Text Style Transfer

Gao Han
gh352@stanford.edu

Zhuoming Li
zli342@stanford.edu

***Abstract*— Text style transfer is a technique to rewrite sentences from one style to a different style while at the same time preserving the semantic contents. The main challenge of this project is how well we can preserve the content meaning after the style transfer process due to the lack of parallel data that connects the source and target styles. In this project, we use GANs to apply style transfer onto textual contents, trained on tweets scraped from Obama and Trump's twitter account. Our style transfer generator is able to rewrite tweets from Trump to ones that has Obama's writing style (based on his tweets) and vise versa, and the generated contents achieves 80% of accuracy based on a classifier we trained to distinguish tweets from Obama and from Trump.**

I. Motivation

In many situations, for a robust language generation system, flexible control over its expression is necessary. Contents whether artificial or not, may need to be expressed in different ways. In the case of anonymization, writings should be converted to a style that's neutral and common. Another example is to help with content understanding - for instance politicians often prefer languages of vagueness and exaggerations, and with style transfer, its possible to strip away the hyperbole to expose the real meaning expressed in plain English. In general, this transformation of style is also closely relevant to dialogue generation, machine translation, and writing assistant applications. This motivates us to build a text style transfer model, with a goal to turn an input sentence into a new style, while preserving the style independent content.

II. Related Work

There have been some studies on text style transfer using various approaches. Shen et al. [1] used variational auto-encoders (VAEs) with GANs trained using non-parrallel texts. They use cross-aligning auto-encoder model which consists of an encoding step to infer the content and a decoding step to generate the sentence with the target style. Then the generated sentence teacher-forced by the original words is compared with the transferred sentence self-fed by previous output logits using a discriminator.

Yang1 et al. [2] also trained a text style transfer with GANs but with a different loss function which is the negative log likelihood (NLL). They find that this algorithm achieves slightly greater BLEU scores than Shen's model. The main difference is that they use an implicitly trained language model as a new type of discriminator, replacing the more conventional binary classifier. The motivation for this change is that error signals from a binary classifier is sometimes insufficient to train the generator to produce fluent language, and optimization can be unstable as a result of the adversarial training step.

III. Dataset and Features

A. Dataset Overview

Since we plan to do text style transfer for Barack Obama and Donald Trump, we obtained their data by scraping tweets from their Twitter accounts @BarackObama and @realDonaldTrump. We use the twitter API to scrape all tweets with a scraper here. Specifically, the data for Trump is from 2017/1/1 to 2018/12/31 (6730 tweets), and the data for Obama is from 2012/1/1 to 2013/12/31 (5275 tweets). The date range is chosen based on their incumbency, because for other dates the data is sparse and the style can be different. For example, prior to 2016, Trump seldomly uses make america great again, and after 2017 Obama only sends a few tweets in a month and is mostly about his family resulting in less content match).

B. Data Preprocessing

Since tweets are somewhat different from normal writings and contains twitter specific marks, we need the following processing before we use the data for training. 1. Mask all targets (), hashtags (#), and numbers to decrease OOV count. 2. Insert spaces to punctuation so that they will be treated as words. This is very important as many punctuation has sentiment and can represent ones writing style (such as ! at the end of the sentence, which is a very strong signal). 3. Break each tweet into several examples, each containing exactly one sentence. We observed that many tweets are fairly long (more than 3 sentences and 20 words), and long ones will be especially hard to train or to get good results. In addition, chopping tweets into several examples can increase the size of our training set as well. 4. Remove irrelevant contents. Though tweets from Trumps account are always written by himself (at least it looks like so), many tweets from Obamas account are drafted by his staff and are normally in the format of President Obama says, which adds noise to the training set. Since the content inside these tweets are still useful data, we removed the prefix and kept the content for these tweets.

C. Label Definition

Tweets downloaded from Obama are labeled as 0 and Tweets from Trump are labeled as 1.

IV. Model Representation

A. Architecture Overview

On a high level, we would like to convert Obama text into its style-free latent representation. Given the latent representation, we want to reconstruct its original Obama style as well as transfer into Trump style as shown in the figure below.

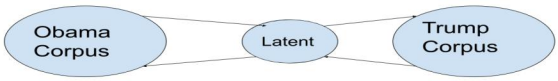

Fig. 1: Transfer between different corpora

Based on Shen et al. [1]'s work, the model consists of an encoder E, a generator G and two discriminators $D1$, $D2$ as illustrated in figure 2.

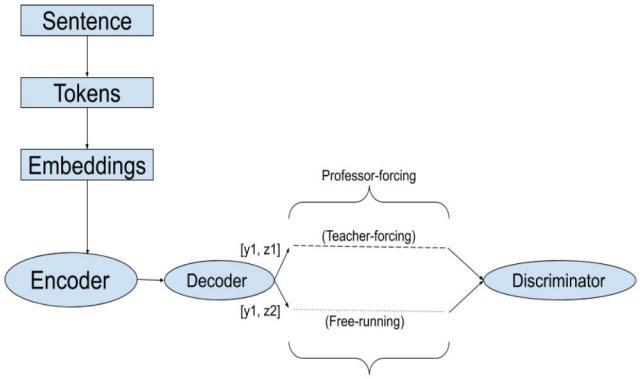

Fig. 2: Model overview

As part of the problem setup, we assume our focus is on two different styles y_1 and y_2. Sentences x_1 and x_2 from Tweets of each style are first broken down into tokens, which then are transformed into embeddings. For embeddings, it's possible to co-train them together with the model or make use of existing pre-trained word embeddings, such as Google News Word2Vec or GloVe embeddings. Each approach has its own pros and cons, and we will detail more of that based on our experiments in the following sections.

Encoder takes word embeddings as input and outputs the corresponding latent representation z. After having obtained latent z, we unroll decoder (a.k.a generator) on latent representation z to generate two sequences of intermediary states: one using teacher-forcing and the other using free-running approach that utilizes the Softmax output of the RNN cell from previous timestamp.

As the last step, we apply discriminator to differentiate between the two sequences, which aims to align the latent space during generation to ensure the transferred x_2 (now of style y_1) matches the population of the original x_1.

B. Model Optimization

There are two objectives that we strive to optimize for. The first one is the reconstruction loss, which measures how well the model performs in terms of transferring x_1 to style y_2 and back to y_1 without losing much of its original content. This property is captured in the following equation and log is added to ensure numerical stability.

$$\begin{aligned}\mathcal{L}_{rec}(\theta_E, \theta_G) &= \mathbb{E}_{x_1 \sim X_1}[-\log p_G(x_1|y_1, E(x_1, y_1))] + \\ &= \mathbb{E}_{x_2 \sim X_2}[-\log p_G(x_2|y_2, E(x_2, y_2))]\end{aligned}$$

The second objective that we care for is the adversarial loss, which aligns the latent space for the transfer of y_1 to y_2 and vice versa. In our case, D_1's job is to distinguish between the real x_1 and transferred x_2, and D_2's used to distinguish between real x_2 and transferred x_1. Together with Professor-forcing technique, the latent space for the two scenarios can be aligned.

$$\mathcal{L}_{adv_1} = -\frac{1}{k}\sum_{i=1}^{k}\log D_1(h_1^{(i)}) - \frac{1}{k}\sum_{i=1}^{k}\log(1 - D_1(\tilde{h}_2^{(i)}))$$

To combine reconstruction loss $mathcal L_{rec}$ and two adversarial losses $\mathcal{L}_{adv_1}$ and $\mathcal{L}_{adv_2}$, we arrive at the final objective of the model:

$$\mathcal{L}_{rec} - \lambda(\mathcal{L}_{adv_1} + \mathcal{L}_{adv_2})$$

V. Results and Discussion

A. Co-trained embedding

We started off with co-training embeddings together with the model. The advantage of this approach is that we have control over the dimensions of the embedding space. Otherwise, we will have to employ dimension reduction techniques, such as random projection (Gaussian or Sparse) [3] or PCA, to adjust the dimension to our specific application, which adds extra overhead and likely introduce noise in the process.

For training, we used the recommended hyper-parameter setting from the original paper and obtained the following graphs for reconstruction loss, adversarial loss and total loss:

Both reconstruction loss and total loss are trending downwards with adversarial loss going up. The training process overall reduced total loss from 76.86 to 68.15 after 20 epochs, and reduced reconstruction loss from 74.80 to 63.89 after 20 epochs.

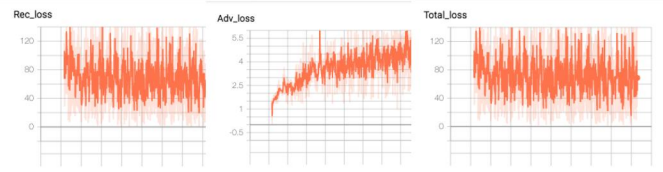

Fig. 3: Graphs for training losses with co-trained embeddings

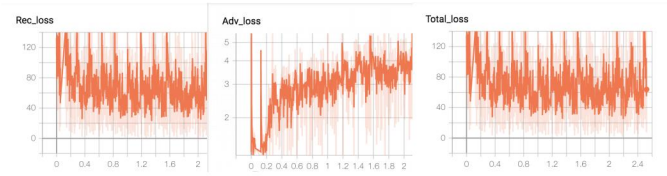

Fig. 4: Graphs for training losses with pre-trained embeddings

We sampled some examples from the trained model. The following table consists of 3 parts: the original input, the reconstructed output of original style and the transferred. The input has two parts: an label of 0 and 1, which refers to Obama and Trump style respectively; and the input sentence to be reconstructed or transferred.

Input	Reconstruction	Transferred
1 make america great again	thank you !	president obama
0 tell congress to get the american	we want to get the american people .	i will be a great job !
0 put it up for a vote .	it's time for the american people .	the u . s .
1 we must maintain a strong southern border .	we will be a great job .	we want to do the american people .

The reconstructed and transferred results are still far from ideal, but they have captured the style of each corpus to a certain degree. For instance, *tell congress to get the american* is referring to encourage congress to get more american people to vote and the reconstructed sentence was able to keep this formation and expanded it further with *american people*. The transferred counterpart is also able to embody the flamboyant Trump style of *i will be a great job !*.

The first example from the above table is attempting to transfer Trump's campaign slogan *make america great again* into Obama's *Yes, we can*. However, the model is not able to perform the transfer likely due to the size of our dataset as well as it would need extra contextual information to enable the model to realize the slogans are counterparts of each other, which is a really difficult learning task.

B. Pre-trained embedding

After experimenting with co-training embeddings, we decided to try out pre-trained embeddings. Google News Word2Vec embeddings is chosen because its ease of use and high quality. Google News embeddings have dimensions of 300 which is bigger than our co-training embedding of dimension 100. As part of the future work, we can choose to reduce its dimension via random project or PCA, which can in turn , speed up training and likely reduce variance.

The reconstruction loss and total loss follow the same patterns as previous model training with co-trained embeddings. With pre-trained embeddings, the model is able to reduce total loss to 66.75 and reconstruction loss to 62.24, which are lower than the co-training case. This observation stays the same after we have repeated the training for each case multiple times, which indicates that the pre-trained Word2Vec embeddings are helping the model to better learn the relationships between words and to better approximate the underlying probability distribution.

We performed the same sampling experiments with the pre-trained embedding and obtained the following contents:

Input	Reconstruction	Transferred
1 make america great again	thank you to vote !	add your name:
0 tell congress to get the american	add your name to make your voice to get up .	thank you to see the house of our country .
0 put it up for a vote .	it's time to see the ⟨ unk ⟩.	in the world .
1 we must maintain a strong southern border .	we will be a great job .	we need to make our country .

Generally speaking, with pre-trained embedding, the model is able to generate longer and more sensible sentences. For instance, *make america great again* is reconstructed into *thank you to vote!*, which includes *to vote!* with an exclamation point that is known to be a consistent theme of Trump's style.

For the second example, the input *tell congress to get the american* is reconstructed as *add your name to mark your voice to get up*. The model understands the underlying latent meaning to get people to vote and rephrased it in a different way.

VI. Conclusion

Making style transfer on texts has started to gain attention recently using both parallel data and non-parallel data. In this work, we are able to perform text style transfer on tweets from Trump and Obama by formulating the task as a decipherment problem using non-parallel data scraped from Twitter websites. We optimize neural networks by forcing the distribution alignment over the latent space, and showed the effectiveness of our method with style evaluation (using a style classification) and human evaluation (by evaluating the quality of generated sentences). The actual task is proven to be quite difficult and the results are not as desirable as we originally expected them to be. This is likely because that the writing style of Trump and Obama are too complex

and subtle for the network to learn and generalize. On top of that, the available training data is limited and there are only so much usable Tweets from Trump and Obama given that we are only interested in the duration when they are in office and minus the filtering criteria.

VII. Contributions

The two of us paired up on all components of this project, including dataset cleaning, feature engineering, model formulation / evaluation, and the write-up of this report and the poster.

Codebase: `https://zhuoming_li@bitbucket.org/zhuoming_li/style_transfer.git`

References

[1] R. B. T. J. Tianxiao Shen, Tao Lei, "Style transfer from non-parallel text by cross-alignment," 2017.

[2] C. D. E. P. X. T. B.-K. Zichao Yang, Zhiting Hu, "Unsupervised text style transfer using language models as discriminators," 2018.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.