Text Style Transfer

Emrullah ERGUN

Advisor: Sylvain REYNAL, Xuan-Son NGUYEN

November 2020

Abstract

In this document we are going to study a method to be able to change the style of a sentence that a chatbot can respond to its user so that it can be customized. This will improve the user experience and make them easier to use. We would like to thank Dr Sylvain REYNAL for his support throughout our project. Generic generation and manipulation of text is challenging and has limited success compared to recent deep generative modeling in visual domain. This paper aims at generating plausible text sentences, whose attributes are controlled by learning disentangled latent representations with designated semantics. We propose a new neural generative model which combines variational auto-encoders (VAEs) and holistic attribute discriminators for effective imposition of semantic structures. The model can alternatively be seen as enhancing VAEs with the wake-sleep algorithm for leveraging fake samples as extra training data. With differentiable approximation to discrete text samples, explicit constraints on independent attribute controls, and efficient collaborative learning of generator and discriminators, our model learns interpretable representations from even only word annotations, and produces sentences with desired attributes of sentiment and tenses. Quantitative experiments using trained classifiers as evaluators validate the accuracy of short sentence and attribute generation.

1 Introduction

The stylistic properties of text have intrigued linguistic researchers for a long time. Enkvist opined that text style is a “concept that is as common as it is elusive” and suggested that style may be described as a linguistic variation while preserving the conceptual content of the text. To give a practical example, the formality of text will vary across settings for similar content; examples include a conversation with friends such as “let’s hang out on Sunday afternoon!”, or a professional email such as “We will arrange a meeting on Sunday afternoon.”

Text style transfer, which aims at modifying an entry with the desired style while preserving The content, which is not very relevant in terms of style, has received increasing attention in recent years. It has been successfully applied to the subtitling of stylized images (Gan et al, 2017), the generation of personalized conversational responses (Zhang et al., 2018a), formalized writing (Rao and Tetreault, 2018), the transfer from offensive to non-offensive language (dos Santos et al, 2018), and other stylized text generation tasks (Zhang et al., 2012). Text style transfer has been explored as a sequential learning task using parallelism (Jhamtani et al., 2017). However, parallel data sets are often not available, and manual annotation of sentences in different styles is costly. The recent wave of deep generating models has stimulated progress in transferring text styles without parallel data by learning disentanglement (Prabhumoye et al. 2018). These methods generally require massive quantities data (Subramanian et al., 2018), and may perform poorly in limited data scenarios. A natural solution to the problem of data scarcity is to use massive data from other fields. However, the direct exploitation of abundant data from other domains is problematic due to discrepancies in the distribution of data across domains. The different domains generally manifest themselves in the lexicon specific to the field. For example, adjectives for feelings such as "delicious", "tasty" and "disgusting" in restaurant reviews could have come out of place in film reviews, where the feeling of being words such as "imaginative", "hilarious", and "hilarious". "dramatic" are more typical. Changing domains is therefore likely to lead to misalignment of characteristics.

Text style transfer (TST) is a relatively new research area. Many of the earlier TST works are heavily influenced by two related research areas: neural style transfer, i.e., transferring styles in images and neural machine translation. We found that a substantial number of TST techniques were adapted from the common methods used in neural style transfer and neural machine translation. In addition, some of the evaluation metrics used in TST are also “inherited” from the neural machine translation task.

2 Application

The research on TST algorithms has many industrial applications and could lead to many commercial benefits. In this section, we summarize these applications and present some potential usages.

2.1 Writing tools

One of the industrial applications of TST algorithms is the design of writing tools. Academics across various domains have widely researched Computer-aided writing tools, and the industry has developed many writing tool applications. The TST methods can be applied as new useful features in existing writing tool applications. The utility of writing style has been widely studied by linguistic and literacy education scholars. The TST algorithms enable writing tool applications to apply the insights from existing linguistics studies to improve the writings of users. For instance, applying TST algorithms enable writing tool users to switch between writing styles for different audiences while preserving the content in their writing. The style evaluation methods developed to evaluate TST algorithms can also be applied to analyze the writing style of users. For instance, the writing tool could analyze the style of a user's business email draft to be too informal and recommend the users to modify his or her writing to make the writing style more formal.

2.2 Persuasion and Marketing

Studies have explored utilizing persuasive text to influence the attitude or behaviors of people, and the insights gained from these studies have also been applied in improve marketing and advertising in the industry. The style of text has an impact on its persuasiveness, and the TST algorithms can be used to convert a text into a more persuasive style. Recent studies have also explored personalizing persuasive strategies according to the user's profile. Similarly, TST algorithms could also be used to structure the text in different persuasive text styles that best appeal to the user profiles. For instance, TST algorithms can be applied to modify a marketing message into an authoritative style for users who appeal to authority.

2.3 Chatbot Dialogue

The research and development of chatbots, i.e., intelligent dialogue systems that are able to engage in conversations with humans, has been one of the longest-running goals in artificial intelligence. Kim et al. conducted a study on the impact of chatbot's conversational style on users and found that when a causal conversational style is used, experiment participants are less likely to persuade a user to perform an action compared to participants who conversed with formal conversational style chatbot. The encouraging results from the study suggest that a user may be influenced by chatbot's conversational styles, and TST algorithms could be exploited to enhance the chatbots' flexibility in conversational styles. TST algorithms can be applied to equip chatbots with the ability to switch between conversational styles, and this makes the chatbots more appealing and engaging to the users. For instance, a chatbot recommending products to customers may adopt a more persuasive conversational style while the same chatbot may switch to a formal conversational style when addressing the customer's complaint.

3 State of the art

3.1 A taxonomy of Text Style Transfer methods

In this section, we first propose a taxonomy to organize the most notable and promising advances in TST research in recent years. Subsequently, we discuss 3 of TST promising for our application models in greater detail.

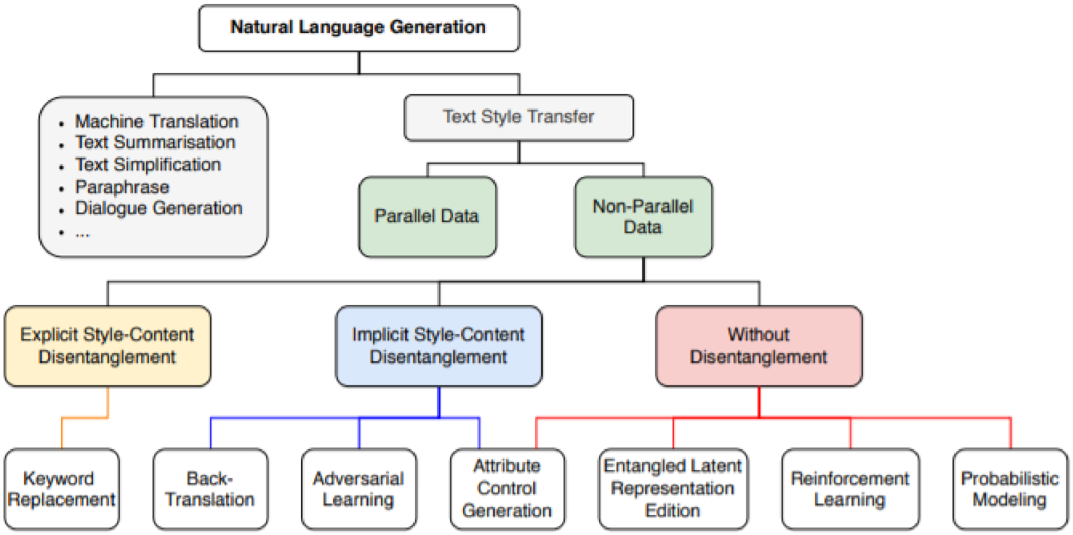


Figure 1: Taxonomy of TST

3.2 Categories of Text Style Transfer Models

To provide a bird-eye view of this field, we classify the existing TST models based on the types of (1) data setting, (2) strategy, and (3) technique used. Fig. 1 summarizes the taxonomy for text style transfer.

3.2.1 Parallel Data.

In this data setting, the TST models are trained with known pairs of text with different styles. Commonly, NMT methods such as sequence-to-sequence (Seq2Seq) models are applied to transfer the style of text. For example, Jhamtani et al. trained a Seq2Seq model with a pointer network on a parallel corpus and applied the model to translate modern English phrases to Shakespearean English.

3.2.2 Non Parallel Data

TST models in the non-parallel data setting aim to transfer the style of text without any knowledge of matching text pairs in different styles. Most of the existing TST studies fall into this category as parallel data sets are scarce in many real-world TST applications.

3.2.3 Explicit Style-Content Disentanglement

In this strategy, the TST models adopted an explicit text replacement approach to generate text of a target style. For instance, Li et al. first explicitly identify parts of the text that is associated with the original style and then replace them with new phrases associated with the target style. The text with the replaced new phrases is then inputted into a Seq2Seq model to generate a fluent text in the target style.

3.2.4 Implicit Style-Content Disentanglement

To disentangle style and content in text implicitly, TST models aim first to learn the content and style latent representations of a given text. Subsequently, the original text's content latent representation is combined with the latent representation of the target style to generate a new text in the target style. Multiple techniques such as back-translation, adversarial learning, and controllable generation have been proposed to disentangle the content and style latent representations.

3.2.5 Without Style-Content Disentanglement

Recent studies have suggested that it is difficult to judge the quality of text style and content disentanglement and the disentanglement is also unnecessary for TST. Therefore, newer TST studies explored performing TST without disentangling the text's style and content. Techniques such as adversarial learning, controllable generation, reinforcement learning, probabilistic modeling, and pseudo-parallel corpus have been applied to perform TST without disentanglement of the text's content and style.

3.2.6 Sequence-to-Sequence Model with Parallel Data

The Sequence-to-Sequence (Seq2Seq) model is core to many natural language generation tasks, and TST is no exception. Generally, a Seq2Seq model is trained on a parallel corpus, where the text of the original style is input into an encoder, and the decoder outputs the corresponding text of the target style. Variants of the general approach were proposed in TST models that trained on parallel datasets. Jhamtani et al. extended the work in Xu et al. by adding a pointer network to the Seq2Seq model to selectively copy word tokens from the input text directly to generate the text in a target style. Carlson added attention mechanism to the Seq2Seq model to evaluate their proposed parallel Bible prose style corpus. Other studies have also attempted to use the parallel dataset to train a seq2seq a semi-supervised fashion, as well as fine-tuning pre-trained models to perform TST. Another interesting approach is to generate pseudo-parallel datasets and apply seq2seq models to perform TST. Jin et al. first constructed a pseudo-parallel corpus by matching text sentences in a source style corpus X with text sentences in target style corpus Y using cosine similarity. Subsequently, a seq2seq model is trained using the constructed pseudo-parallel corpus to perform TST. Nikolov and Hahnloser improve the pseudo-parallel corpus generation with a hierarchical method that computes similarity scores at document and sentence levels to find parallel text pairs across different style corpus.

Liao et al. first generated a pseudo-parallel dataset and then applied a dual-encoder seq2seq framework to disentangle the content from style for text style transfer. Zhang et al. proposed and experimented with a few pseudo-parallel dataset generation methods. Specifically, the researchers explored simultaneous training, pre-training, and fine-tuning data augmentation methods to generate pseudo-parallel data for TST tasks. A major drawback of conventional Seq2Seq models is that training requires large parallel corpora, which are scarce in the TST domain. Most of the TST studies have moved on and experimented with performing TST without parallel datasets.

3.2.7 Reinforcement Learning

Reinforcement learning has also been applied to perform TST. For instance, Luo et al. proposed to learn two seq2seq models between two styles via reinforcement learning, without disentangling style and content. Fig.2 illustrates the proposed dual reinforcement learning framework. The authors considered the learning of source-to-target style and target-to-source style as a dual-task. The style classifier reward, Rs and reconstruction reward, Rc , are designed to encourage style transfer accuracy and content preservation. The overall reward is the harmonic mean of the two rewards, and it is used as the feedback signal to guide learning in the dual-task structure. As such, the model can be trained via reinforcement learning without any use of parallel data or content-style disentanglement Gong et al. proposed a reinforcement learning-based generator-evaluator framework to perform TST. Similar to previous TST works, the proposed model employs an attention-based encoder-decoder mode to transfer and generate target style sentences. However, unlike the previous models that utilize a style classifier to guide the generation process, the proposed model employed a style classifier, semantic model, and a language model to provide style, semantic, and fluency rewards respectively to guide the text generation. The authors' intuition is that the transfer of text style should not only ensure the transfer of style and content preservation but also generate fluent sentences.

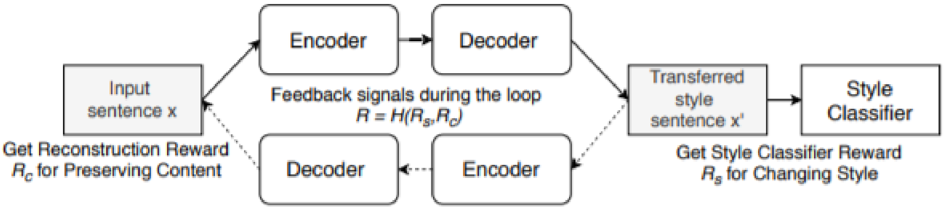

Encoder
Decoder
Input sentence x
Feedback signals during the loop
$R = H(R_s, R_c)$
Transferred style sentence x'
Style Classifier
Get Reconstruction Reward
R_c for Preserving Content
Decoder
Encoder
Get Style Classifier Reward
R_s for Changing Style

Figure 2: reinforcment learning schema

3.2.8 Attribute Control Generation

Unlike an autoencoder, which learns a compressed representation for an input data, the Variational Autoencoder (VAE) learns the parameters of a probability distribution representing the data. The learned distribution can also be sampled to generate new data samples. Therefore, the generative nature of VAE makes it widely explored and utilized in many natural language generation tasks. Hu et al. proposed a TST model that utilized VAE to learn a sentence's latent representation z and leverage a style classifier to learn a style attribute vector s The probabilistic encoder of VAE also functions as an additional discriminator to capture variations of implicitly modeled aspects, and guide the generator to avoid entanglement during attribute code manipulation. Finally, z and s are input into a decoder to generate a sentence in the specific style

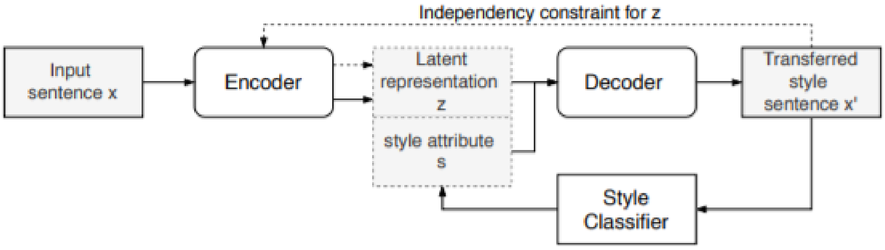


Figure 3: attribute control gen

DAST-C An attribute-controlled TST model that performs TST in a domain-aware manner. Two variants are proposed: The Domain Adaptation Style (DAST) model and DAST with generic content information (DAST-C). In these models, latent style attributes and domain vectors are learned to perform TST across domains.

Control Gen algorithm An attribute-controlled TST model that utilized variational auto-encoders and style classifier to guide the learning of a style attribute to control the generation of text in different styles.

4 Method

We Choose the Control Gen algorithm to perform our task as it was easiest project to reproduce.

4.1 Introduction

In this report we tackle the problem of controlled generation of text. That is, we focus on generating realistic sentences, whose attributes can be controlled by learning disentangled latent representations. To enable the manipulation of generated sentences, a few challenges need to be addressed. A first challenge comes from the discrete nature of text samples. The resulting non-differentiability hinders the use of global discriminators that assess generated samples and back-propagate gradients to guide the optimization of generators in a holistic manner, as shown to be highly effective in continuous image generation and representation modeling. A number of recent approaches attempt to address the non-differentiability through policy learning which tends to suffer from high variance during training, or continuous approximations where only preliminary qualitative results are presented. As an alternative to the discriminator based learning, semi-supervised VAEs minimize element-wise reconstruction error on observed examples and are applicable to discrete visibles. This, however, loses the holistic view of full sentences and can be inferior especially for modeling global abstract attributes (e.g., sentiment). Another challenge for controllable generation relates to learning disentangled latent representations. Interpretability expects each part of the latent representation to govern and only focus on one aspect of the samples. Prior methods on structured representation learning lack explicit enforcement of the in dependence property on the full latent representation, and varying individual code may result in unexpected variation of other unspecified attributes besides the desired one.

We base our generator on VAEs in combination with holistic discriminators of attributes for effective imposition of structures on the latent code. End-to-end optimization is enabled with differentiable softmax approximation which anneals smoothly to discrete case and helps fast convergence. The probabilistic encoder of VAE also functions as an additional discriminator to capture variations of implicitly modeled aspects, and guide the generator to avoid entanglement during attribute code manipulation. Our model can be interpreted as enhancing VAEs with an extended wake-sleep procedure, where the sleep phase enables incorporation of generated samples for learning both the generator and discriminators in an alternating manner. The generator and the discriminators effectively provide feedback signals to each other, resulting in an efficient mutual bootstrapping framework. We show a little supervision (e.g., 100s of annotated sentences) is sufficient to learn structured representations. Besides efficient representation learning and enabled semisupervised training, another advantage of using discriminators as learning signals for the generator, as compared to conventional conditional reconstruction based methods, is that discriminators of different attributes can be trained independently. That is, for each attribute one can use separate labeled data for training the respective discriminator, and the trained discriminators can be combined arbitrarily to control a set of attributes of interest. In contrast, reconstruction based approaches typically require every instance of the training data to be labeled exhaustively with all target attributes, or to marginalize out any missing attributes which can be computationally expensive.

4.2 Variational AutoEncoder

The variational autoencoder is a generative model that is based on a regularized version of the standard autoencoder. This model imposes a prior distribution on the hidden codes $\overrightarrow{z}$ which enforces a regular geometry over codes and makes it possible to draw proper samples from the model using ancestral sampling. The VAE modifies the autoencoder architecture by replacing the deterministic function ϕ enc with a learned posterior recognition model, $q(\overrightarrow{z}|x)$. This model parametrizes an approximate posterior distribution over $\overrightarrow{z}$ (usually a diagonal Gaussian) with a neural network conditioned on x. Intuitively, the VAE learns codes not as single points, but as soft ellipsoidal regions in latent space, forcing the codes to fill the space rather than memorizing the training data as isolated codes. If the VAE were trained with a standard autoencoder's reconstruction objective, it would learn to encode its inputs deterministically by making the variances in $q(\overrightarrow{z}|x)$ vanishingly small . Instead, the VAE uses an objective which encourages the model to keep its posterior distributions close to a prior $p(\overrightarrow{z}), generallyastandardGaussian(\mu = \overrightarrow{0}, \sigma = \overrightarrow{1})$. Additionally, this objective is a valid lower bound on the true log likelihood of the data, making the VAE a generative model. This objective takes the following form:

4.3 Model and architecture

We build our framework starting from variational autoencoders (§2) which have been used for text generation (Bowman et al., 2015), where sentence $\hat{x}$ is generated conditioned on latent code z. The vanilla VAE employs an unstructured vector z in which the dimensions are entangled. To model and control the attributes of interest in an interpretable way, we augment the unstructured variables z with a set of structured variables c each of which targets a salient and independent semantic feature of sentences. We want our sentence generator to condition on the combined vector (z, c), and generate samples that fulfill the attributes as specified in the structured code c. Conditional generation in the context of VAEs (e.g., semi-supervised VAEs is often learned by reconstructing observed examples given their feature code. However, as demonstrated in visual domain, compared to computing element-wise distances in the data space, computing distances in the feature space allows invariance to distracting transformations and provides a better, holistic metric. Thus, for each attribute code in c, we set up an individual discriminator to measure how well the generated samples match the desired attributes, and drive the generator to produce improved results. The difficulty of applying discriminators in our context is that text samples are discrete and non-differentiable, which breaks down gradient propagation from the discriminators to the generator. We use a continuous approximation based on softmax with a decreasing temperature, which anneals to the discrete case as training proceeds. This simple yet effective approach enjoys low variance and fast convergence. Intuitively, having an interpretable representation would imply that each structured code in c can independently control its target feature, without entangling with other attributes, especially those not

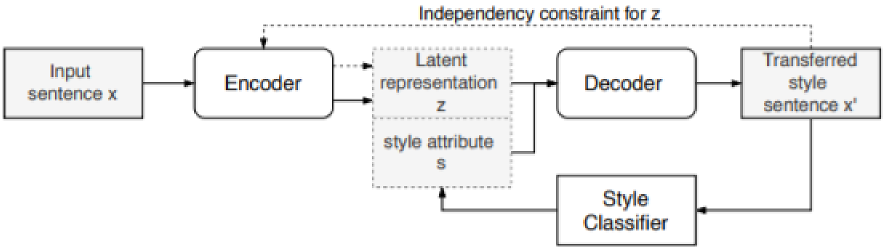
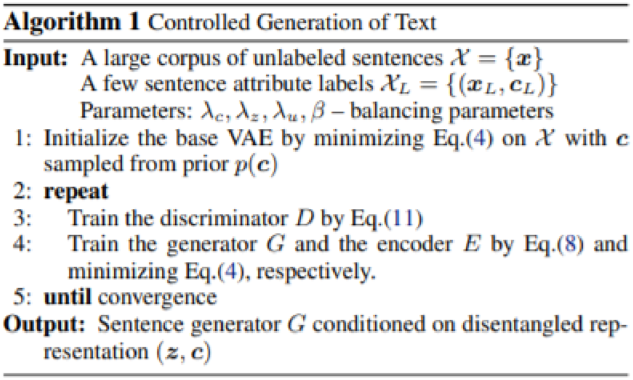


Figure 4: The generative model, where z is unstructured latent code and c is structured code targeting sentence attributes to control. Blue dashed arrows denote the proposed independency constraint, and red arrows denote gradient propagation enabled by the differentiable approximation.

explicitly modeled. We encourage the independency by enforcing those irrelevant attributes to be completely captured in the unstructured code z and thus be separated from c that we will manipulate. To this end, we reuse the VAE encoder as an additional discriminator for recognizing the attributes modeled in z, and train the generator so that these unstructured attributes can be recovered from the generated samples. As a result, varying different attribute codes will keep the unstructured attributes invariant as long as z is unchanged. Figure 4 shows the overall model structure. Our complete model incorporates VAEs and attribute discriminators, in which the VAE component trains the generator to reconstruct real sentences for generating plausible text, while the discriminators enforce the generator to produce attributes coherent with the conditioned code. The attribute discriminators are learned to fit labeled examples to entail designated semantics, as well as trained to explain samples from the generator. That is, the generator and the discriminators form a pair of collaborative learners and provide feedback signals to each other.



Algorithm 1 Controlled Generation of Text

```
Input: A large corpus of unlabeled sentences $\mathcal{X} = \{\boldsymbol{x}\}$
       A few sentence attribute labels $\mathcal{X}_L = \{(\boldsymbol{x}_L, \boldsymbol{c}_L)\}$
       Parameters: $\lambda_c, \lambda_z, \lambda_u, \beta$ – balancing parameters
Initialize the base VAE by minimizing Eq.(4) on $\mathcal{X}$ with $\boldsymbol{c}$
  sampled from prior $p(\boldsymbol{c})$
repeat
    Train the discriminator $D$ by Eq.(11)
    Train the generator $G$ and the encoder $E$ by Eq.(8) and
    minimizing Eq.(4), respectively.
until convergence
Output: Sentence generator $G$ conditioned on disentangled representation $(\boldsymbol{z}, \boldsymbol{c})$
```

4.4 Summarization

We have derived our model and its learning procedure. The generator is first initialized by training the base VAE on a large corpus of unlabeled sentences, through the objective of minimizing Eq.(4) with the latent code c at this time sampled from the prior distribution $p(\mathrm{c})$. The full model is then trained by alternating the optimization of the generator and the discriminator, as summarized in Algorithm 1. Our model can be viewed as combining the VAE framework with an extended wake-sleep method. Specifically, in Eq.(10), samples are produced by the generator and used as targets for maximum likelihood training of the discriminator. This resembles the sleep phase of wake-sleep. Eqs.(6)-(7) further leverage the generated samples to improve the generator. We can see the above together as an extended sleep procedure based on "dream" samples obtained by ancestral sampling from the generative network. On the other hand, Eq.(4) samples c from the discriminator distribution $q_D(\mathrm{c}|x)$ on observation x, to form a target for training the generator, which corresponds to the wake phase. The effective combination enables discrete latent code, holistic discriminator metrics, and efficient mutual bootstrapping. Training of the discriminators need supervised data to impose designated semantics. Discriminators for different attributes can be trained independently on separate labeled sets. That is, the model does not require a sentence to be annotated with all attributes, but instead needs only independent labeled data for each individual attribute. Moreover, as the labeled data are used only for learning attribute semantics instead of direct sentence generation, we are allowed to extend the data scope beyond labeled sentences to, e.g., labeled words or phrases. As shown in the experiments (section 4), our method is able to effectively lift the word level knowledge to sentence level and generate convincing sentences. Finally, with the augmented unsupervised training in the sleep phrase, we show a little supervision is sufficient for learning structured representations.

5 Model Structure

We now describe our model in detail, by presenting the learning of generator and discriminators, respectively.

5.1 Generator Learning

The generator G is an LSTM-RNN for generating token sequence $\hat{x} = \hat{x}_1, ..., \hat{x}_T$ conditioned on the latent code (z, c), which depicts a generative distribution:

$$\hat{\boldsymbol{x}} \sim G(\boldsymbol{z}, \boldsymbol{c}) = p_G(\hat{\boldsymbol{x}}|\boldsymbol{z}, \boldsymbol{c}) \ = \prod\nolimits_t p(\hat{x}_t|\hat{\boldsymbol{x}}^{<t}, \boldsymbol{z}, \boldsymbol{c}), \qquad (1)$$

where $\hat{x} < t$ indicates the tokens preceding $\hat{x}t$. The generation thus involves a sequence of discrete decision making which samples a token from a multinomial distribution parametrized using softmax function at each time step t:

$$\hat{x}_t \sim \text{softmax}(\boldsymbol{o}_t/\tau), \tag{2}$$

where o_t is the logit vector as the inputs to the softmax function, and $\tau > 0$ is the temperature normally set to 1. The unstructured part z of the representation is modeled as continuous variables with standard Gaussian prior $p(z)$, while the structured code c can contain both continuous and discrete variables to encode different attributes (e.g., sentiment categories, formality) with appropriate prior p(c). Given observation x, the base VAE includes a conditional probabilistic encoder E to infer the latents z:

$$\boldsymbol{z} \sim E(\boldsymbol{x}) = q_E(\boldsymbol{z}|\boldsymbol{x}). \tag{3}$$

Let θG and θE denote the parameters of the generator G and the encoder E, respectively. The VAE is then optimized to minimize the reconstruction error of observed real sentences, and at the same time regularize the encoder to be close to the prior $p(z)$:

$$\mathcal{L}_{\text{VAE}}(\boldsymbol{\theta}_G, \boldsymbol{\theta}_E; \boldsymbol{x}) = \text{KL}(q_E(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z})) - \mathbb{E}_{q_E(\boldsymbol{z}|\boldsymbol{x})q_D(\boldsymbol{c}|\boldsymbol{x})}\left[\log p_G(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{c})\right], \tag{4}$$

where $KL(..||..)$ is the KL-divergence; and $q_D(c|x$ is) the conditional distribution defined by the discriminator D for each structured variable in c: Here,

$$D(\boldsymbol{x}) = q_D(\boldsymbol{c}|\boldsymbol{x}). \tag{5}$$

for notational simplicity, we assume only one structured variable and thus one discriminator, though our model specification can straightforwardly be applied to many attributes. The distribution over (z, c) factors into q_E and q_D as we are learning disentangled representations. Note that here the discriminator D and code c are not learned with the VAE loss, but instead optimized with the objectives described shortly. Besides the reconstruction loss which drives the generator to produce realistic sentences, the discriminator provides extra learning signals which enforce the generator to produce coherent attribute that matches the structured code in c. However, as it is impossible to propagate gradients from the discriminator through the discrete samples, we resort to a deterministic continuous approximation. The approximation replaces the sampled token x^t (represented as a one-hot vector) at each step with the probability vector in Eq.(2) which is differentiable w.r.t the generator's parameters. The probability vector is used as the output at the current step and the input to the next step along the sequence of decision making. The resulting "soft" generated sentence, denoted as $\tilde{G}_\tau(z, c)$, is fed into the discriminator1 to measure the fitness to the target attribute, leading to the following loss for improving G: The

$$\mathcal{L}_{\text{Attr},c}(\boldsymbol{\theta}_G) = -\mathbb{E}_{p(\boldsymbol{z})p(\boldsymbol{c})}\left[\log q_D(\boldsymbol{c}|\widetilde{G}_\tau(\boldsymbol{z}, \boldsymbol{c}))\right]. \tag{6}$$

temperatureτ (Eq.2) is set to $\tau \to 0$ as training proceeds, yielding increasingly peaked distributions that finally emulate discrete case. The simple deterministic approximation effectively leads to reduced variance and fast convergence during training, which enables efficient learning of the conditional generator. The diversity of generation results is guaranteed since we use the approximation only for attribute modeling and the base sentence generation is learned through VAEs. With the objective in Eq.(6), each structured attribute of generated sentences is controlled through the corresponding code in c and is independent with other variables in the latent representation. However, it is still possible that other attributes not explicitly modeled may also entangle with the code in c, and thus varying a dimension of c can yield unexpected variation of these attributes we are not interested in. To address this, we introduce the independency constraint which separates these attributes with c by enforcing them to be fully captured by the unstructured part z. Therefore, besides the attributes explicitly encoded in c, we also train the generator so that other non-explicit attributes can be correctly recognized from the generated samples and match the unstructured code z. Instead of building a new discriminator, we reuse the variational encoder E which serves precisely to infer the latents z in the base VAE. The loss is in the same form as with Eq.(6) except replacing the discriminator conditional q_D with the encoder conditional q_E: Note that, as the discriminator in Eq.(6),

$$\mathcal{L}_{\text{Attr},z}(\boldsymbol{\theta}_G) = -\mathbb{E}_{p(\boldsymbol{z})p(\boldsymbol{c})}\left[\log q_E(\boldsymbol{z}|\widetilde{G}_\tau(\boldsymbol{z},\boldsymbol{c}))\right]. \quad (7)$$

the encoder now performs inference over generated samples from the prior, as opposed to observed examples as in VAEs. Combining Eqs.(4)-(7) we obtain the generator objective: where λ_c and λ_z are balancing parameters. The variational

$$\min_{\boldsymbol{\theta}_G} \mathcal{L}_G = \mathcal{L}_{\text{VAE}} + \lambda_c \mathcal{L}_{\text{Attr},c} + \lambda_z \mathcal{L}_{\text{Attr},z}, \quad (8)$$

encoder is trained by minimizing the VAE loss, i.e., $min_{\theta_E} \mathcal{L}_{VAE}$.

5.2 Discriminator Learning

The discriminator D is trained to accurately infer the sentence attribute and evaluate the error of recovering the desired feature as specified in the latent code. For instance, for categorical attribute, the discriminator can be formulated as a sentence classifier; while for continuous target a probabilistic regressor can be used. The discriminator is learned in a different way compared to the VAE encoder, since the target attributes can be discrete which are not supported in the VAE framework. Moreover, in contrast to the unstructured code z which is learned in an unsupervised manner, the structured variable c uses labeled examples to entail designated semantics. We derive an efficient semisupervised learning method for the discriminator. Formally, let θ_D denote the parameters of the discriminator. To learn specified semantic meaning, we use a set of labeled examples $\mathcal{X}_L = (x_L, c_L)$to train the discriminator D with the following objective: Besides, the conditional generator G is also capable of synthesizing

$$\mathcal{L}_s(\boldsymbol{\theta}_D) = -\mathbb{E}_{\mathcal{X}_L}\left[\log q_D(\boldsymbol{c}_L|\boldsymbol{x}_L)\right]. \qquad (9)$$

(noisy) sentence-attribute pairs $(\hat{x}, c)$ which can be used to augment training data for semi-supervised learning. To alleviate the issue of noisy data and ensure robustness of model optimization, we incorporate a minimum entropy regularization term. The resulting objective is thus:

$$\mathcal{L}_u(\boldsymbol{\theta}_D) = -\mathbb{E}_{p_G(\hat{\boldsymbol{x}}|\boldsymbol{z},\boldsymbol{c})p(\boldsymbol{z})p(\boldsymbol{c})}\big[\log q_D(\boldsymbol{c}|\hat{\boldsymbol{x}}) + \beta\mathcal{H}(q_D(\boldsymbol{c}'|\hat{\boldsymbol{x}}))\big], \qquad (10)$$

where $\mathcal{H}(q_D(c'|\hat{x}))$ is the empirical Shannon entropy of distribution q_D evaluated on the generated sentence $\hat{x}$; and β is the balancing parameter. Intuitively, the minimum entropy regularization encourages the model to have high confidence in predicting labels. The joint training objective of the discriminator using both labeled examples and synthesized samples is then given as:

$$\min_{\boldsymbol{\theta}_D} \mathcal{L}_D = \mathcal{L}_s + \lambda_u\mathcal{L}_u, \qquad (11)$$

where λ_u is the balancing parameter.

6 Experiences

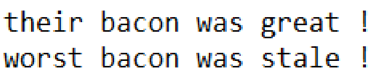
6.1 Parameter Setting

The generator and encoder are set as single-layer LSTM RNNs with input/hidden dimension of 300 and max sample length of 15. Discriminators are set as ConvNets. Detailed configurations are in the supplements. To avoid vanishingly small KL term in the VAE module (Eq.4) (Bowman et al., 2015), we use a KL term weight linearly annealing from 0 to 1 during training. Balancing parameters are set to $\lambda_c = \lambda_z = \lambda_u = 0.1$, and β is selected on the dev sets. At test time sentences are generated with Eq.(1).

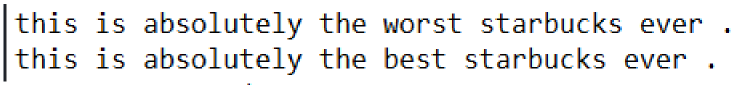
7 Results

We trained our model onto the YELP sentiment Data set. Our goal is to pass from positive sentence to a negative one.

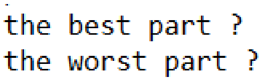
We notice after studying the results that a certain category of sentences are successfully transformed while another category ends in failure. Sentences where an adjective is used to qualify the quality of a restorer/object/person are very well handled because the model will modify the adjective in question to transform the style.



```
their bacon was great !
worst bacon was stale !
```

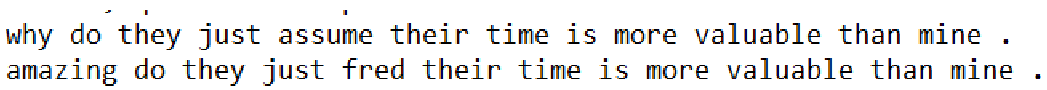


```
this is absolutely the worst starbucks ever .
this is absolutely the best starbucks ever .
```

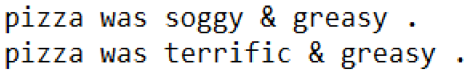


```
the best part ?
the worst part ?
```

The model also fails to use the right theme in its vocabulary and ends up producing sentences where a pizza is said to be ’terrific’.



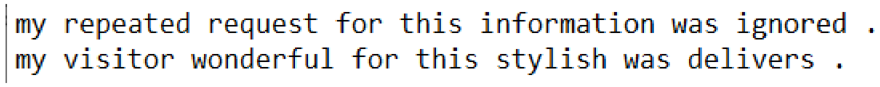
```
why do they just assume their time is more valuable than mine .
amazing do they just fred their time is more valuable than mine .
```



```
pizza was soggy & greasy .
pizza was terrific & greasy .
```

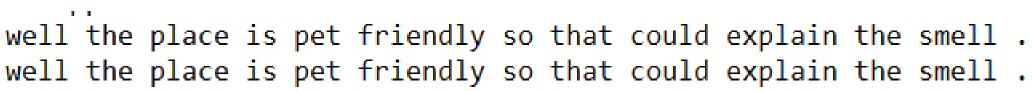
The DAST algorithm presented at the beginning of our report proposes an approach to correct this problem nevertheless.

Neutral structures are also badly managed and lead to a failure because the model will look for the source attributes that we specified in the c attribute labeling for a part of our data in the unlabeled data.



```
my repeated request for this information was ignored .
my visitor wonderful for this stylish was delivers .
```

Moreover it seems that it does not modify anything when it detects that the sentence already has the desired style attribute.



```
well the place is pet friendly so that could explain the smell .
well the place is pet friendly so that could explain the smell .
```

8 Conclusion

The reproduction of Zhiting Hu et al’s Model allowed us to understand the springs and stakes of Text Style Transfer. This allowed us to broaden my knowledge in this field and gave me a real experience in implementation because I was exposed to a lot of problems, especially those related to versioning. My work on the state of the art in the domain was beneficial to me in getting the main principles out of it and proved to me that there was still a lot of work to be done before achieving a perfect transformation.

Their approach combines VAEs with attribute discriminators and imposes explicit independency constraints on attribute controls, enabling disentangled latent code. Semi-supervised learning within the joint VAE/wake-sleep framework is effective with little or incomplete supervision. Hu et al. (2017) develop a unified view of a diverse set of deep generative paradigms, including GANs, VAEs, and wake-sleep algorithm. Their model can be alternatively motivated under the view as enhancing VAEs with the extended sleep phase and by leveraging generated samples. Interpretability of the latent representations not only allows dynamic control of generated attributes, but also provides an interface that connects the end-to-end neural model with conventional structured methods. For instance, we can encode structured constraints (e.g., logic rules or probabilistic structured models) on the interpretable latent code, to incorporate prior knowledge or human intentions ; or plug the disentangled generation model into dialog systems to generate natural language responses from structured dialog states. Though we have focused on the generation capacity of our model, the proposed collaborative semi-supervised learning framework also helps improve the discriminators by generating labeled samples for data augmentation (e.g., see Figure 4). More generally, for any discriminative task, we can build a conditional generative model to synthesize additional labeled data. The accurate attribute generation of their approach can offer larger performance gains compared to previous generative methods.

9 Références

Nils Erik Enkvist. 2016. Linguistic stylistics. Vol. 5. Walter de Gruyter GmbH Co KG. Cicero dos Santos, Igor Melnyk, and Inkit Padhi. 2018. Fight-

ing Offensive Language on Social Media with Unsupervised Text Style Transfer. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). 189–194. Harsh Jhamtani,

Varun Gangal, Eduard Hovy, and Eric Nyberg. 2017. Shakespearizing Modern Language Using Copy-Enriched Sequence to Sequence Models. In Proceedings of the Workshop on Stylistic Variation. 10–19 Hu, Zhiting, Yang, Zichao,

Salakhutdinov, Ruslan, and Xing, Eric P. On unifying deep generative models. arXiv preprint arXiv:1706.00550, 2017. Zhirui Zhang, Shuo Ren, Shujie

Liu, Jianyong Wang, Peng Chen, Mu Li, Ming Zhou, and Enhong Chen. 2018. Style Transfer as Unsupervised Machine Translation. CoRR abs/1808.07894 (2018). arXiv:1808.07894 http://arxiv.org/abs/1808.07894 Sudha Rao and Joel

Tetreault. 2018. Dear Sir or Madam, May I Introduce the GYAFC Dataset: Corpus, Benchmarks and Metrics for Formality Style Transfer. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). 129–140 Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style Transfer Through Back-Translation. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 866–876 Yi Zhang, Tao Ge, and Xu Sun.

2020. Parallel Data Augmentation for Formality Style Transfer. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, Retrieve, Generate: a Simple Approach to Sentiment and Style Transfer. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). 1865–1874 Wei Xu, Alan Ritter, Bill Dolan, Ralph Grishman, and Colin Cherry.

2012. Paraphrasing for style. In Proceedings of COLING 2012. 2899–2914 Bowman, Samuel R, Vilnis, Luke, Vinyals, Oriol, Dai, Andrew M, Jozefowicz, Rafal, and Bengio, Samy. Generating sentences from a continuous space. arXiv preprint Zhijing Jin, Di Jin, Jonas Mueller, Nicholas Matthews, and Enrico Santus. 2019. IMaT: Unsupervised Text Attribute Transfer via Iterative Matching and Translation. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). 3088–3100. Nikola I. Nikolov and Richard H. R. Hahnloser. 2018. Large-scale Hierarchical Alignment for Author Style Transfer. CoRR abs/1810.08237 (2018). arXiv:1810.08237 http://arxiv.org/abs/1810.08237 Yi Liao, Lidong Bing, Piji Li, Shuming Shi, Wai Lam, and Tong Zhang. 2018. Quase: Sequence editing under quantifiable guidance. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 3855–3864. Yi Zhang, Tao Ge, and Xu Sun. 2020. Parallel Data Augmentation for Formality Style Transfer. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics Fuli Luo, Peng Li, Jie Zhou, Pengcheng Yang, Baobao Chang, Xu Sun, and Zhifang Sui. 2019. A dual reinforcement learning framework for unsupervised text style transfer. In Proceedings of the 28th International Joint Conference on Artificial Intelligence. AAAI Press, 5116–5122 Hongyu

Gong, Suma Bhat, Lingfei Wu, JinJun Xiong, and Wen-mei Hwu. 2019. Reinforcement Learning Based Text Style Transfer without Parallel Training Corpus. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). 3168–3180 Zhiting Hu, Zichao Yang,

Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 1587–1596 github :$https : //github.com/asyml/texar/tree/master/examples/text_style_transfer$.