LaTextGAN for generating novel tweets in the style of Donald J. Trump

Gerrit Bartels and Jacob Dudek

Cognitive Science, Universität Osnabrück

Implementing Artificial Neural Networks with Tensorflow

Prof. Dr. Michael Franke

April 04, 2021

No one knows more about deep learning than I do! ~ **Jonald T. Drump**

Abstract

As part of the project for the "Implementing Artificial Neural Networks with Tensorflow" course at the University of Osnabrück, we adopted the approach from Donahue and Rumshisky (2019) and created a LaTextGAN to generate novel tweets in the style of the former US President Donald J. Trump. The LaTextGAN architecture the authors proposed in their pape "*Adversarial Text Generation Without Reinforcement Learning*" is a combination of an Autoencoder and a Generative Adversarial Network. It was originally used to generate novel English sentences based on the Toronto Book Corpus. In order to imitate Trump's tweeting style and behaviour we created a corpus[1] containing all his tweets, starting from the creation of his account in May 2009 up until his recent ban from twitter in January 2021. Furthermore, we evaluated different data preprocessing methods and several variations of the standard LaTextGAN to find the most promising architecture for the task at hand.

1. Introduction

Today, when trying to generate text, one can choose from a vast variety of generative models, one of which is the variational autoencoder (VAE) (Kingma and Welling, 2013). The basic idea of autoencoders (AE) is to generate a compressed representation of the data, which still includes most of the important information - this representation is called the embedding. AEs consist of an encoder - decoder structure, however, in their naive approach they cannot be used as generative models. The reason for this is that the encoder does not learn dense representations around the input vectors. VAEs overcome this problem by learning dense representations, allowing the decoder to work with random input vectors and generate artificial samples.

Generative Adversarial Networks (GANs) on the other hand are designed towards generating artificial data by utilising a simultaneously trained generator and discriminator model (Goodfellow et al., 2014). In a counterplaying fashion, the generator attempts to create convincing samples that the discriminator misclassifies as true samples. Through this process the generator learns to create artificial samples that capture the true underlying distribution. Essentially, this can be seen as a two-player minimax game, ideally played until an equilibrium is reached. To circumvent problems occurring in GANs, i.e. instability of learning, mode collapse and vanishing gradients, the Wasserstein GAN (WGAN) was developed (Arjovsky et al., 2017). In WGANs the discriminator is exchanged with an approximation to the Wasserstein-metric[2]. Due to disjoint supporting manifolds when switching the underlying similarity measure, vanishing gradients are effectively tackled. Since approximating the Wasserstein metric requires a Lipschitz continuous function, Arjovsky proposed to clip the weights of the discriminator to a fixed range. However, this can lead to unwanted side effects as shown by Gulrajani et al. (2017). In their improved version they introduce an alternative, by directly penalizing the norm of the discriminator's gradients to be at most 1 with respect to its input.

The class of GANs has shown great success in image generation, with application in fields like data augmentation, style based generation and image upsampling. However, certain problems arise when trying to generate text with GANs. The main problem here is that text is discrete. Usually, when training a GAN the gradients are passed on from the discriminator to the generator. However, while using the generator as a Language model its current prediction involves greedily picking[3] the next word. This selection is non-differentiable, making it impossible to apply the

[1] Our corpus is based on a dataset from the Trump Twitter Archive V2

[2] also known as Earth-Mover's Distance

[3] e.g. taking the argmax over a softmax distribution

backpropagation algorithm without further adjustments.

Several solutions have been proposed to circumvent this problem. One of them is the reinforcement learning approach from Yu et al., (2017), that uses so-called policy gradients to find the optimal policy for generating text. In their variant the generator represents aforementioned policy where state s is defined as the already generated text and the action a as the selection of the next word for the sentence. The role of the discriminator is to yield a reward for each generated sentence, where high rewards correspond to the discriminator misclassifying the sentence as real. These rewards are then used to improve the generator's policy via the REINFORCE algorithm (Sutton and Barto, 2018). To account for intermediate rewards Yu et al. use a special method called Monte-Carlo rollouts. Problems in the reinforcement learning approach are the high variance in the estimation for the gradient due to the limited amount of samples and the large state-action space. This leads to an unstable training process, slow convergence and leaves parts of the state-action space unexplored. Altogether, those points might entail a poor sentence quality.

Apart from that, there are also other non-RL approaches dealing with the problem of discrete text. For example, Kusner and Hernández-Lobato (2016) have shown that greedily picking from a softmax distribution can be approximated efficiently by sampling from a so-called Gumbel-softmax distribution. This results in a continuous and differentiable function.

Alternatively, Donahue and Rumshisky (2019) developed a latent-space GAN for text called LaTextGAN that uses a combination of an AE and an improved WGAN to tackle the problem. For generating novel tweets in the style of Donald J. Trump, we decided to reimplement this architecture.

1.1 The LaTextGAN

The idea behind the LaTextGAN is to generate new sentences from a continuous low-dimensional space instead of working with discrete text directly. Donahue and Rumshisky achieved this by first training an AE on a given dataset such that the encoder is able to produce low-dimensional sentence embeddings from it. The task of the generator is now to create novel samples from this learned latent space that can be decoded back into sentences by the AEs decoder. During training the discriminator has to correctly classify sentence embeddings coming from both the encoder (real samples) and the generator (fake samples). Figure 1 shows the schematic of the original architecture. Since the AE is dealing with sequential data LSTM networks are used to read in (encoder) and reconstruct (decoder) the sentences. LSTMs were chosen due to their innate capability to deal with longer sequences more effectively. This property can be attributed to the usage of an additional cell state serving as the long term memory. Additionally, the vanishing gradient problem of vanilla RNNs is mitigated by the usage of the input, output and forget gates (Hochreiter and Schmidhuber, 1997). Furthermore, the LaTextGAN authors decided to implement the GAN as the improved Wasserstein variant we described above. To ease the vanishing gradient problem for deeper generator and discriminator networks a ResNet architecture was used.

2. Dataset & Preprocessing

To build our training corpus we used Trump's complete Twitter history from the Trump Twitter Archive. As our goal was to generate novel tweets in the style of Trump we excluded all retweets, keeping only those tweets that he wrote himself. To improve the quality of our data (~47k tweets) we applied several preprocessing techniques. This included removing non-English tweets and most special characters, except those essential to sentence structure and twitter itself e.g. "@", "#". Moreover, we removed links and placeholders for images.

Before applying nltk's TwitterTokenizer we introduced "<NUM>" as a unique token for numbers in order to reduce the overall amount of tokens in our data. Since we train a model for

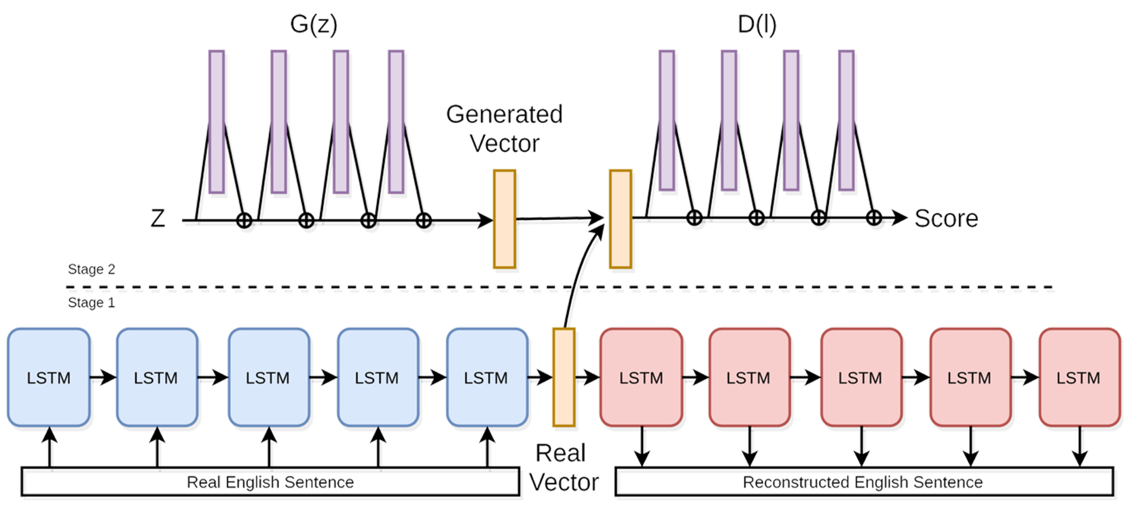

Figure 1: Donahue‘s and Rumshisky‘s schematic of the original LaTextGAN architecture.

natural language generation (NLG) we included unique start of sequence “<Start>” and end of sequence “<End>” tokens to every tweet. Lastly, we removed all tokens with a frequency below two as well as the tweets they appear in. This left us with around 29k tweets. Setting the frequency threshold higher than that would have resulted in a too small dataset. Nonetheless, we also trained a variant containing all tokens (~45k tweets) to rule out that removing all associated tweets results in a far worse performance. All models received padded batches as input, allowing us to train them on different sequence lengths. To prevent any influence the padded values (zeroes) might entail, we applied zero-masking in all layers.

Contrary to the LaTextGAN authors we decided to pre-obtain the 200-dimensional word embeddings by using a skip-gram model rather than training them directly on the task. This decision has again been made due to the limited amount of training data.

3. Models & Hyperparameters

In the following we will go over our different model implementations and parameter choices. As a baseline we implemented a vanilla LaTextGAN similar to the approach from Donahue and Rumshisky (2019). Building upon that, we designed three variations, each increasing in complexity.

To integrate the pre-trained word embeddings from our skip-gram model we included non-trainable embedding layers to each AE. Following the authors, we set the cell size of the LSTMs to 100 for the encoders and to 600 for the decoders. We chose to leave out recurrent dropout for the encoder LSTMs because it significantly slowed down training[4] without benefitting the model's performance. To ensure a more stable and faster training process of the AEs we decided to make use of teacher forcing (Jurafsky & Martin, 2009). However, when later using the trained decoder to reconstruct sentences from the generators output we have to switch from teacher forcing to an inference mode. This allows the decoder to use its previous prediction as input for the next time step. Initially, we provide the start of sequence token and the generator's latent representation. Each step then consists of projecting the LSTM output onto vocabulary size and applying the softmax operation. From the resulting probability distribution the next word in the sequence must be picked, for which we used greedy sampling. The decoding procedure itself stops when an end of sequence token has been generated or the decoder has generated the maximum sequence length of 78 tokens[5].

[4] Recurrent Dropout prevents tensorflow from using the highly optimized CuDNN LSTMs

[5] length of longest tweet in corpus after preprocessing has been applied

Regarding the WGAN generator and discriminator we adopted the idea to implement them as ResNets with 40 Residual Blocks each. Every block is then of the form:

$$F(x) = H(x) + x \text{ and } H(x) \text{ is defined as}$$
$$H(x) = relu(x \cdot W_1 + b_1) \cdot W_2 + b_2$$

and each layer in a single block consists of 100 units. These skip-connections allow us to train a fairly deep WGAN which, as shown by Gulrajani et al. (2017), is beneficial to the network's performance when being combined with their improved WGAN loss. Like the authors, we enforced the unit gradient norm only along straight lines that are uniformly sampled between pairs of points from the data and generator distribution. The new loss objective is then:

$$L = \underset{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g}{\mathbb{E}} [D(\tilde{\boldsymbol{x}})] - \underset{\boldsymbol{x} \sim \mathbb{P}_r}{\mathbb{E}} [D(\boldsymbol{x})] + \lambda \underset{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}}{\mathbb{E}} \left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2 \right]$$

where P_g is the generator-, P_r the data- and P_x the new sampling distribution described above. Additionally, we set the penalty weighting (λ) to the recommended value of 10. To further stabilize the training of the WGAN, we implemented its training procedure such that the generator is updated once for every ten updates to the discriminator (Donahue and Rumshisky, 2019).

To achieve faster convergence for all models we also selected the Adam optimizer (Kigma and Ba, 2014) and set the learning rate to $\alpha = 5 \cdot 10^{-4}$ for the AEs and to $\alpha = 1 \cdot 10^{-4}$ for the WGANs. With said hyperparameters we trained the AE for 50 epochs, to subsequently train the WGAN over a period of 150 epochs. Since we deal with a relatively large vocabulary (~16k tokens) one-hot encoding our AE targets was computationally infeasible for our hardware. Therefore, Sparse Categorical Cross Entropy (SCCE) is a good alternative loss function to CCE when dealing with categorical data but having integer targets.

As a first variation we introduced a stacked variant of the standard LaTextGAN by adding an additional LSTM layer to both encoder and decoder. The idea behind that is to process the input sequence on different levels of abstraction in order to extract a more sophisticated sentence representation. Since we use a stacked encoder that now returns two embeddings (last hidden state of each LSTM layer) the generator has to be adapted to generate new sentences in this learned latent space. We chose to do so by adding an additional 40 blocks deep ResNet to the generator that receives its own random input and can operate on a different abstraction level. Variants two and three additionally introduce bidirectionality to the baseline and stacked LaTextGAN architectures. By being able to process the input sequences once going with and once going against time, the encoder should be able to produce richer contextual representations. As final encoder output we decided to concatenate the hidden states of the forward and backward LSTMs. In theory this should help to train a better LaTextGAN that is able to generate more advanced novel tweets.

4. Evaluation

4.1 Latent Space Analysis

As the goal is to generate new tweet representations from the underlying encoder distribution, it is crucial to examine how well the latent space of the generator resembles the one of the encoder. To make them comparable we project the high-dimensional tweet embeddings, both from encoder and generator, into a 2-dimensional space using Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE)[6].

[6] First we reduced the number of dimensions to 50 using PCA and then applied t-SNE as it is recommended by its developers when dealing with high dimensional data.

4.2 BLEU-4 Scores

In order to compare our models to each other and to the results of Donahue and Rumshisky (2019), we also utilized BLEU-4 scores as an automatic evaluation metric. The scores were calculated for uni-, bi-, tri- and four-grams with uniform weights (0.25, 0.25, 0.25, 0.25). We carried this out with 500 generated tweets from each model and set all tweets as reference data.

4.3 Analysis of results

Generally, all AEs exhibited a monotonically decreasing loss for both train and test data that started to saturate around the 20th epoch. For the WGAN training, the loss looks less well behaved. Even though discriminator and generator loss approach each other, they never fully converge. However, we found that training them for 150 epochs yielded the best results. This training time is significantly longer compared to the original authors, which we suspect is due to their larger amount of training data.

When looking at Figures 2 and 3, one can see that every generator was fairly able to capture the underlying encoder distribution. One noticeable difference in the stacked variations are the two distinct clusters in the encoder's latent space. This is attributable to the two hidden states the encoder returns in those variants.
Upon closer inspection of these t-SNE plots (Figure 3) we found that several point clouds of the generators embedding differ from the real underlying distribution. While both generator embeddings exhibit two clusters, they do not completely share the same boundary as the underlying encoder embeddings. This is more pronounced in the non-bidirectional variant. A reason for the generally worse overlap in the stacked architectures might be an increased difficulty when trying to capture such twofold embeddings.

These findings also coincide with the BLEU-4 scores we obtained from all models (Figure 4). Except for the case of the Bidirectional Stacked LaTextGAN the scores decrease with increasing model complexity where the largest difference occurs when switching from the standard to the stacked LaTextGAN variation. Moreover, we obtained a higher BLEU score for the standard variant when removing rare words compared to the standard variant trained on all words. With that we have shown that it was justified to work with the smaller but cleaner dataset. When comparing our scores to the one of the LaTextGAN from the original paper, we can see that our standard LaTextGAN achieved an almost equal score. However, one should keep in mind that we have worked with very different datasets, making an exact comparison difficult.

Lastly, we will evaluate how well our models were able to imitate Trump's tweeting style. For that we will look at the quality of each model's generated tweets (Appendix). What is directly noticeable is that every model has problems creating grammatically and syntactically correct sentences. This may be primarily due to working with Twitter Data in the first place, but the low availability of overall data certainly contributes to the problem.

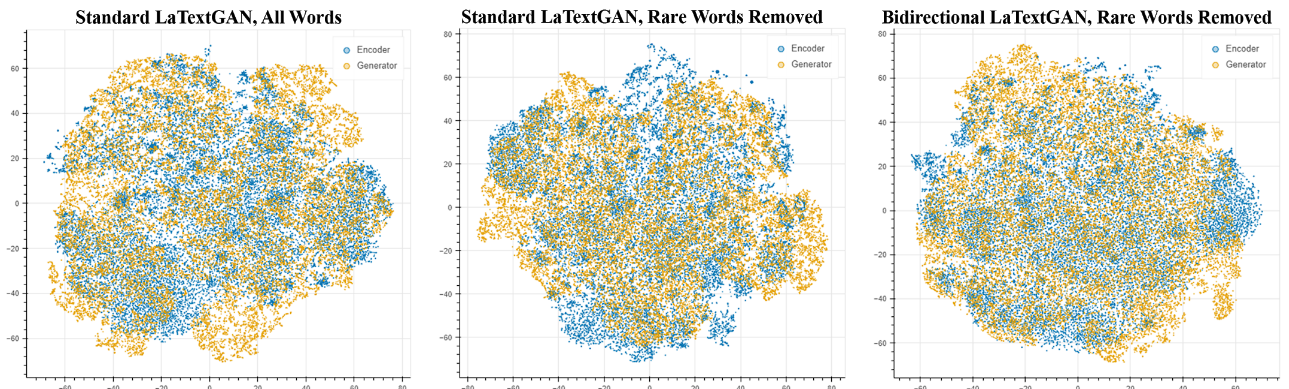

Figure 2: Latent Space Embeddings from all non-stacked variants. Both Encoder (blue) and Generator (orange) provided 500 sequence vectors. First PCA and then t-SNE were applied to project the higher dimensional vectors to 2 dimensions.

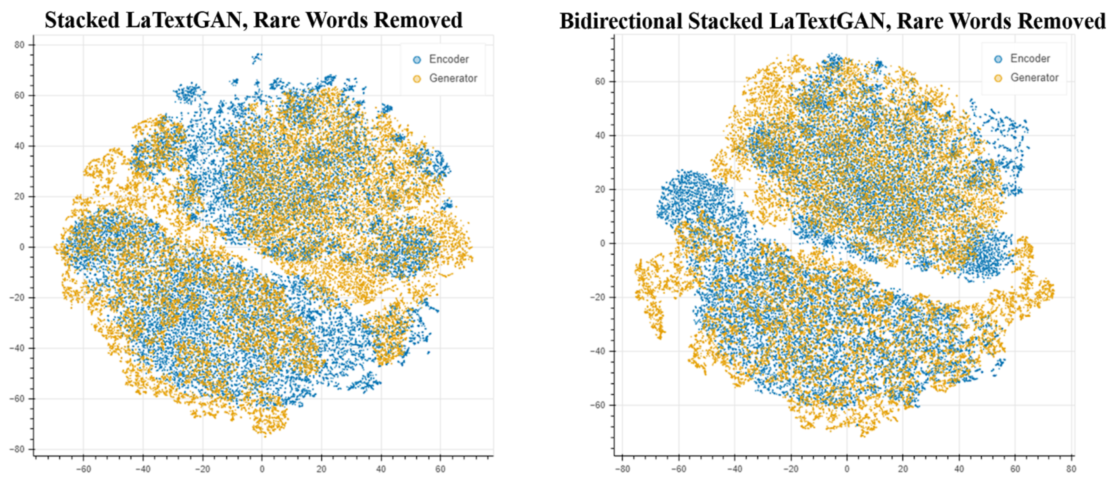

Figure 3: Latent Space Embeddings from all stacked variants. Both Encoder (blue) and Generator (orange) provided 500 sequence vectors. First PCA and then t-SNE were applied to project the higher dimensional vectors to 2 dimensions.

In terms of sequence length, each model produces a variety of different tweets with an average token count fairly similar to our training data (Figure 5). Especially longer sequences often fail to keep the context consistent within the same tweet. This seems to be more pronounced in those models that also received a lower BLEU score. Shorter tweets that consist of only a few words have the best quality. Nonetheless, there are also longer sequences that are mostly syntactically correct and carry meaning. Overall, in all models there are few generated tweets that are exceptionally good with many of them exhibiting the problems described above. That being said, we think features of Trump are definitely present in the generated tweets, especially in terms of his characteristic linguistic style. However, we believe that results still leave room for improvements upon which we will touch in the next section.

LaTextGAN	Bleu-4 Score
Standard, All Words	0.6554
Standard, RWR	0.6739
Bidirectional, RWR	0.5684
Stacked, RWR	0.4391
Bidirectional Stacked, RWR	0.5235

Figure 4: Bleu-4 scores for all LaTextGAN variants (RWR = Rare Words Removed).

Source	Avg. Tokens
Training Data	25.50
Standard, All Words	27.39
Standard, RWR	26.94
Bidirectional, RWR	17.83
Stacked, RWR	23.60
Bidirectional Stacked, RWR	23.85

Figure 5: Average tokens per tweet. For the LaTextGAN variants the score was calculated over 1000 generated tweets (RWR = Rare Words Removed).

5. Discussion

After having evaluated our different architectures, there are still open questions and suggestions from our side for further improvements. One important point of criticism is our comparatively low amount of training data. We thought about enriching our dataset with public speeches held by Trump. However, this would have conflicted with our idea to generate text in the form of classical tweets. Nevertheless, training with a larger dataset might have resulted in a more pronounced difference in the performance of our proposed architectures. Unfortunately, our generated tweets often lack long term dependencies of context - especially for longer tweets. One explanation might be choosing to generate whole tweets instead of single sentences[7] which certainly adds to the problem of low available training data.

Another point worth mentioning is how we evaluated the performance of our models. Automatic evaluation methods such as BLEU or ROUGE are commonly reported and widely accepted in the field of NLG. However, as they only measure direct word-by-word similarity, not considering the actual meaning, they often fail to evaluate generated sentences at a higher level. That is why human evaluation is necessary to properly evaluate the quality of generated text. Preferably, we would also have had humans judge the goodness of each models' generated tweets to obtain a more sophisticated comparability. But unfortunately, we had to omit this methodology due to limited time and resources. Measuring how well the generator actually captures the underlying distribution also remains a difficult task, as no proper evaluation metric exists. Therefore, usually judging the overlap of t-SNE embeddings is a common but not perfectly accurate practise.

Looking at our results and the corresponding architectures we found several aspects that might be interesting to investigate in further research. We would recommend implementing a more sophisticated AE that is able to close the gap between train and test loss. This is particularly important as the decoder has to correctly reconstruct sentences from the unseen samples produced by the generator. One idea would be to use a variational Autoencoder instead. Additionally, we would suggest to stop using teacher forcing when training stabilizes. as it was shown that using it permanently might result in worse sample quality (Fedus et al., 2018). Another suggestion from us is to replace the greedy picking operation during sentence reconstruction with Beam Search Decoding. With this we would increase the probability of obtaining higher quality sentences by keeping track of k[8] partial reconstructions.

[7] The original LaTextGAN authors chose a more reasonable max sequence length of 20 tokens.
[8]Hyperparameter that denotes the Beam size

But apart from altering our architectures it might be interesting to look at other approaches for stylized text generation. Over the last years the so-called transformers have shown great success in many deep learning areas. Among these is also the field of NLP with famous examples like GPT-3 or BERT. Therefore, applying them to our task of generating tweets in the style of Donald J. Trump might pose an interesting alternative. For example Zeng et al. (2020) proposed a novel generative adversarial architecture that uses transformers to produce stylized text extracted from a given reference.

6. Conclusion

To conclude, the LaTextGAN poses an interesting architecture that combines cutting edge concepts to utilize GANs for generating text. We have developed different addons to the original approach and evaluated their performance. From our findings we conclude that the vanilla LaTextGAN seems to handle our task the best, at least when working with limited amounts of data. However, more sophisticated approaches, like the bidirectional-stacked autoencoder, still present interesting features that in theory should be able to further increase the quality of generated text. Even though we were able to generate some decent tweets, many of them still lack correct syntax and semantic meaning. Nevertheless, our models adapt the style of Donald J. Trump, in terms of his characteristic writing style on twitter, fairly well. Lastly, we have proposed several ideas to improve the existing framework providing many opportunities for future research.

References

Donahue, David, and Anna Rumshisky. "Adversarial text generation without reinforcement learning." *arXiv preprint arXiv:1810.06640 (2019).*

Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." *arXiv preprint arXiv:1312.6114 (2013).*

Goodfellow, Ian J., et al. "Generative adversarial networks." *arXiv preprint arXiv:1406.2661 (2014).*

Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks." International conference on machine learning. PMLR, 2017.

Gulrajani, Ishaan, et al. "Improved training of wasserstein gans." *arXiv preprint arXiv:1704.00028 (2017).*

Yu, Lantao, et al. "Seqgan: Sequence generative adversarial nets with policy gradient." *Proceedings of the AAAI conference on artificial intelligence*. Vol. 31. No. 1. 2017.

Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. 2nd Edition. MIT press, 2018, p.326-331.

Kusner, Matt J., and José Miguel Hernández-Lobato. "Gans for sequences of discrete elements with the gumbel-softmax distribution." *arXiv preprint arXiv:1611.04051 (2016).*

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

Jurafsky, Daniel, and James H. Martin. "Speech and Language Processing: An introduction to speech recognition, computational linguistics and natural language processing." Upper Saddle River, NJ: Prentice Hall (2009), p. 220.

Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980 (2014).*

Fedus, William, Ian Goodfellow, and Andrew M. Dai. "Maskgan: better text generation via filling in the_." *arXiv preprint arXiv:1801.07736 (2018).*

Zeng, Kuo-Hao, Mohammad Shoeybi, and Ming-Yu Liu. "Style example-guided text generation using generative adversarial transformers." *arXiv preprint arXiv:2003.00674 (2020).*

Appendix

Collection of Tweets from all Models

Standard LaTextGAN, All Words:

- my prayers are with you that your father is a friend of someone who can make things personally , he can make a deal . i want to meet them <End>
- just left <num> points of the united states for president trump , including myself , i have met with the greatest hostage negotiator in history , and i have watched on the u . s . a . g . sulzberger , and i told him that i wanted to speak out of the press . i have no choice , no one else ! <End>
- will be interviewed by @seanhannity . make america great again ! <End>

Standard LaTextGAN, Rare Words Removed:

- there is no surprise medical billing , including the horrible crimes and illegal aliens last week , and never even heard of this bill . this is a total disgrace . we must put on a very good future ! <End>
- despite all horrible statements on me and clear that my administration has done a great job , but the media refuses to talk about them in the last <num> years . they have been saying that they couldnt get elected , and now the republican party is a joke ! <End>
- pervert alert is part <num> . <num> million people watched . i guess he is a loser <End>

Bidirectional Standard LaTextGAN, Rare Words Removed:

- @tigerwoods has never seen the credit of my statements about me and the united kingdom . i look forward being so stupid ! <End>
- remember every member of president obama say that i took office in the first step being made . pure fiction . <End>
- ill be playing with president putin and , tiger tomorrow . real deal thanks <End>

Stacked LaTextGAN, Rare Words Removed:

- get rid of autism . @mittromney and rubio could have a real step towards in action . if i said . christian <End>
- big announcement for david sanders on @foxandfriends the nation that has been treated by the people who have done by the trump national anthem . if no parent , or cut your taxes , and greatness of lives . he has no talent , it is booming ! <End>
- now that failed leadership is involved in syria , something else , including the horrible language , perhaps one thing , perhaps <num> years ago , who would be used to the justice department rule of becoming president ! <End>

Bidirectional Stacked LaTextGAN, Rare Words Removed:

- with everyone had a few republican house members , after <num> years of the great economic success . we must end up your taxes , if biden wins , china , or the democrats produced your job . the american dream will never be allowed to happen again , the american dream is <End>
- its a very interesting president who will be criticized again , south korea is talking about the tea party ? pathetic ! <End>
- some very important first lady will be done by everyone . in any event ! <End>