN-generated tweets were written by real humans, and to measure to what extent the informality of the language in tweets facilitates this.

1.3 SeqGAN

SeqGAN (Yu et al., 2016) provides the framework to solve the two main problems that arise in applying GANs to text generation. The first is that GANs expect the data to be continuous, however text is a form of discrete data (sequence of discrete tokens). The second is that GANs can only calculate the loss for the partially generated sequence as it is being generated, but has no way of knowing what the resulting loss of the entire sequence will be.

To resolve these issues, SeqGAN models the generator network as an agent in reinforcement learning, the previously generated words as the state, and the word to be generated as the action. The reward for the generator comes from the evaluation of the generated sequence by the discriminator. This is then fed back to the generator to guide its learning. By modeling the generative network as a stochastic parametrized policy, the problem of discontious data is resolved by performing policy gradient.

2 Methods

In this section, we outline the methods we used throughout the study. We began by mining tweets from Twitter using the Twitter API. Then, we modified an existing SeqGAN implementation to use our tweet data, and we trained the GAN. A summary of the training parameters used is provided.

2.1 Mining Tweets

To mine tweets from Twitter, one needs to use the Twitter API for developers (Twitter, Inc., 2020a). Thus, we registered our project as an app with Twitter Developers, granting us API access tokens. Then, we used Tweepy, a Python package that provides a framework to use the API in Python.

We decided on the following sets of accounts to mine for our datasets. For news accounts, we used CNN Breaking News (`@cnnbrk`), the New York Times (`@nytimes`), CNN (`@cnn`), BBC Breaking News (`@bbcbreaking`), BBC World News (`@bbcworld`), the Economist (`@theeconomist`), Reuters (`@reuters`), the Wall Street Journal (`@wsj`), the Washington Post (`@washingtonpost`), the Associated Press (`@ap`), NBC's Breaking News (`@breakingnews`), and Fox News (`@foxnews`). For university accounts, we used Princeton University (`@princeton`), Yale University (`@yale`), Harvard University (`@harvard`), Dartmouth College (`@dartmouth`), Columbia University (`@columbia`), Brown University (`@brownuniversity`), the University of Pennsylvania (`@penn`), Cornell University (`@cornell`), Stanford University (`@stanford`), the Massachusetts Institute of Technology (`@mit`), the California Institute of Technology (`@caltech`), and the University of Chicago (`@uchicago`). Notice that each list contains twelve accounts.

For each account, we used Tweepy to pull the most recent 200 tweets. The resulting 2400 tweet-datasets were shuffled and split half-and-half into two sets: a training set and a testing set. In the case of the universities dataset, these were again split half-and-half, due to training time limitations and a desire to see if a smaller dataset improved performance, yielding a 600 tweet training set and a 600 tweet testing set. The most recent data collection took place on January 8, 2020, so our news-related tweets involve much reporting about Iran, the killing of General Soleimani, President Trump, and the wildfires in Australia, among other things.

The tweets then undergo some processing before being passed on to the GAN. We first remove all URLs from the harvested tweets. Then, all characters are replaced with their lowercase equivalents. Finally, the tweets are tokenized using the Natural Language ToolKit (NLTK) tokenizer (Bird et al., 2009). It is in this form that the GAN receives the data and generates new tweets. A downside to using the NLTK tokenizer, we have found, is that Twitter handles and hashtags are broken, i.e. '@princeton' becomes '@' 'princeton', '#PrincetonU' becomes '#' 'PrincetonU', etc., when these should realistically be treated as single words, in the opinion of the authors.

2.2 SeqGAN Implementation

The implementation of SeqGAN that we used is the SeqGAN model available on GitHub with user `williamSYSU`'s TextGAN repository (williamSYSU, 2019). TextGAN is an impressive framework providing PyTorch implementations of several textual GAN models, including SeqGAN, which all use the same centralized methods for accessing datasets, saving generated samples, calculating metrics, etc. In this implementation, the generator is modeled as a recurrent neural network, and the discriminator is modeled as a convolutional neural network. Our work, then, primarily involved modifying TextGAN to read our tokenized Tweets and determining which training parameters to use.

2.3 Training parameters

Training with SeqGAN occurs in two sections: pre-training and adversarial training. First, the generator is pre-trained using standard maximum likelihood estimation methods. The discriminator is then pre-trained using real data in conjunction with the generated samples produced my the generator at the end of pre-training. Then, adversarial training begins.

In pre-training, the relevant parameters are the number of MLE training epochs for the generator, the number of training steps for the discriminator, the number of epochs per training step for the discriminator, and the learning rates for the generator and discriminator. We set the parameters as described in Table 1.

Generator MLE epochs	80
Generator learning rate	0.01
Discriminator training steps	5
Discriminator epochs per step	3
Discriminator learning rate	0.01

Table 1: Parameters used in pre-training. The generator undergoes 80 pre-training epochs and the discriminator undergoes 15 pre-training epochs.

In adversarial training, we have many of the same parameters to play with. We adjust the number of generator training steps per adversarial epoch, the number of discriminator training steps per adversarial epoch, the number of discriminator epochs per training step, and the learning rates for each network. We set the parameters as in Table 2.

Adversarial epochs	150
Generator training steps	1
Generator learning rate	0.0001
Discriminator training steps	5
Discriminator epochs per step	3
Discriminator learning rate	0.01

Table 2: Parameters used in pre-training. Note that the number of training steps listed above is steps *per* adversarial epoch; i.e. for each adversarial epoch, the generator trains once and the discriminator trains fifteen times.

Another decision that has to be made for adversarial training is the loss function. We use a relativistic loss function: the log of the sigmoid of the difference between a real and a fake sample.

The SeqGAN model turns out to be highly sensitive to these parameters. As we will discuss later, this set turned out to be less-than-stellar, but we were unable to find a set of parameters which performed better in our small hyperparameter search.

3 Results and analysis

To evaluate the performance of our models, we tracked our GANs with four metrics: the negative log likelihood of the generated samples, the accuracy of the discriminator, BLEU scores for the generated samples using 2-, 3-, 4-, and 5-grams, and the loss functions for each network. The negative log likelihood and BLEU scores were tracked every ten epochs for the generator over both pre-training and adversarial training. The losses were recorded at every epoch of adversarial training alone. The discriminator accuracy was recorded at every "d-step," five of which make an epoch during adversarial training.

BLEU scores are typically used as a metric for machine translation, but, as originally done in the original SeqGAN paper (Yu et al., 2016), we use them as a measure of the degree of similarity between the generated tweets and the actual tweets, using the entire training set as the reference.

3.1 News-like Tweets

Surveying the news dataset, we found that most of the tweets have a decently recognizable style, like that of newspaper headlines. Further, there are a few recognizable phrases that are repeated often, like "More:" or "Follow live updates:", followed by URLs which were removed in the preprocessing of the tweets. As such, our hope was that the GAN would be able to pick up on these features of the tweets.

After training our model on the news dataset using the parameters in subsection 2.3, we surveyed the samples generated by the GAN over the course of the adversarial training. Below are a few of these news-like tweets:

- the british has attacked a top iranian production for water
- president trump has make a furore over the internet their airport sustained by his intelligence force
- hong kong cancer students want to resolve strikes in new york city. follow for live updates:
- a federal judge voiced skepticism for a misconduct live " @foxnewsnight

A cursory reading of these tweets shows us that the GAN was able to pick up on the headline-esque style of news tweets, and also it successfully uses phrases like "follow for live updates:." We also see that it uses the Twitter handles of other news accounts in the correct ways, by adding them at the end of tweets, sort of like references.

We also have the evalutation metrics described above to evaluate the performance of the GAN on the news dataset. Plots of these metrics over time may be found in Figure 1.

In these plots, we see evidence that the adversarial training made rather little impact on the performance of the GAN. At the start of adversarial training, BLEU scores plateau, the negative log likelihood begins to increase, the discriminator achieves 100% accuracy, and the losses become almost instantaneously zero. It turns out that we see very similar results with the university-like tweets, so we shall defer more in-depth discussion until subsection 3.3.

3.2 University-like Tweets

Looking at our universities dataset, we see that there are fewer general stylistic trends. Universities tend to tweet about a diverse range of things, each with their own style, from announcements about awards given to students or faculty to fun tweets like "Happy Monday!" to tweets about events happening on campus. The result is a very varied style and a wide vocabulary. Our hope was that this would allow the GAN more freedom in making up novel tweets while simultaneously retaining decently structured English; the result seems to be general confusion.

Unlike the news-like tweets, the university-like tweets resemble the universities dataset only somewhat. Here are some examples:

- a harvard conference new uchicago's study by @columbiauenergy. more
- " uchicago continues at new @browncsdept.
- wondered researchers have a skills holds changes in a neighbors while inspiration in mathematics

We see some positive things, such as how the GAN references universities as the subjects of these tweets and how it uses the Twitter handles of university departments as we would expect ("@columbiauenergy", "@browncsdept", etc.). A dead giveaway that these are not real tweets, though, is the egregious mixing of universities; it is somewhat

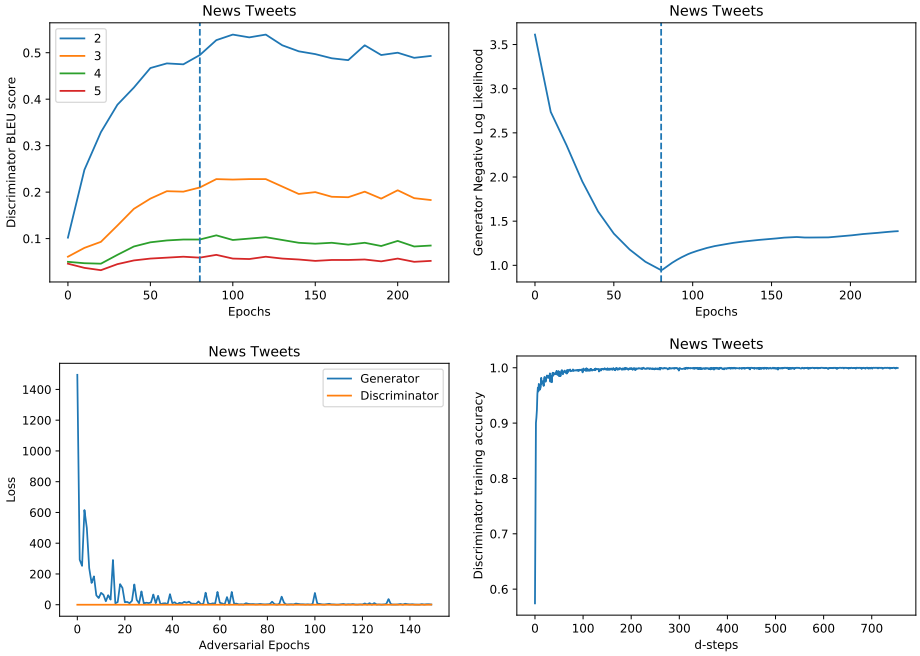

Figure 1: Evaluation metrics over the course of training on the news dataset. The first 80 epochs of (a) and (b) are pre-training; the dashed vertical line denotes the beginning of adversarial training. Losses and discriminator accuracy are only plotted over adversarial training epochs. Five "d-steps" make one adversarial epoch in the accuracy plot.

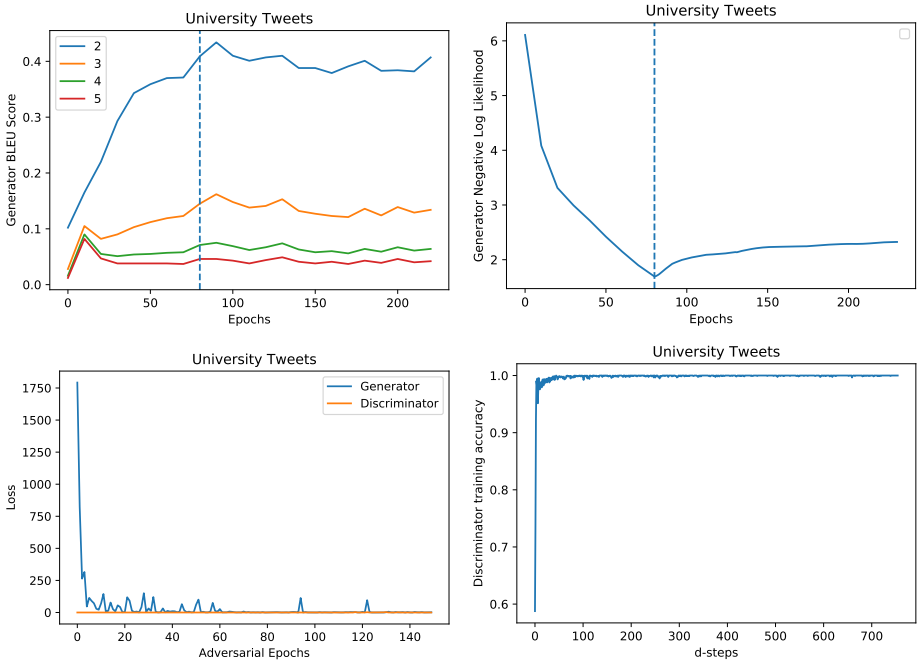

Figure 2: Evaluation metrics over the course of training on the universities dataset. As with Figure 1, the first 80 epochs of (a) and (b) are pre-training; the dashed vertical line denotes the beginning of adversarial training. Losses and discriminator accuracy are only plotted over adversarial training epochs. Five "d-steps" make one adversarial epoch in the accuracy plot.

humorous to think that Harvard would host a conference about a UChicago study that was actually performed by Columbia University. Further, the tweets unfortunately do not make a lot of sense.

The evaluation metrics over the course of training are plotted in Figure 2. The performance of the GAN on the universities dataset looks quite similar to its performance on the news dataset.

3.3 Discussion

The trends we see in the plots of the evalutation metrics show us that progress really halts at the start of adversarial training. The BLEU scores plateau. The generator negative log likelihood begins to increase, when it should decrease. The losses almost immediately drop to 0. The discriminator accuracy immediately goes to 100%. Why?

We believe that this has everything to do with our training parameters. The parameters we use allow the discriminator to very quickly "figure out" the generator, and the generator is from then on stuck in a random well. Each epoch, it performs effectively randomly, and then it receives feedback which is entirely unhelpful. The solution, we believe, lies in performing a wider hyperparameter sweep to find a set of training parameters that performs better.

The solution may also, however, depend on the size of the datasets. For our studies, our training sets consisted of 1200 tweets (news) and 600 tweets (universities). As such, it is entirely possible that this is too small, enabling the discriminator to learn precisely which are coming from the original dataset and calling anything else "wrong."Because the generator is performing basically randomly, this works, and the discriminator can always tell generated tweets apart from real ones. This performance from both the discriminator and the generator is bad, as we want the discriminator to learn which *patterns* make a tweet real, and we want the generator to receive feedback that helps it perform non-randomly. With significantly larger datasets, then, the discriminator cannot simply learn which precisely which tweets are in the training set, and thus the generator may have more of an edge in the adversarial training.

Another question, though, is why do the news-like tweets *seem* better than the university-like tweets when the performance of the GAN is about the same? This, we believe, has to do with the nature of the datasets. News accounts tend to tweet several times about related things; in the days leading up to January 8th, for instance, Iran and President Trump were pretty regular topics. Further, when an event occurs, each of the twelve accounts likely tweets about it. This means that the news dataset has a lot of repetition of given keywords, giving it a better shot at understanding how terms like "Iran" or "strike" might be used.

The universities dataset, on the other hand, does *not* lend itself to the repetition of terms. Universities rarely tweet multiple times about the same subject, and the tweets of one university are usually about events related to that specific university. This means that a given account does not often repeat its own terms—outside of, for example, the university name and its departments—and there is little repetition across accounts, as, for example, Princeton tweets about Princeton and UChicago tweets about UChicago. The result is that the vocabulary is much more varied and leaves the GAN with fewer internal structures to learn, explaining why the university-like tweets were seemingly worse.

4 Conclusions and Future Work

To conclude, we have found that, with our existing implementation and training parameters, the GAN does not perform much better than a simple MLE-trained generative model. The result, we expect, has to do largely with the parameters used, as SeqGAN is highly sensitive to its input parameters. It may also be dependent on the size of the dataset; perhaps larger datasets would improve performance.

As such, there are many possible directions that could be taken for future work. Firstly, one could conduct a thorough hyperparameter search to see if a different set of training parameters would improve the model's performance. We attempted to explore different parameters, but were as yet unable to find a set which performed better.

It would also be interesting to experiment with different datasets. We think the GAN could potentially generate very realistic tweets with a more focused dataset about one subject, where there would be many instances of the same words; for example, tweets from news outlets only about the Hong Kong protests. This would improve the GAN performance by eliminating the lack of repetition that we believe was responsible for the failures on the universities dataset. As mentioned previously, significantly larger datasets may also be helpful. The issue with these, however, is that SeqGAN is based

on reinforcement learning, which performs very slowly on large datasets.

This opens up the possibility of using different GAN implementations that do not use reinforcement learning. Possible GAN models to try include an autoencoder approach (Donahue and Rumshisky, 2018), a Gumbel-softmax approximation of the discrete text distribution (Kusner and Hernndez-Lobato, 2016), or the Wasserstein GAN (Arjovsky et al., 2017).

Contributions

The division of labor was as follows.
Connor:

- wrote Tweet mining script
- modified and ran TextGAN

Kathryn:

- wrote Tweet processing script
- analyzed resulting metrics and plotted results

Both parties worked on poster and paper.

Acknowledgments

We would like to thank Professor Karthik Narasimhan for his advising on this project, as well as the rest of the COS 484 staff for their aid over the course of the semester.

References

Martin Arjovsky, Soumith Chintala, and Lon Bottou. 2017. Wasserstein GAN.

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

David Donahue and Anna Rumshisky. 2018. Adversarial text generation without reinforcement learning. *CoRR*, abs/1810.06640.

Ian Goodfellow and Jean Pouget-Abadie et al. 2014. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

Tero Karras, Samuli Laine, and Timo Aila. 2018. A style-based generator architecture for generative adversarial networks.

Matt J. Kusner and Jos Miguel Hernndez-Lobato. 2016. GANs for sequences of discrete elements with the Gumbel-softmax distribution.

Twitter, Inc. 2020a. Twitter Developer.

Twitter, Inc. 2020b. Twitter.com.

williamSYSU. 2019. TextGAN-PyTorch.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2016. SeqGAN: Sequence generative adversarial nets with policy gradient. *CoRR*, abs/1609.05473.