Style Transfer with Disentangled ARAEs

Alex Lin Melissa Yu

Abstract

In this project, we examine the *style transfer* problem, which asks us to change one attribute of a sentence (e.g. sentiment) without altering the overall style and grammar of that sentence. Our approach involves the Adversarially Regularized Autoencoder (ARAE) (Zhao et al., 2017) to explicitly disentangle the latent attribute of interest from the other stylistic attributes that we wish to keep. We compare the performance of this Disentangled ARAE to results from the original ARAE on the Yelp dataset.

1. Introduction

This project explores the *style transfer* problem. Given a source sentence and some attribute of that sentence (e.g. sentiment, formality, time period), we wish to produce another sentence that has the same overall style and grammar of the original sentence, yet differs in its attribute of interest. For example, if the attribute of interest is formality, here might be an example of style transfer:

Informal: *I'd say it is punk though.*
Formal: *However, I do believe it to be punk.*

The first sentence is informal, yet this lack of formality is stripped away in the second sentence, which holds the same structure and meaning. Here is another example with Shakespearean English vs. Modern English.

Shakespearean English: *So thrive my soul –*
Modern English: *My soul depends on it –*

Having a model that can freely switch and translate between these modes can be quite useful. There have been models proposed that tackle this exact problem in natural language processing, such as the Adversarially Regularized Autoencoder (ARAE) (Zhao et al., 2017). In this work, we change some aspects of this model to formulate the Disentangled ARAE for the style transfer problem. Section 2 covers some background and Section 3 presents related work on the problem. Section 4 formally presents the mathematical model and Section 5 covers the training procedure. Section 6 covers methods, and Section 7 covers results and a discussion.

2. Background

While latent variable models have enjoyed success in tackling computer vision problems, they have generally been less effective in the domain of natural language processing (NLP). One particularly useful application of latent variable models to NLP is the *style transfer* problem, in which we seek to transfer the content of a source sentence to a new sentence that follows certain presentation constraints specified by the user. Specifically, in this paper we consider the problem of sentiment transfer, in which we seek to preserve the content and style of a sentence but modify its sentiment from positive to negative, or vice versa.

The problem of *unaligned style transfer* seeks to perform style transfer without using parallel data (like that typically used for sentence translation or summarization), but instead, simply training on a non-parallel corpus of sentences which cover the same content distributions, but are not explicitly paired.

3. Related Work

3.0.1. Generative Adversarial Networks

The generative adversarial network (GAN) (Goodfellow et al., 2014) framework establishes a min-max game between two neural networks, the generator G and the discriminator D, in order to match the generative process to the data distribution. Suppose the generator network $G(\boldsymbol{z})$ maps some hidden representation $\boldsymbol{z}$ to the data space, and the discriminator network $D(\boldsymbol{x})$ predicts the probability that a sample $\boldsymbol{x}$ comes from the true data distribution p_d (positive samples), and not G's generative process (negative samples). The objective is

$$\min_G \max_D \; \mathbb{E}_{p_d(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{p(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \tag{1}$$

The network consists of the discriminator, which achieves the best classification accuracy over samples of $\boldsymbol{x}$, and the generator, which maximally confuses this best discriminator, so that $D(\boldsymbol{x}) = 1/2$.

3.0.2. Semi-Supervised Adversarial Autoencoders

Adversarial autoencoders (AAE) (Makhzani et al., 2015) are probabilistic autoencoders which perform variational inference by utilizing generative adversarial networks to regularize the encoder. Let $\boldsymbol{x}$ be the input following the true data distribution $p_d(\boldsymbol{x})$, and let $\boldsymbol{z}$ be the latent code vector generated by the autoencoder. We wish to approximate the encoding distribution $q(\boldsymbol{z} \mid \boldsymbol{x})$, informed by the prior $p(\boldsymbol{z})$. Additionally, we would also like to estimate the decoding distribution $p(\boldsymbol{x} \mid \boldsymbol{z})$.

Recall the variational autoencoder's objective (Kingma & Welling, 2013)

$$\min_q \ \mathbb{E}_{q(\boldsymbol{z} \mid \boldsymbol{x})}[-\log p(\boldsymbol{x} \mid \boldsymbol{z})] + KL(q(\boldsymbol{z} \mid \boldsymbol{x}) \mid\mid p(\boldsymbol{z})) \quad (2)$$

AAE simply replaces the KL divergence term with an adversarial network on top of the latent code, which guides the aggregated posterior

$$q(\boldsymbol{z}) = \int_{\boldsymbol{x}} q(\boldsymbol{z} \mid \boldsymbol{x}) p_d(\boldsymbol{x}) d\boldsymbol{x} \quad (3)$$

to match the prior $p(\boldsymbol{z})$. In this model, the encoder of the autoencoder is also the generator of the adversarial network: $G(\boldsymbol{z}) = q(\boldsymbol{z} \mid \boldsymbol{x})$. The model has two objectives; the first seeks to minimize the reconstruction error $\mathcal{L}(\boldsymbol{x}, \boldsymbol{x}')$ of the network, while the second GAN objective seeks to match the generative process for $\boldsymbol{z}$ with the prior process.

Semi-supervised AAE leverages the generative description of unlabeled data to achieve better classification performance than can be obtained using only labeled data. In this scheme, data $\boldsymbol{x}$ is generated from a combination of a latent class vector $\boldsymbol{y} \sim p_y(\boldsymbol{y})$ and a latent "style" vector $\boldsymbol{z} \sim p_z(\boldsymbol{z})$. The encoder is thus $q(\boldsymbol{y}, \boldsymbol{z} \mid \boldsymbol{y})$ and the decoder is $p(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z})$. Adversarial networks $D_y(\boldsymbol{y})$ and $D_z(\boldsymbol{z})$ are attached on top of both latent codes to enforce the two prior distributions.

4. Model

The basic ARAE architecture follows adversarial autoencoders (AAEs) in combining variational autoencoders (VAEs) with GANs (generative adversarial networks) (Zhao et al., 2017). A diagram of how this model can be used for unaligned style transfer can be found in Figure 1.

Given a vocabulary $\mathcal{V}$, each input to the ARAE is a sentence $\boldsymbol{x} = \{x_1, x_2, \ldots\}$ where each word $x_t \in \mathcal{V}$. We define $\mathcal{X} = \mathcal{V}^T$ (where T is the variable length of a sentence) as the observation space and $\mathcal{C}$ as the latent code space. In the VAE part of the ARAE, $\boldsymbol{x}$ is passed through an encoder $q_\phi : \mathcal{X} \to \mathcal{C}$ to form a latent code $\boldsymbol{c} = q_\phi(\boldsymbol{x})$. This code represents the stylistic attributes of the sentence.

In the style transfer problem, we assume that $\boldsymbol{x}$ has some true attribute/class of interest $y^* \in \mathcal{Y}$. We wish to generate some $\tilde{\boldsymbol{x}}$ that has a different attribute/class $\tilde{y} \neq y^*$ such that its corresponding stylistic attributes except for $\tilde{y}$ are the same as those of $\boldsymbol{x}$.

Note that q is represented in our architecture as a neural network with parameters ϕ.

The latent code is then put through a classifier $f_u : \mathcal{C} \to [0, 1]$ that returns the probability of the sentence having the attribute of interest. (Note that in this case, we assume that the attribute is binary – e.g. positive or negative sentiment). Similarly, f is also a neural network with parameters u.

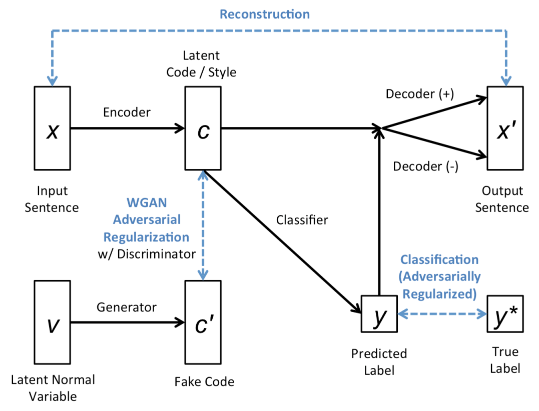
Let the decision of the classifier be $y \in \{0, 1\}$, where 1 corresponds to one class and 0 correspond to the other. During classification training, this decision is compared to the true label $y^* \in \{0, 1\}$. This classifier is in the ARAE architecture so that we can use the encoder to adversarially train against it, thereby ensuring that $\boldsymbol{c}$ does not hold any information about y^* – a property that is necessary for the style transfer procedure.

Then, conditioning on the latent code $\boldsymbol{c}$ and the true class y^*, the decoder $r_\psi : \mathcal{C} \times \{0, 1\} \to \mathcal{X}$ returns a reconstructed sentence $\boldsymbol{x}' = r_\psi(\boldsymbol{c}, y^*)$. Note that if we wanted to transfer the style of the sentence to a sentence in the other class, we could simply decode with $r_\psi(\boldsymbol{c}, 1 - y^*)$. Again, r is a neural network (or a series of neural networks) with parameters ψ. See Section 5 for additional details.

The latent code $\boldsymbol{c}$ is adversarially regularized by a GAN so that its distribution does not deviate too far from a flexible prior. The base distribution of the GAN is $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ where $\boldsymbol{I}$ is the identity matrix. A generator $g_\theta : \mathbb{R}^{\dim(\boldsymbol{I})} \to \mathcal{C}$ takes samples $\boldsymbol{v} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and tries to fool a corresponding discriminator $d_w : \mathcal{C} \to [0, 1]$ by using fake codes $\boldsymbol{c}' = g_\theta(\boldsymbol{v})$. Both the generator g and the discriminator d are neural networks with parameters θ and w respectively.

The Disentangled ARAE, our proposed modification to this model, aims to explicitly separate the attribute of interest y^* from the style of the sentence. Here, "style" refers to all latent variables except for y^*. The motivation is that working with this structure allows us to not have to adversarially train the classifier, which could be unstable and also increase the computational time of training the model. Our inspiration comes from the completely disentangled structure of the original Adversarial Autoencoder that was used for supervised and semi-supervised training (Makhzani et al., 2015).

The main change that we make is to have the latent code $\boldsymbol{c}$ be passed through a multilayer perceptron $a_\alpha : \mathcal{C} \to \mathcal{Z}$ where α is the parameters of the network and $\mathcal{Z}$ is the latent style space. The result is the latent style $\boldsymbol{z} \in \mathcal{Z}$. Separately,



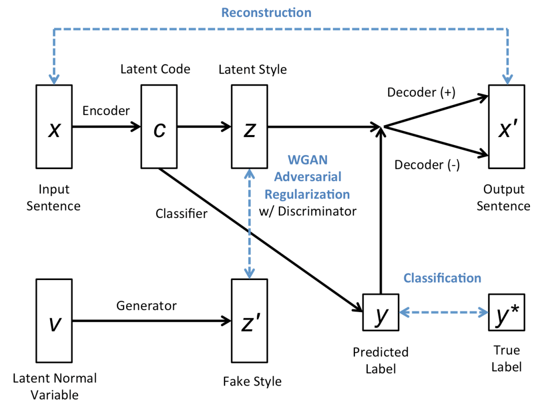


Figure 1. Diagrams and training procedures for ARAE (*top*) and our modified Disentangled ARAE (*bottom*).

$\boldsymbol{c}$ is also passed through the classifier f_u to generate y from $f_u(\boldsymbol{c})$. Thus, we have disentangled the style from the class.

The style $\boldsymbol{z}$ (not the code $\boldsymbol{c}$) will now be passed along with the class y to the decoder, redefined as $r_\psi : \mathcal{Z} \times \{0,1\} \to \mathcal{X}$. In addition, the generator $g_\theta : \mathbb{R}^{\dim(\boldsymbol{I})} \to \mathcal{Z}$ of the Disentangled ARAE is used to regularize $\boldsymbol{z}$ instead of $\boldsymbol{c}$. See Figure 1 for the architecture of the model.

5. Training

The ARAE trains its various components in several different steps so that it can construct a desirable latent code space $\mathcal{C}$. First, the ARAE trains the reconstruction loss of the autoencoder. This loss is defined as the cross-entropy between the data's true distribution and the model's assigned distribution

$$\begin{aligned}\mathcal{L}_{\text{rec}}(\boldsymbol{x}) &= y^* \cdot \log r_\psi(q_\phi(\boldsymbol{x}), y^*) \\ &\quad + (1-y^*) \cdot \log\left[1 - r_\psi(q_\phi(\boldsymbol{x}), y^*)\right]\end{aligned}$$

Note in Figure 1 that there are two decoders in style transfer – one for the positive class $r_\psi^1 : \mathcal{C} \to \mathcal{X}$ and one for the negative class $r_\psi^0 : \mathcal{C} \to \mathcal{X}$. In fact, during training, we choose the decoder to use for $\boldsymbol{x}$ based on the true class y^*. That is,

$$r_\psi(\boldsymbol{c}, y^*) = r_\psi^{y^*}(\boldsymbol{c})$$

During reconstruction training, we backpropagate the gradients of $\mathcal{L}_{\text{rec}}$ to update the encoder's parameters ϕ and the decoder's parameters ψ.

Second, the ARAE trains the discriminator to minimize its loss function. Given a true latent code $\boldsymbol{c}$, this is

$$\mathcal{L}_{\text{disc}}^{\text{true}}(\boldsymbol{c}) = \log d_w(\boldsymbol{c})$$

whereas if given a generated latent code $\boldsymbol{c'}$ this is

$$\mathcal{L}_{\text{disc}}^{\text{fake}}(\boldsymbol{c'}) = \log\left[1 - d_w(\boldsymbol{c'})\right]$$

The discriminator's parameters w are then updated during backpropagation and also contained within the ball $[-\epsilon, \epsilon]^{\dim(w)}$ for some hyperparameter ϵ.

Third, the ARAE trains the classifier to minimize its loss function.

$$\mathcal{L}_{\text{class}}(\boldsymbol{c}, y^*) = y^* \cdot \log f_u(\boldsymbol{c}) + (1-y^*) \cdot \log[1 - f_u(\boldsymbol{c})]$$

The classifier's parameters u are then updated during backpropagation.

Fourth, the ARAE trains the encoder adversarially to maximize the classifier's loss function and minimize

$$-\mathcal{L}_{\text{class}}(q_\phi(\boldsymbol{x}), y^*)$$

The encoder's parameters ϕ are then updated during backpropagation.

And finally, the ARAE trains both the encoder and the generator adversarially to maximize the discriminator's loss function. The encoder wishes to minimize

$$-\mathcal{L}_{\text{disc}}^{\text{true}}(q_\phi(\boldsymbol{x}))$$

whereas the generator wishes to minimize

$$-\mathcal{L}_{\text{disc}}^{\text{fake}}(g_\theta(\boldsymbol{v}))$$

where $\boldsymbol{v} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Again, we do backpropagation on these losses to update ϕ for the encoder and θ for the generator. The full algorithm for ARAE is given in Algorithm 1.

Disentangled ARAE is slightly different from the original ARAE setup. First, Disentangled ARAE also trains reconstruction, but has an intermediate MLP converting $\boldsymbol{c}$ to $\boldsymbol{z}$. That is,

$$\begin{aligned}\mathcal{L}_{\text{rec}}(\boldsymbol{x}) &= y^* \cdot \log r_\psi(a_\alpha((q_\phi(\boldsymbol{x})), y^*) \\ &\quad + (1-y^*) \cdot \log\left[1 - r_\psi(a_\alpha(q_\phi(\boldsymbol{x})), y^*)\right]\end{aligned}$$

Again, there are two decoders – one for each class. However, they map $\mathcal{Z}$ to $\mathcal{X}$ instead. In addition, when we backpropagate the gradients, we are tuning α for a in addition to the parameters of the encoder and the decoder.

Second, the discriminator's loss in Disentangled ARAE is defined on the latent style $\boldsymbol{z}$ instead of the latent code $\boldsymbol{c}$.

Third, the classifier in Disentangled ARAE minimizes its loss function in the same way as before. However, there is no adversarial training of the encoder, because part of the objective of the project was to investigate the necessity of this step for the style transfer problem.

Finally, both the encoder and the generator are trained adversarially, yet there is an MLP in between the discriminator and the encoder. That is, the encoder wishes to minimize

$$-\mathcal{L}_{\text{disc}}^{\text{true}}(a_\alpha(q_\phi(\boldsymbol{x})))$$

whereas the generator wishes to minimize

$$-\mathcal{L}_{\text{disc}}^{\text{fake}}(g_\theta(\boldsymbol{v}))$$

The full details for Disentangled ARAE are in Algorithm 2.

6. Methods

6.1. Data

Following the convention of previous work in the field, we use the Yelp reviews dataset (Cui, 2015) published during the 2015 Yelp Dataset Challenge, which contains real review data collected for 61K businesses across four countries (U.S., UK, Germany and Canada), together with sentiment labels (either "positive" or "negative"). In total, there are 1,569,264 review text samples, with a vocabulary of 9603 words. We split the corpus into two sets of unaligned positive and negative reviews, similarly to Shen et al.; Zhao et al..

In order to facilitate comparison of this approach with others, we use the same training, validation and testing sets used in Shen et al. and Zhao et al.. Specifically, the validation data consists of 38,205 positive and 25,278 negative instances, the testing data of 76,392 positive and 50, 278 negative instances, and the training data of 267,314 positive and 176,787 negative samples.

6.2. Implementation

In the ideal case, our network could learn disentangled latent codes y (regularized to follow a categorical distribution $Cat(k)$) which would capture fine-grained sentiments behind Yelp reviews such as different star ratings. However, we only test binary label representations here (positive or

Algorithm 1 ARAE for Style Transfer

```
for each training iteration do
    (1) Train Reconstruction
    Sample $(\boldsymbol{x}^{(1)}, y^{*(1)}), \ldots, (\boldsymbol{x}^{(M)}, y^{*(M)})$ from data.
    Backprop loss $\frac{1}{M}\sum_{m=1}^{M} \mathcal{L}_{\text{rec}}(\boldsymbol{x}^{(m)})$ to update $\phi, \psi$.
    (2) Train Discriminator
    Sample $(\boldsymbol{x}^{(1)}, y^{*(1)}), \ldots, (\boldsymbol{x}^{(M)}, y^{*(M)})$ from data.
    Compute $\boldsymbol{c}^{(m)} = q_\phi(\boldsymbol{x}^{(m)})$ for $m = 1, \ldots, M$.
    Sample $\boldsymbol{v}^{(m)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ for $m = 1, \ldots, M$.
    Compute $\boldsymbol{c}'^{(m)} = g_\theta(\boldsymbol{v}^{(m)})$ for $m = 1, \ldots, M$.
    Backprop loss $\frac{1}{M}\sum_{m=1}^{M} \mathcal{L}_{\text{disc}}^{\text{true}}(\boldsymbol{c}^{(m)}) + \mathcal{L}_{\text{disc}}^{\text{fake}}(\boldsymbol{c}'^{(m)})$ to update $w$.
    Clip discriminator parameters $w$ to $[-\epsilon, \epsilon]^{\dim(w)}$.
    (3) Train Classifier
    Sample $(\boldsymbol{x}^{(1)}, y^{*(1)}), \ldots, (\boldsymbol{x}^{(M)}, y^{*(M)})$ from data.
    Compute $\boldsymbol{c}^{(m)} = q_\phi(\boldsymbol{x}^{(m)})$ for $m = 1, \ldots, M$.
    Backprop loss $\frac{1}{M}\sum_{m=1}^{M} \mathcal{L}_{\text{class}}(\boldsymbol{c}^{(m)}, y^{*(m)})$ to update $u$.
    (4) Train Encoder Adversarially
    Sample $(\boldsymbol{x}^{(1)}, y^{*(1)}), \ldots, (\boldsymbol{x}^{(M)}, y^{*(M)})$ from data.
    Compute $\boldsymbol{c}^{(m)} = q_\phi(\boldsymbol{x}^{(m)})$ for $m = 1, \ldots, M$.
    Backprop loss $-\frac{1}{M}\sum_{m=1}^{M} \mathcal{L}_{\text{class}}(\boldsymbol{c}^{(m)}, y^{*(m)})$ to update $\phi$.
    (5) Train Encoder and Generator Adversarially
    Sample $(\boldsymbol{x}^{(1)}, y^{*(1)}), \ldots, (\boldsymbol{x}^{(M)}, y^{*(M)})$ from data.
    Sample $\boldsymbol{v}^{(m)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ for $m = 1, \ldots, M$.
    Backprop loss $-\frac{1}{M}\sum_{m=1}^{M} \mathcal{L}_{\text{disc}}^{\text{true}}(q_\phi(\boldsymbol{x}^{(m)}))$ to update $\phi$.
    Backprop loss $-\frac{1}{M}\sum_{m=1}^{M} \mathcal{L}_{\text{disc}}^{\text{fake}}(g_\theta(\boldsymbol{v}^{(m)}))$ to update $\theta$.
end for
```

Algorithm 2 Disentangled ARAE for Style Transfer

```
for each training iteration do
    (1) Train Reconstruction
    Sample $(\boldsymbol{x}^{(1)}, y^{*(1)}), \dots, (\boldsymbol{x}^{(M)}, y^{*(M)})$ from data.
    Backprop loss $\frac{1}{M}\sum_{m=1}^{M}\mathcal{L}_{\text{rec}}(\boldsymbol{x}^{(m)})$ to update $\phi, \psi$.
    (2) Train Discriminator
    Sample $(\boldsymbol{x}^{(1)}, y^{*(1)}), \dots, (\boldsymbol{x}^{(M)}, y^{*(M)})$ from data.
    Compute $\boldsymbol{z}^{(m)} = a_\alpha(q_\phi(\boldsymbol{x}^{(m)}))$ for $m = 1, \dots, M$.
    Sample $\boldsymbol{v}^{(m)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ for $m = 1, \dots, M$.
    Compute $\boldsymbol{z'}^{(m)} = g_\theta(\boldsymbol{v}^{(m)})$ for $m = 1, \dots, M$.
    Backprop loss $\frac{1}{M}\sum_{m=1}^{M}\mathcal{L}_{\text{disc}}^{\text{true}}(\boldsymbol{z}^{(m)}) + \mathcal{L}_{\text{disc}}^{\text{fake}}(\boldsymbol{z'}^{(m)})$ to update $w$.
    Clip discriminator parameters $w$ to $[-\epsilon, \epsilon]^{\dim(w)}$.
    (3) Train Classifier
    Sample $(\boldsymbol{x}^{(1)}, y^{*(1)}), \dots, (\boldsymbol{x}^{(M)}, y^{*(M)})$ from data.
    Compute $\boldsymbol{c}^{(m)} = q_\phi(\boldsymbol{x}^{(m)})$ for $m = 1, \dots, M$.
    Backprop loss $\frac{1}{M}\sum_{m=1}^{M}\mathcal{L}_{\text{class}}(\boldsymbol{c}^{(m)}, y^{*(m)})$ to update $u$.
    (4) Train Encoder and Generator Adversarially
    Sample $(\boldsymbol{x}^{(1)}, y^{*(1)}), \dots, (\boldsymbol{x}^{(M)}, y^{*(M)})$ from data.
    Sample $\boldsymbol{v}^{(m)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ for $m = 1, \dots, M$.
    Backprop loss $-\frac{1}{M}\sum_{m=1}^{M}\mathcal{L}_{\text{disc}}^{\text{true}}(a_\alpha(q_\phi(\boldsymbol{x}^{(m)})))$ to update $\phi$.
    Backprop loss $-\frac{1}{M}\sum_{m=1}^{M}\mathcal{L}_{\text{disc}}^{\text{fake}}(g_\theta(\boldsymbol{v}^{(m)}))$ to update $\theta$.
end for
```

negative reviews). The remainder of our model parameters largely follow those specified for Yelp/Yahoo style transfer in Zhao et al.: For the latent noise representation $\boldsymbol{z}$, we use 500 dimensions. Both the encoder and decoder are parameterized as one-layer LSTM's, with 500 hidden units each. We train two decoders, one for each style label. Both the MLP for learning $\boldsymbol{z}$ from $\boldsymbol{c}$ and style classifier are three-layer MLP's with hidden layer sizes 300-200-100 and ReLU activations.

The parameters are optimized in mini-batches of size 64 using Stochastic Gradient Descent for the autoencoder (including the MLP for $\boldsymbol{z}$) and classifier, while the Adam algorithm is used for the discriminator and generator.

Our model is implemented on top of the Python / Pytorch software for ARAE published by the authors (Zhao et al., 2017). We adapt their work for our use by modifying the network architecture and training regime as described in previous sections. Baseline statistics are taken from previously published works in the field (Shen et al., 2017; Zhao et al., 2017).

6.3. Evaluation

We report three metrics for our models and compare them to the results reported by Zhao et al.:

Transfer: A measure of how successfully a model has altered the sentiment of a sentence. We calculate this metric by training a supervised binary sentiment classifier on real sentences (comprising the entire validation set described earlier) using the `fastText` library (Joulin et al., 2016), and calculating the multi-class precision and recall (micro-averaged over both classes) after applying the classifier to transferred sentences in the test set.

Perplexity: A measure of how fluent the generated sentences are. We calculate this metric by training a 5-gram language model with modified Kneser-Ney smoothing on real sentences using the KenLM library (Heafield, 2011), and then calculating its perplexity on transferred sentences for both classes.

BLEU: A measure of how consistent the transferred text is, compared to the source.

7. Results and Discussion

The metric results for the previously described model settings are shown in Table 7, after 10 epochs of training. For comparison, the table also displays metrics for several other recently published sentiment transfer models in the literature, all trained and tested on the same Yelp reviews dataset we utilize.

Further, we also present a random sample of transferred sentences from our test set in Table 7.

Examining Table 7, we can see that our model achieves the highest transfer percentage, compared to other models, by over 10%, indicating that the model consistently successfully alters the sentiment of sentences between the two classes. This is not surprising, given our use of two decoders, one for each class. However, the perplexity of our model is higher than both the AE and ARAE models, an attribute which can be seen by examining the generated sentences, which are less fluent than their sources. Moreover, our model also achieves a very low BLEU score – the transferred sentences are quite unlike the originals.

Overall, our findings constitute a negative result, perhaps indicating that the latent codes z have failed to fully capture the style and structure of the original source sentences. ARAE's adversarial training of the encoder to minimize the classifier's accuracy offers a way of directly disentangling the latent codes $\boldsymbol{z}$ from the sentiment of the sentence, and because of this modification, outperforms our model at a very small expense of lower transfer percentage. This is surprising, because we initially expected the main contribution of the adversarial training of the encoder to simply be removing all sentiment information from the latent representation. Based on our results, it appears that the adversarial training not only removes sentiment information,

Classifier	Transfer	Perplexity	BLEU
Cross-Aligned AE	77.1%	65.9	17.75
AE	59.3%	31.9	37.28
ARAE ($\lambda^{(1)} = 1$)	73.4%	29.7	31.15
ARAE ($\lambda^{(1)} = 10$)	81.8%	27.7	20.18
Ours	95.9%	34.7	5.16

Table 1. Baseline sentiment transfer performance on the Yelp reviews dataset (Cui, 2015), compared to our method. Our results are averaged over 5 trials. The other results are taken from Zhao et al..

Positive ⇒ Negative	go there , eat , enjoy !	go there , get bad service !
	we will be back each and everytime we want to eat out !	we will be back and we are never going to eat there !
	the food is excellent and the service is exceptional !	the food is ok but the service is lacking !
	both dishes prepared with quality veggies .	both dishes tasted old & old .
	the waitresses are friendly and helpful .	the employees are rude and condescending .
	i will be going back !	i wo n't be back !
	great food !	horrible food !
	i was just one of those lucky enough to find this one !	i was just standing there to get to get me to leave !
	this is the best chinese we have found in this area .	this is the worst hospital i have ever been in awhile .
	excellent service , good food , and nice atmosphere .	bad service , bad food , not fresh either .
Negative ⇒ Positive	the women who work here , hate their lives .	the people who work here are pretty decent too .
	i would not recommend this store for any purpose .	i would recommend this place for any pet needs .
	food was cold and not warm at all .	food was delicious and very good as well .
	i wish i could give no stars .	i love my hair , thank you .
	the pool is small and nothing special .	the office is clean and very comfortable .
	i would never stay here again .	i would recommend them to anyone .
	over all the food is crap .	all of the food is wonderful .
	check in was a disaster .	check it out for me .
	place is pretty terrible .	place is pretty good .
	worst hotel experience ever .	best mexican restaurant ever .
	this casino is very sketchy and i would not let anybody come here .	this location is very cute and i would recommend this place to anyone .
	the carne asada has almost no flavor .	the patio itself has had excellent flavor .

Table 2. Sample sentences from the testing set transferred by our model.

but in doing so, enables the latent codes z to learn a better and more complete representation of the source sentence, which can then be used by the decoder to construct more consistent and fluent transferred sentences.

In the future, we plan to explore applying this architecture to semi-supervised training problems. Our model is particularly well-suited to this application, in contrast to ARAE, because it does not leverage adversarial training for the encoder against the classifier, and thus has the potential to produce a more accurate classifier alongside a latent code representation of style and other semantic features.

8. Conclusion

This paper presented an implicitly disentangled latent variable generative model for text, and showed that failure to explicitly disentangle the label (sentiment) and style attributes of the text through adversarial training of the encoder led to improved transfer performance, but much poorer sentence fluency and consistency. These results suggest that the adversarial training of the encoder not only explicitly removes sentiment information from the latent code z, but also contributes to an improved latent code representing the stylistic and contextual features of the source sentence. Future work includes applying this model architecture to semi-supervised training problems and comparing its performance with the ARAE model, as well as other semi-supervised models.

References

Cui, Yan. An evaluation of yelp dataset. *arXiv preprint arXiv:1512.06915*, 2015.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. Curran Associates, Inc., 2014.

Heafield, Kenneth. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pp. 187–197. Association for Computational Linguistics, 2011.

Joulin, Armand, Grave, Edouard, Bojanowski, Piotr, and Mikolov, Tomas. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

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

Makhzani, Alireza, Shlens, Jonathon, Jaitly, Navdeep, and Goodfellow, Ian J. Adversarial autoencoders. *CoRR*, abs/1511.05644, 2015.

Shen, Tianxiao, Lei, Tao, Barzilay, Regina, and Jaakkola, Tommi. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pp. 6833–6844, 2017.

Zhao, Junbo, Kim, Yoon, Zhang, Kelly, Rush, Alexander M, LeCun, Yann, et al. Adversarially regularized autoencoders for generating discrete structures. *arXiv preprint arXiv:1706.04223*, 2017.