Translating Tweets from Trumpese to Sanderese with Transformers and CycleGANs

Rogério A. Guimarães Jr. [1]

Abstract

Transformers can achieve state of the art performance in most of the classic language modeling tasks, and CycleGANs have be proving to be be very effective in mapping elements from a domain to another without paired training data. Therefore, we suggest that we can use transformers to provide a representation of language that can be used by CycleGANs to perform machine translation tasks without paired training data too. We propose to use this idea to approach a very simple machine translation task that still involves natural language: idiolect-to-idiolect translation in the short texts in tweets.

1. Introduction

People write, speak, and therefore tweet, in uniquely personal ways. Thus, it should be no surprise that antagonist personalities like Donald Trump and Bernie Sanders have very distinguishable twitter accounts, and most people would find it very easy identifying a tweet from one politician over the other. Therefore, we could ask ourselves what would Trump's tweets look like if they were written by Sanders and vice-versa.

The distinctive and unique way each person uses language is called idiolect, and thus the problem can be framed is a machine translation task from the idiolect of person B to the idiolect of person A. This can be defined as a sequence-to-sequence task, with the input being a sentence in the idiolect of person B, and the output being that sentence translated to the idiolect of person A.

Transformers (Vaswani et al., 2017) have been successfully used in language modeling tasks, and thus in machine translation too. However, to train a transformer one usually needs a large dataset of inputs and outputs examples of the sequence-to-sequence task. For some tasks, such as idiolect translation from *Trumpese* to *Sanderese*, it would be very hard or impossible to assemble such datasets. However, we do have lots of examples from both idiolects publicly available in the twitter accounts of Donald Trump and Bernie Sanders. Therefore, we could, in principle, use Cycle-Consistent Adversarial Neural Networks (CycleGANs) to train a model that is able to translate from one idiolect to the other.

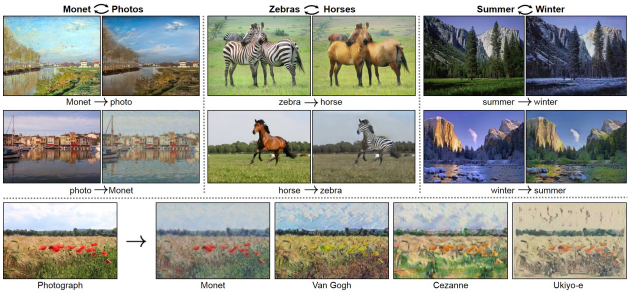

Figure 1. Given any two unordered image collections X and Y, CycleGANs learns to automatically "translate" an image from one into the other and vice versa: (left) Monet paintings and landscape photos from Flickr; (center) zebras and horses from ImageNet; (right) summer and winter Yosemite photos from Flickr. Example application (bottom): using a collection of paintings of famous artists, our method learns to render natural photographs into the respective styles. Figure and caption from (Zhu et al., 2017).

CycleGANs were first proposed by (Zhu et al., 2017) as a way to approach image-to-image translation tasks in which there are no paired datasets of input and output images. Examples of such translations can be seen in Figure 1. We used their model to impose the following constraints on our translator:

1. Translated tweets from Trumpese to Sanderese should not be distinguishable from original Sanders' tweets, and vice-versa.

2. A Trump tweet translated from Trumpese to Sanderese and then back to Trumpese should be similar to the original tweet, and vice-versa.

[1]Department of Electric Engineering and Computer Science (EECS), Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. Correspondence to: Rogério A. Guimarães Jr. <rjunior@mit.edu>.

3. A Sanders tweet translated from Trumpese to Sanderese should be similar to the original tweet and vice versa.

In their initial publication, CycleGANs were used only in image-to-image tasks. New works extended their applicability to music convertion between two genres (Brunner et al., 2018), and voice conversion between two people (Kaneko et al., 2019). However, there is little to no work on their applicability to machine translation. This is a very hard task because of its discrete and recursive nature. We chose idiolect translation because it is the most basic natural language translation task that can be proposed, since two people with distinctive idiolects of the same language still share most of their vocabulary and grammar. In the specific case of using CycleGANs, this extreme similarity and constraint 3, previously mentioned, that the model enforces make a huge pressure for the generator to "paraphrase" the original sentence and thus build a coherent language model, which is not the case for translation between two distinctive languages.

We will show how we adapted the CycleGAN model to the sequence-to-sequence task using transformers, show the results of training this model in data from Trump and Sanders tweets, and discuss the difficulties in training that stopped us from achieving satisfactory results in practice.

2. Implementation

The model was implemented in PyTorch and is publicly available on https://github.com/rogerioagjr/twitterCycleGAN[1].

2.1. Datasets

We created a dataset with the last 1960 tweets, excluding retweets, from both @realDonaldTrump and @BernieSanders. The data were obtained using the twitter developer API. It was then tokenized and padded to form sequences of 50 tokens. The final vocabulary included 8,531 tokens which comprised English words and special characters. URLs were excluded from the tweets.

2.2. Generator Architecture

The generators are the models that translate a sentence from one idiolect to another. Let X be the domain of Trump tweets and Y be the domain of Sanders tweets. We constructed two generators $G : X \to Y$ and $F : Y \to X$. Each generator was composed of a encoder and a decoder.

2.2.1. Encoder

The encoder had several layers. First, it had an embedding layer to convert a batch of sequences of word tokens (size $N \times L$, in which N is the batch size and L is the maximum length of a tweet) into a sequence of embedded representations of each word (size $N \times L \times E_G$, in which E_G is the embedding size). Then, a positional embedding layer similar to the original transformer paper added information about the position of each word to their embeddings. This batch of embedded sentences then went through M layers of transformer multi-headed transformer encoders, each with H heads and a feed forward fully connected head layer of hidden size F_G. The output of the encoder was a batch of representations of the input sentences of size $N \times L \times E$, since the embedding size remains constant in the transformer encoder. It is very important to notice that, in our model, both generators shared the same vocabulary and the same weights in their embedding layer.

2.2.2. Decoder

The decoder was extremely similar to the encoder. It used the embedded representation of the input sentence from the last layer of the encoder (memory) and the current state of the translation of the tweet (input) to predict the next token in the translation. The input was converted into an embedded representation using a embedding layer, augmented with positional information through the positional encoder, and then went through M layers of multi-headed transformer decoders (H heads and hidden size F). The last layer was a linear layer so that the output corresponded to the weight that would be given to each token in the vocabulary for being the next token in the translation. The token with maximum weight was then predicted.

Every sentence begins with the <sos> token, and a decoder is used to predict the next token until the maximum length is reached or a <eos> token is predicted. Sentences are padded so that the output batch of a decoder has the same dimensions $(N \times L)$ of the input batch that was given to the encoder initially.

2.3. Discriminator Architecture

The discriminators try to distinguish the real tweets of a user from the fake tweets that were translations produced by the generators. Given a sequence of tokens, they try to predict the probability that it represents a real tweet from the distribution they are modeling. We constructed two discriminators $D_X : X \to (0, 1)$ and $D_Y : Y \to (0, 1)$.

The discriminator was a simplified version of the encoder.

[1]The implementation of a transformer for language modeling in https://pytorch.org/tutorials/beginner/transformer_tutorial.html and the implementation of image-to-image CycleGANs in https://github.com/aitorzip/PyTorch-CycleGAN were consulted for reference

It was composed of an embedded layer (of embedded size $E_D << E_G$), a positional encoder layer, a single-headed transformer encoder layer with a fully connected final layer of hidden size $F_D << F_G$, and a linear layer with a a sigmoid activation function to map every sentence in the input batch to a single real number between 0 and 1, the probability that the sentence is real.

The discriminator needs to be much more simple than the generator because its task is much easier. A too complex discriminant would be too hard for a generator to defeat, and thus there would be no learning.

2.4. Loss Functions

There are three loss functions that are used in the training, each of them to enforce one of the constraints mentioned in the introduction. They were all modeled as in the original CycleGAN paper. The major innovation presented in this subsection is the use of a embedding function ϕ in the cycle and identity loss functions.

2.4.1. Adversarial Loss

The adversarial loss enforces the first constraint that translated fake tweets should be indistinguishable from real tweets. Its objective is described in Eq. (1).

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] \\ + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x)))]\end{aligned} \tag{1}$$

The objective balances the accuracy of the discriminator in the real and fake tweets. Therefore G is trying to minimize, creating fake tweets $G(x)$ as similar as possible to the real tweets y, while D_Y is trying to maximize it by improving its accuracy.

We also need to consider the analogous adversarial loss on the reverse translation $\mathcal{L}_{\text{GAN}}(F, D_X, Y, X)$.

2.4.2. Cycle Loss

The cycle loss enforces the second constraint that $F(G(x)) \approx x$ (forward consistency) and that $G(F(y)) \approx y$ (backward consistency). Its objective is described in Eq. (2).

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|\phi(F(G(x))) - \phi(x)\|_1] \\ + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|\phi(G(F(y))) - \phi(y)\|_1]\end{aligned} \tag{2}$$

Both generators G and F try to minimize this objective. We need to use a function ϕ that maps sentences to embedded representations of them so that the L1 norm can be meaningful.

2.4.3. Identity Loss

The identity loss enforces the second third that $G(y) \approx y$ (forward identity) and that $F(x) \approx x$ (backward identity). Its objective is described in Eq. (3).

$$\begin{aligned}\mathcal{L}_{\text{idt}}(G, F) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|\phi(G(y)) - \phi(y)\|_1] \\ + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|\phi(F(x)) - \phi(x)\|_1]\end{aligned} \tag{3}$$

Both generators G and F try to minimize this objective. The identity loss was used only in a specific type of task in (Zhu et al., 2017), photos to paintings. They realized that without this constraint, the model could map the colors in the original photo to arbitrarily chosen new colors in the mapped fake painting. Similarly, we believe that, in language modeling, the generator initially has too much freedom in mapping words from domain X to words in the domain Y. Since we are doing idiolect translation and many words should be mapped to the same words (in the same way that colors in the photos should be mapped to the same colors in the painting), we can use this loss to enhance the learning of the generators.

2.4.4. Full Objective

The full objective is a waited sum of the previous objectives, in which λ_{cyc} and λ_{idt} control the weight of each objective, as show in Eq. (4).

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ + \lambda_{\text{cyc}}\mathcal{L}_{\text{cyc}}(F, G) + \lambda_{\text{idt}}\mathcal{L}_{\text{idt}}(F, G)\end{aligned} \tag{4}$$

The generators G and F try to minimize this objective, while the discriminators D_Y and D_X try to maximize it. Therefore, to find optimal values for G and F, our model tries to solve Eq. (5).

$$G^*, F^* = \arg\min_{G,F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y) \tag{5}$$

2.5. Training Details

The training loop is divided in two parts. For each batch of data, we first use it to optimize the generators, and then to optimize the discriminators.

2.5.1. Generators

To optimize the generators, we simply need to calculate the loss function $\mathcal{L}$ and back-propagate the gradient loss to the

weights of G and F. We use Adam to perform the learning steps.

The first relevant detail is that, in order to achieve better stabilization and higher quality results, we replaced the log likelihood objective in L_{GAN} bya a least-square loss, just like in (Zhu et al., 2017).

The second relevant detail is our choice of ϕ for $\mathcal{L}_{\text{cyc}}$ and $\mathcal{L}_{\text{idt}}$. Instead of pre-training an embedding in the data, we decided to use the shared embedding layer of the decoders of F and G. Our decision was based on two considerations.

First, we can use the CycleGAN architecture to continually improve the representativeness of ϕ by making it a parameter to be optimized by the model. We could not optimize it if we left it untied to either the generators or discriminators, because the loss would not be pressuring it to improve.

Second, we chose to tie it to the embedding layer of the generators because to pressure that as the models learn new embedded representations for the words, these representations will still be reflected in the objective. Otherwise, words that the model could find out to be very similar could be treated as very different in ϕ, in which way $\mathcal{L}_{\text{cyc}}$ and $\mathcal{L}_{\text{idt}}$ would not enforce the constraints they should.

2.5.2. Discriminators

The discriminators need only to learn to be better classifiers, and thus can be evaluated just with respect to $\mathcal{L}_{\text{GAN}}$. Each of them will be optimized using a batch of real data and a batch of fake translated data.

The only detail that must be noticed is that we apply the strategy presented in (Shrivastava et al., 2017) to reduce model oscillation. Instead of of using the batch of fake data that has just been produced by the latest generators, we maintain a memory buffer with the 50 previously translated tweets for each domain X and Y, and use them to optimize the discriminators.

2.6. Hardware Used

The model was run on a virtual machine in Google Cloud Platform with 8 vCPUs, 30 GB of memory, and 1 NVIDIA Tesla T4 GPU. The results that will be presented in the next section were obtained after 1h40min of training.

3. Experimental Results

Unfortunately, our experiments showed results far from desirable. We will report our best experiment with regards to final loss for the generators, and use it to discuss the practical difficulties of training our model.

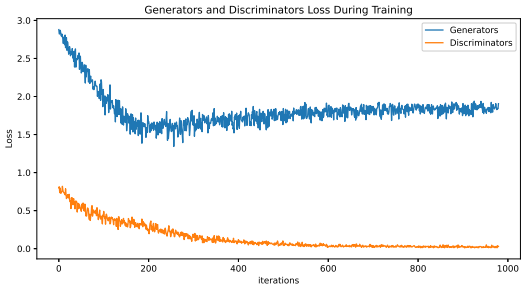

Figure 2. Objective loss for the generators and discriminators over the training epochs.

3.1. Experiment Settings

In our experiment, we used a batch size $N = 20$, which gave as 98 batches of real tweets from Trump and from Sanders in each epoch. We trained for 10 epochs, and the learning rate was constant for generators and discriminators until the 4^{th} epoch, after which it started to decay exponentially. The initial learning rate was 0.0002 for the generators and 0.00002 for the discriminators.

The generators had a embedded size $E_G = 256$, a number of heads $H = 8$, a number of transformer encoder and decoder layers $M = 6$, and a hidden size $F_G = 1024$ in the fully connected layer of these encoders and decoders. We also used a dropout $p = 0.1$ in the embedding layers of the encoder and decoder.

The discriminator was very simple and had a embedded size $E_D = 32$, and a hidden size $F_D = 64$ in the fully connected layer of its transformer encoder.

We used objective weights $\lambda_{\text{cyc}} = 10$ and $\lambda_{\text{idt}} = 5$.

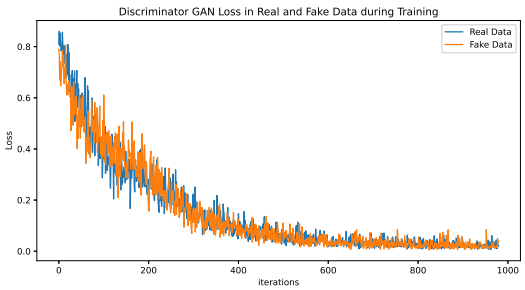

Figure 3. Adversarial loss for the discriminators over the training epochs in real and fake translated tweets.

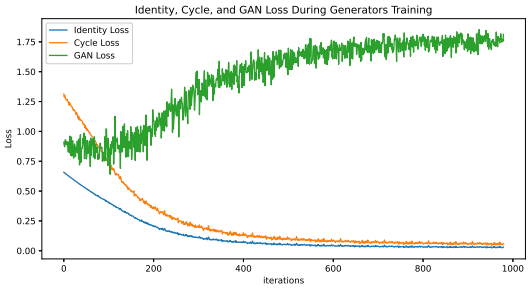

Figure 4. Adversarial, Cycle, and Identity loss for the generators over the training epochs

3.2. Training Loss

The generator was not able to compute with the discriminator, even though the former was much more complex than the latter, and had a ten times larger training step. We can see in Figure 2 that the discriminators quickly converge to very small loss values, while the generators cannot perform as well. Notice that even though we present the objective loss as something to be maximized by the discriminators, we implement it such that both the generators and discriminators want to minimize the values showed in all plots presented.

Furthermore we can see in Figure 3 that the discriminators achieve a very low loss both in real and in fake translated tweets.

When we analyze the weighted adversarial, cycle, and identity loss of the generators in Figure 4 we see that they are having no problems in minimizing cycle and identity loss, but that their adversarial loss actually increases over time.

3.3. Translation Example

The Trumpese-to-Sanderese translation performed by this model is anecdotal. The translated sentences have no meaning and mostly consist of repeated words. For example, when given the following tweet from trump shown in Figure 5, the model generates the Sanders tweet shown in Figure 6.

Figure 5. Example of real Trump tweet

Figure 6. Example of fake Sanders tweet generated by the model as a translation from the Trump tweet in Figure 5

4. Conclusion

The model did not produce the expected results. Specifically, it was not able to overcome a much simpler discriminator, and could not produce coherent text. However, there are some difficulties that can be pointed out and further assessed before we conclude that this model is unable to perform this machine translation task, and we will work with them in the future with more time.

First, GANs are hard to train, and CycleGANs are even harder. They depend on hyperparameters that can provide a very subtle equilibrium between generator and discriminator that allows for a nurturing competitions that leads both of them to systematically improve at similar paces. Maybe some other set of hyperparameters can achieve better results. Furthermore, even though differences in learning rates did not affect the final outcome in our experiments, maybe performing more training steps in the generator per epoch would allow for it to reach the discriminator.

Second, performing a translation with the transformer is a recursive task. Since we need a full translation for the discriminator to evaluate, we need to perform the step of choosing the most likely next token in the translation until it is over, and this is a difficult function to back-propagate. The $\arg\max$ function is not even completely differentiable. If we could find a way to train the Generator using a single next prediction, we believe it would perform much better.

Third, we need to understand why the model can not even produce coherent sentences, since it could just copy the input data to the output. The generator needs to somehow be able to produce a minimally coherent language model in order to perform the translation task. Maybe pre-training it on a language modeling task or increasing the weight of $\mathcal{L}_{\text{idt}}$ might yield better results.

Since this is the most basic machine translation task that can be performed in natural language, we shouldn not *fire* this model. I believe it is still possible to address the aforementioned issues and make this model do a *great job!* in this task.

References

Brunner, G., Wang, Y., Wattenhofer, R., and Zhao, S. Symbolic music genre transfer with cyclegan. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 786–793. IEEE, 2018.

Kaneko, T., Kameoka, H., Tanaka, K., and Hojo, N. Cyclegan-vc2: Improved cyclegan-based non-parallel voice conversion. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6820–6824. IEEE, 2019.

Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., and Webb, R. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2107–2116, 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.