Karlsruhe Institute of Technology (KIT)
Institute for Anthropomatics and Robotics (IAR)
Department of Informatics

Unsupervised Style Transfer

Valentin Rublack

Submitted as the Bachelor's thesis, 24.09.2018 - 23.01.2019

Statement of Originality

Ich versichere hiermit, dass ich die Arbeit selbstständig verfasst habe, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht habe und die Satzung des Karlsruher Instituts für Technologie zur Sicherung guter wissenschaftlicher Praxis beachtet habe.

Karlsruhe, 23. Januar 2019

Valentin Rublack

Acknowledgement

I would like to thank Dr. Niehues, Mr. Constanting and Prof. Waibel from the Interactive Systems Lab for their supervision and assistance. Furthermore, I'd like to thank DAAD (German Academic Exchange Service) for the funding of my CLICS scholarship which helped me write my thesis abroad at Carnegie Mellon University, Pittsburgh.

Abstract

The task of style transfer in the domain of text mandates changing the original style of the sentence into a target style while preserving the remaining elements of the sentence. We propose three different methods for achieving this and train them for the style positive/negative. We do this without requiring aligned data, i.e. pairs of original and rewritten sentences, thus we call our approaches unsupervised. The first approach is to use an adversarial style classifier to eliminate the style from the hidden representation. The second one is to change the hidden representation at test time guided by the gradient of a style classifier. The third one is to delete style words from the original sentence and train a model to replace them. We use the Transformer architecture for all models, a neural network architecture that can often handle sequence data more efficiently and better than Recurrent Neural Networks. An evaluation is performed with automated metrics.

Contents

Chapter 1

Introduction

In general, the term *style transfer* refers to applying the style of one data sample to the content of another data sample.

1.1 With Images

In the domain of images, there have been impressive results, most notably the transfer of the style of a particular painter to a photo as in [Gatys et al., 2015] (see Figure 1.1) or depicting a scene in various seasons or daytimes as in [Luan et al., 2017]. If the domain of style transfer is interpreted more broadly, it can include various other tasks where the goal is to change a particular attribute but preserve the rest. An example of this is [Liu et al., 2017], where attributes such as glasses are added to a portrait of a person or the breed of an animal is changed. Another example are the infamous *Deep Fakes*, where the face of person is replaced by the face of another person.

Figure 1.1: Image style transfer with content image on the left, style image in between and transferred image on the right (from [Gatys et al., 2015])

1.2 With Text

The domain of text has also received a large amount of attention within the last two years. The goal here is similar as in the image domain and possible styles include positive sentiment, political slant or gender.

Unlike with images, it is feasible to compose at least small parallel datasets that contains the source sentence and a desired result of the transfer. However, unsupervised approaches have received a larger number of publications, because composing large parallel datasets is very costly. Not having a parallel dataset also necessitates the development of unsupervised evaluation metrics for the style transfer, since a reference-approach such as BLEU isn't possible.

It's important to note that the separability of content and style, which would be required in order for style transfer to work perfectly, is never fully present in practical applications. For example, the sentence *The food was good.* contains the word *good* which is very positive although essential to the meaning, so replacing it will invariably change the meaning of the sentence as well. So the problem should rather be phrased as changing the style while preserving the topic, rather than the content.

Chapter 2

Background Theory

2.1 Transformer Neural Networks

Introduced in [Vaswani et al., 2017], Transformer Neural Network are a different approach to sequence-to-sequence models, originally motivated by machine translation. Unlike Recurrent Neural Networks (RNNs), there is no recurrence between time steps, allowing for better parallel computation. They achieve better translation results with fewer computational steps.

Similar to RNNs, the Transformer consists of an encoder which converts the text to a latent representation, and a decoder which converts it back to text (Figure 2.1). They both consist of a fixed number of layers, which consist of several sub-layers themselves. Encoder layers have two sublayers: self-attention, which attends on the previous layer, followed by a simple feed-forward layer. A residual connection is added to each of these sublayers. Decoder layers are similar, with the exception that between self-attention and feed-forward, an attention layer that attends on the encoder output is added. Furthermore, the self-attention receives a mask in the time dimension, such that it can't attend on future timesteps. All of the attention in the model is instantiated as multi-head attention (see subsection 2.1.1). The encoder receives the source sequence as the input, while the decoder receives the encoder output and the target. Note that information cannot flow from the target to previous timesteps in the decoder due to masking. Since we don't have recurrence anymore but the order of the input sequence matters,

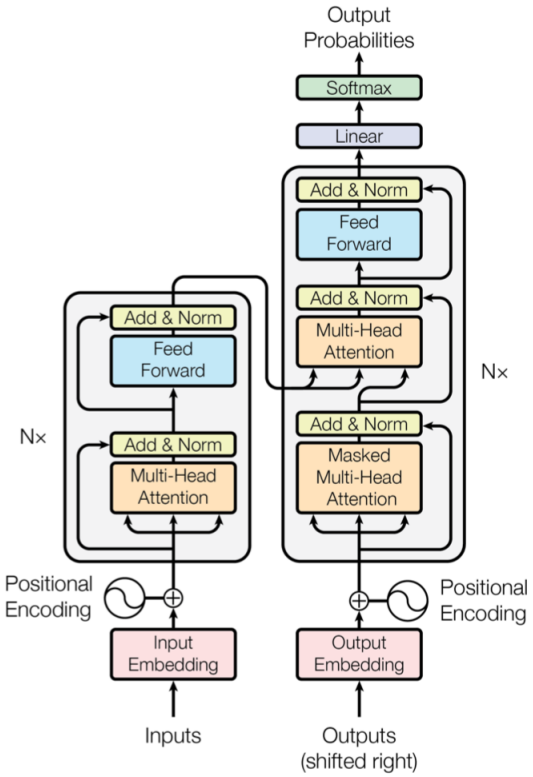


Figure 2.1: Transformer architecture (from [Vaswani et al., 2017])

it needs to receive a position-specific encoding. This is realized by sine and cosine functions with different frequencies for different dimensions, that receive the relative position of a token as the input. The result is added to the embeddings.

The output of each encoder- as well as decoder layer is of size d_{model} t, where t is the length of the sequence. The size of the feed-forward sublayers is d_{inner}.

2.1.1 Multi-head attention

Attention in general is a way of relating information from previous vectors to the current hidden state. In the context of encoder-decoder models, these previous vectors are usually hidden states of the encoder and the current hidden state is in the decoder. More specifically, if h_i is an encoder state and d_j the current decoder state, our resulting context vector c_j of the attention becomes

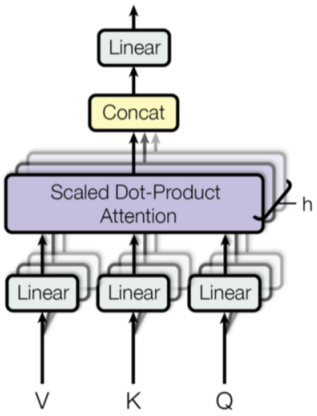


Figure 2.2: Multi-head attention with h different subspaces (from [Vaswani et al., 2017])

$$c_j = \sum_{i=1}^{M} \alpha_{i;j} h_i$$

and the attention weights $\alpha_{i;j}$ are computed as

$$\alpha_{i;j} = \frac{\exp(\text{score}(d_{j-1;h_i}))}{\sum_{l=1}^{M} \exp(\text{score}(d_{j-1}; h_l))}$$

where score, also called compatibility function, is often a simple dot-product. c_j is then usually added to d_j, the result of which is being used to compute d_{j+1}. Intuitively, this represents a weighted sum of previous hidden states, the weights of which are determined by the score-function. Frequently, the terms *query*, *key* and *value* are used, which are d_{j-1}; h_i and h_i here, respectively. Together with the softmax-function, this leads to a simpler equation:

$$c_j = \text{softmax}(\text{score}(query; key)) \quad value \tag{2.1}$$

The novelty in Transformer Networks is the so-called *multi-head attention*. It applies different linear transformations to query, key and value in parallel in order to project them into subspaces of dimension d_k. The attention is then performed on each of these subspaces as in Equation 2.1, where $\text{score}(query; key) = query\ key^T = \sqrt{d_k}$. The scaling by $\sqrt{d_k}$ is done to counteract vanishing gradients when the result of the score-function becomes too large. For this reason, this kind of attention in the subspaces is called *scaled dot-product attention*. After this is computed, the

results from each subspace are concatenated, and followed by another linear transformation, they result in the final context vector of the multi-head attention (see Figure 2.2).

Chapter 3

Related work

3.1 Style Transfer

The domain of style transfer has received much attention recently. As described earlier, the goal here is to change the style of a sentence as much as possible while leaving the rest as unchanged as possible.

3.1.1 Supervised

There are a few fully supervised approaches which require parallel data, more precisely a sentences in one style and that sentence transferred into another style, representing our target.

[**Rao and Tetreault, 2018**] first create a parallel dataset by formally rewriting informal sentences from Yahoo answers. The task then essentially becomes an application of Neural Machine Translation (NMT) by translating from informal to formal. In addition to the manually rewritten sentences, they obtain more parallel data by creating a rule-based model from informal to formal (e.g. by changing words like *awweeesoommme* into *awesome*) and running it on the informal dataset. The work mainly focused on direction

informal to formal, since the opposite direction is often very arbitrary (there are many ways of making a formal sentence informal), thus making working with references difficult.

3.1.2 Unsupervised

Instead of relying on parallel data, many pieces of work focus on unsupervised approaches, since large parallel data is hard to obtain. However, we still have style labels for each sentence in the dataset.

[**Fu et al., 2017**] served as an inspiration for some of the models presented in this thesis. They used an autoencoder which attempts to reconstruct the original sentence, while removing the style from the latent representation. Unlike in this work, they achieve the latter by adding the entropy of a classifier for the latent representation to the total loss. In order to incorporate the target style into the representation, they train separate decoders on each style. They train their models on a dataset of Amazon-reviews for the style positive/negative and on a dataset of research paper titles and online news titles for the style paper/news. They also introduce metrics to evaluate the adequacy of the style and content of the generated sentence.

[**Shen et al., 2017**] enforces the decoder in an autoencoder-model to produce sentence of a desired style by a process called *cross-alignment*. The goal of the encoder and decoder is to produce hidden states, in which the original style is not detectable anymore. This is somewhat similar to [Fu et al., 2017], however the model is already conditioned on the target style and tries to generate a sentence with the target style, which is indistinguishable from a real sentences with the target style, as per a discriminator. The discriminator receives the sampled output of the net, whose error is then backpropagated into the decoder. One discriminator for each style is thus used. Note that the concepts of the discriminator in this paper and the classifier in [Fu et al., 2017] are quite different: here, the discriminator is conditioned on the target style and receives input that should already have been transferred, while the classifier in [Fu et al., 2017] receives data before the tar-

get style is incorporated. They composed and used a dataset consisting of restaurant reviews from Yelp which we also use in this thesis.

[**Prabhumoye et al., 2018**] translate to an intermediate language, in order to implicitly weaken the style, before translating back to the original language. The NMT-model performing the backtranslation has a separate decoder for each style, similar to [Fu et al., 2017]. However, it is additionally guided by a classifier, which operates on the sampled output of the decoder. The datasets used are the Yelp-dataset from [Shen et al., 2017] for the style positive/negative, as well as other ones for the styles gender and political slant.

[**Mueller et al., 2017**] uses an approach where at test time, gradient ascent is performed on the latent representation to cause it to be classified as the desired style. The modified latent representation is then fed to a decoder. This step is performed at test time. They apply this method to transfer negative to positive sentences and modern-style sentences to sentences written by Shakespeare.

[**Yang et al., 2018**] uses a language model only trained on the target style to guide the generator, whose output it receives. The assumption is that in order to be given a high probability by the language model, the sentence has to be in the same style as the training data for the language model, hence ensuring both grammatical and stylistic adequacy. It is argued that language models can provide more precise feedback than classifier. The latent representation is also fed to a decoder which aims to reproduce the input sentence, ensuring preservation of the original content in the latent representation.

[**Li et al., 2018**] proposes removing words strongly associated with the original style from the source sentence (not hidden representation) and then using a Sequence-to-Sequence (seq2seq) model conditioned on the target style and the modified sentence to perform the style transfer. The concept of removing the source style and then incorporating the target style is shared with [Fu et al., 2017], however at a different level. A more elaborate model tries to find a similar sentence to the source sentence after removing style-words. It then uses a seq2seq-model conditioned on the style words of the similar sentence in addition to the modified sentence. This gives the seq2seq further guidance on finding replacements

for the deletions.

3.2 Paraphrasing

Closely related to Style Transfer is the domain of Paraphrasing. The difference is that Paraphrasing only aims to differently express the content, while Style Transfer also aims to change the style of the text, which can't be entirely separated from changing the content.

[Gupta et al., 2017] employ an unsupervised approach where Variational Autoencoders (VAE) are used to create a latent representation of the sentence and then paraphrases are generated in a non-deterministic manner.

[Mallinson et al., 2017] uses an NMT-model to translate to a pivot language, combining multiple translations and back-translating.

3.3 Adversarial Training

[Ganin and Lempitsky, 2014] proposes a training method for eliminating a particular attribute from the latent representation. They employ a classifier for that attribute that receives the latent representation and train it in an adversarial manner. This is quite similar to [Fu et al., 2017], however the focus in this work lies on images instead of text.

Chapter 4

Architectures

We experiment with three very different architectures for this task. We describe them assuming there are only two possible styles such as positive/negative, although this is not a necessary restriction.

4.1 Adversarial Autoencoder

The high-level model architecture we will use in this section is closely related to the one proposed by [Fu et al., 2017]. It consists of:

an encoder $E(x; \ _E)$ receiving input sequence x and outputting a latent representation z

a classifier $C(z; \ _C)$ attempting to predict style l_x from the latent representation

a decoder $D_t(z; \ _D)$ attempting to reconstruct x from the latent representation, specific to target style label t

C is trained in an adversarial fashion against E and D for the purpose of removing as much style information from z as possible. Ideally, z is a representation of the content of x with its style removed. In order to apply the target style, we use a decoder trained only on decoding

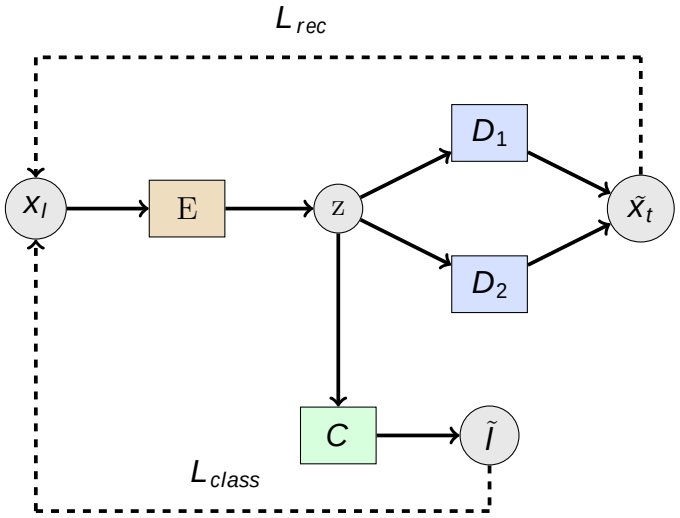


Figure 4.1: Adversarial autoencoder architecture with input-sequence x_l with style-label l. The encoder E produces latent representation z which is decoded by D_t into $\tilde{x}_t$ (L_{rec}), where t is our target-style. Additionally, our adversarial classifier C tries to predict l from z (L_{class}).

source sentences of one particular style. Mathematically, our classification loss is the cross entropy between style label l_x and predicted style:

$$L_{class} = \log p(l_x | E(x);\ _C) \tag{4.1}$$

The reconstruction loss becomes the cross entropy between the original sentence x and the output of the decoder:

$$L_{rec} = \sum_{i=1}^{M} \log p(x_i | D_{l_x}(E(x;\ _E);\ _D) \tag{4.2}$$

Our training objective for C is to find weights $_C$ that minimize the classification loss:

$$\min_C\ _{class} L_{class} \tag{4.3}$$

Intuitively, we seek to make the classifier good at classifying the latent representation. Our training objective for E and D is to find weights $_E;\ _D$ that minimize the reconstruction loss and maximize the classification loss:

$$\min_{E,\ D_{I_x}} \quad {}_{rec}L_{rec} \qquad {}_{adv}L_{class} \tag{4.4}$$

Intuitively, we try to output a sentence close to x while making it hard for C to classify the style given the latent representation.

Our approach slightly differs from the original approach proposed in [Fu et al., 2017], where a second adversarial loss L_{adv2}, being the entropy of C's predictions, was used to make E produce a style-independent latent representation. We tried this approach but preliminary results were poor, that is the accuracy of C reached 96%. This meant that L_{adv2} was too weak of a signal against L_{adv}, so z still contained all of the style, and as a result the decoder output was just x.

4.1.1 Low-level Architecture

Transformer Networks

As a second deviation from the approach in [Fu et al., 2017], we use Transformer Neural Network (see section 2.1), which E and D are a part of. Again, d_{model} denotes the model size, meaning the corresponding dimension of the output of all Transformer layers have this size, and d_{inner} denotes the size of the fully-conntected sublayers inside each layer.

Bottleneck

The encoder of the original Transformer produces a context vector z which we use as our latent representation to be classified by the classifier C. Since the original context vector is quite high-dimensional and this could provide a challenge to the classifier, we experiment with different modifications to reduce the dimensionality of z:

avg-bottleneck We average over the length-dimension of Z_T, resulting in size d_{model} 1. To make the decoder compatible again, we set its input length to 1.

ff-bottleneck We change the output size of the feed-forward layer inside the last Transformer-layer from d_{model} to $d_{bottleneck}$, resulting in a Z_T of size $d_{bottleneck}$ $len(x)$. To make the residual connection in the last encoder layer work again, we eliminate the excess dimensionality by averaging. For the encoder-decoder attention work again, we simply duplicate Z_T to make it compatible with the decoder dimension again.

no-bottleneck No modification is done to the original Transformer architecture, where Z_T is of size d_{model} $len(x)$

Classifier

Since z has a variable-sized time dimension for most of our bottleneck options, we feed each timestep of size d_{model} to an RNN that has d_C hidden units of the type Long Short-Term Memory (LSTM) (see [Hochreiter and Schmidhuber, 1997]). This is followed by a feed-forward layer and a softmax.

4.1.2 Training

There are multiple ways of implementing the adversarial training. In each case, every mini-batch only contains samples of one style.

alternate-batch For each mini-batch, we optimize $_C$ for C and then $_E$; $_{D_l}$ for E and D_l.

alternate-epoch For each epoch, we optimize either $_C$ for C or $_E$; $_{D_l}$ for E and D_l. In the first epochs we train them jointly, however.

grad-rev Following [Ganin and Lempitsky, 2014], we insert a gradient reversal layer just before the LSTM layer of C. We then jointly optimize $_C$, $_E$ and $_{D_l}$ to minimze $_{rec}L_{rec} +$ $_{adv}L_{class}$.

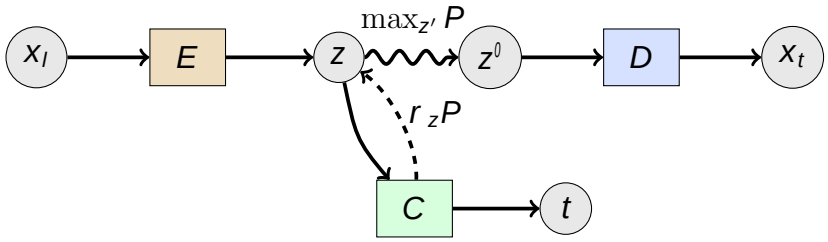


Figure 4.2: Classifier-guided revision model with one decoder during inference with input sequence x_l and style-label l. The encoder E produces latent representation z. We perform gradient ascent on z with the objective to be given higher probability for our target style t by C, and get z^0. The decoder D produces our transferred sequence x_t.

4.2 Classifier-guided Revision

We experiment with an approach similar to [Mueller et al., 2017]. Our model architecture is similar as in section 4.1, although for one of our models we only have one decoder:

an encoder $E(x;\ _E)$ receiving input sequence x and outputting a latent representation z

a classifier $C(z;\ _C)$ attempting to predict style l_x from the latent representation

a decoder $D(z;\ _D)$ attempting to reconstruct x from the latent representation, *not speci c to target style label t* for one model. For the other model, D_t is specific to t as in section 4.1.

Like in section 4.1, we use a Transformer for E and D.

4.2.1 Training

Unlike in section 4.1, the training objective is to minimize the reconstruction loss and classifier loss (see Equation 4.2 and 4.1) together:

$$\min_{E,\ D_{l_x}} \quad _{rec}L_{rec} + \ _{class}L_{class} \tag{4.5}$$

In practice, we train only reconstruction for 100 epochs and then only classifier for another 100 epochs, reaching a final validation classifier accuracy of 0.92.

We train the model with two decoders by using Decoder D_{l_x} specific to the source label, as in section 4.1.

4.2.2 Inference

The main idea behind this approach is executed during inference. We first compute our latent representation as $z = E(x_l;\ _E)$, where l is the style label of our input sequence, and then compute the log-probability that this is our target style label t as $P(z) = \log p(l = tjZ;\ _C)$. Then we compute the gradient of P with respect to z:

$$r_z P = \frac{@P}{@z} \tag{4.6}$$

and change z in the direction of the gradient in order to maximize P, also known also gradient ascent:

$$z^{0} = z + \quad r_z P \tag{4.7}$$

This is repeated for it iterations.

For the model with two decoders, decoder D_t specific to the target label is used.

4.3 Deleting Salient Words

In this section, we experiment with another model that is inspired by the work of [Li et al., 2018]. The idea is to remove style from the input sequence by directly deleting words associated with that style instead of removing style from a latent representation. For example, if our positive

input sentence was *The food was really good.*, we would simply delete the word *good*. We then pass this sentence to a seq2seq model additionally conditioned on the original style which aims to reconstruct our original sentence without deletions. For inference, we then condition the model on the opposite style.

4.3.1 Preprocessing

Our approach requires that style-specific words be deleted. We explore different ways of determining, which ones these words are. In every case, by deleting we mean replacing the token with a unique delete-token.

Statistical approach

Let x_l be the input sequence with l being the style label. Let $\bar{l}$ be the opposite style of l. As in [Li et al., 2018], we define the *saliency* of an n-gram u, u x_l as

$$\text{sal}_1(u; l) = \frac{\text{count}(u; D_l) +}{\text{count}(u; D_{\bar{l}}) +} \tag{4.8}$$

where $\text{count}(u; D_s)$ is the number of occurrences of u in the subset of dataset D that has label s, and is a smoothing-constant. Intuitively, the more often an n-gram occurs in one style compared to the other style, the higher our sal_1 will be.

We now delete n-grams that have a saliency greater than a fixed threshold . We iterate over the n-gram size n from largest to smallest, otherwise often times larger n-grams wouldn't be found anymore because a smaller n-gram had already been deleted from it. Let x_l^{del} denote the sentence with salient n-grams removed.

Gradient-based approach

We investigate a different approach, where instead of statistical methods, we delete words with the aid of a neural network, similar to the work of [Li et al., 2015]. More specifically, if C is

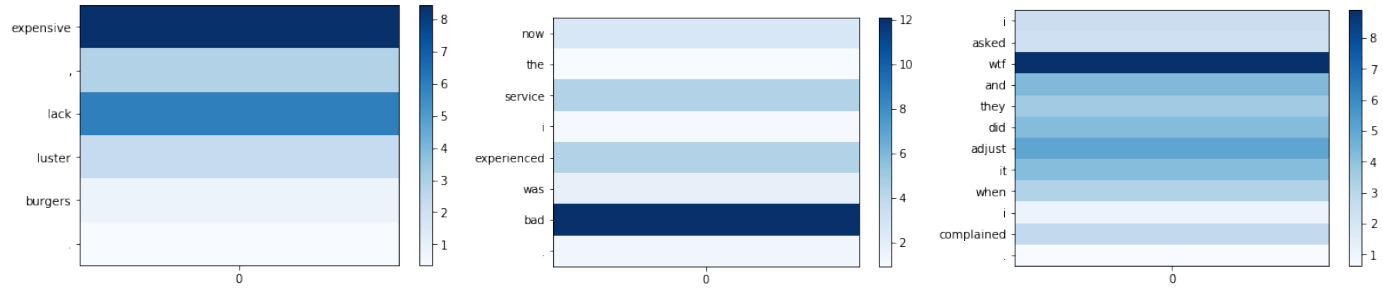


Figure 4.3: The sal_2-score is shown for every word in three different sentences. The metric does a good job in the first two leftmost sentences but fails to assign the word *complained* a high score in the rightmost sentence.

our style-classifier from subsection 5.2.1, we redefine the saliency for 1-grams as

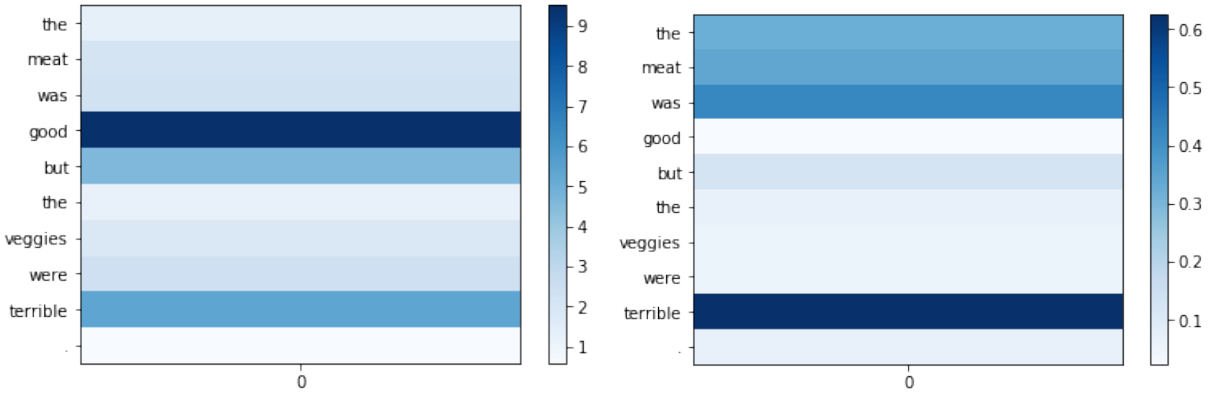
$$\text{sal}_2(u; l) = j \, r_{Emb[u]} \, p(\bar{l}|x; \ c)j \tag{4.9}$$

$r_{Emb[u]} \, p(\bar{l}|x; \ c)$ is gradient of the classification of the whole input sequence with respect to the embedding for the word $Emb[u]$. This will give words a high score whose embeddings will change the contribution for class $\bar{l}$ (our target-style) much when changed, and we hope that this coincides with a human-perceived relevance for the sentiment of the sentence. We only consider 1-grams in this approach. We use the model output before the softmax function. Furthermore, we observe that the value of sal_2 is almost the same whether we use $p(\bar{l}|x; \ c)$ or $p(l|x; \ c)$, and for this reason we always just use $p(0|x; \ c)$.

However, it is difficult to find an absolute threshold for sal_2 because the absolute magnitude appears to be fairly meaningless. We therefore decide to delete the k words with the highest saliency, with k being determined by the following heuristic dependent on the sentence length m:

$$k(m) = \begin{cases} 1; & \text{if } m \in [1; 4] \\ 2; & \text{if } m \in [5; 7] \\ 3 & \text{otherwise} \end{cases} \tag{4.10}$$

The longer the sentence, the more style-specific words tend to be present. However, often k is too large or too small if chosen like this, i.e. more words are deleted than needed or more style-specific words need to be deleted, respectively. Further work needs to explore this.



(a) sal_2 (b) sal_3, positive target

Figure 4.4: The sentence *The meat was good but the veggies were terrible.* contains both positive (*good*) and negative (*terrible*) words. If the target style is positive, we only want to delete *terrible* but not *good*, which is also what we get with our second definition sal_3. However, our first definition sal_2 isn't dependent of the target-style so it will delete both words.

As state before, we found out that the saliency as defined above is largely independent of our target-style, but ideally different words should be deleted for different styles (see Figure 4.4). We therefore experiment with another definition of saliency which is affected more by the target-label:

$$\text{sal}_3(u; l) = j\, r_{\,Emb[u]}\, p(\bar{l}jx;\ \ c)j \quad j\, r_{\,Emb[u]}\, p(ljx;\ \ c)j \tag{4.11}$$

With this definition, a word is deemed more salient if the embedding contributes more to the probability of the sentence being the target style than the original style. As we put this metric to use, we find that it is in fact very dependent on our target style but gives very poor values for deletion and for this reason, we don't pursue it further.

4.3.2 Training

We simply train a seq2seq model on reconstructing the original sequence x_l, conditioned on the modified sequence x_l^{del} and the original style label l, using reconstruction loss L_{rec} from Equation 4.2. We use the transformer architecture with two style-specific decoders, similar to section 4.1 but without a classifier.

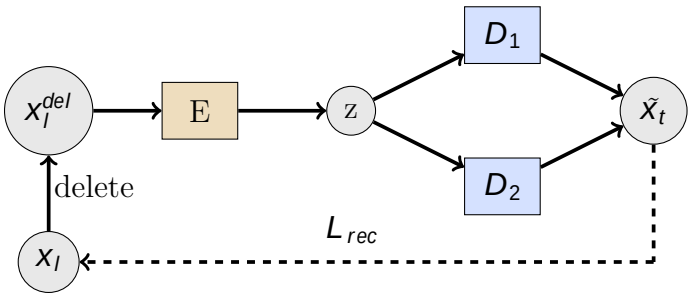


Figure 4.5: Deleting salient words architecture during training. We delete style-words from the original sentence x_l using one of the discussed methods and obtain x_l^{del}. Encoder E produces a latent representation z which is decoded by D_l into a sentence of the same style and as similar to x_l as possible (L_{rec}). During inference, the decoder of target style t is used instead.

4.3.3 Inference

During inference, we condition the decoder on the target style instead of the original style. In our implementation, this means that we use the style-specific decoder for the target style (see Figure 4.5).

4.3.4 Backtranslation

As suggested in [Li et al., 2018], this unsupervised style transfer can be used to generate a parallel dataset. If D_l is our original dataset, we run our model to transfer these to the opposite style, obtaining $\hat{D}_{\bar{l}}$. We then train a simple seq2seq model with one encoder and one decoder on back-translating to the original style $\hat{D}_l$ from the generated input $\hat{D}_{\bar{l}}$. We choose the generated, presumably low-quality data as the input and the human-written, high-quality data as the target because this way, the training signal comes from the target. Note that we can only use a model for one direction, so we have one model for *positive ! negative* and one for *negative ! positive*.

Chapter 5

Results

5.1 Datasets

Yelp Contains restaurant reviews from yelp divided into positive and negative, released by [Shen et al., 2017]. The sentiment is very strong and can be accurately predicted by a classifier (see subsection 5.2.1). It has 444101 sentences in the training set and 126670 in the validation set, 60.2% of both have a negative label. However, for performance reasons we report all metrics only on the first 1000 sentences of the validation set of each style.

5.1.1 Preprocessing

For tokenization, we use the `tokenizer.perl` from the Moses machine translation system [1].

We also experiment with *byte pair encoding (BPE)*, which is a way to reduce the vocabulary size of the model while still being able to differentiate between different rare tokens (as opposed

[1] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl

Style	Train	Dev	Valid
Positive	267314	38205	76392
Negative	176787	25278	50278

Table 5.1: Splits for the Yelp-dataset (number of sentences)

to replacing with an unknown token), as described in [Sennrich et al., 2015]. At train time, it learns a set of merge operations as follows: in the beginning, there are only tokens of length 1 (characters) with their corresponding frequency in the corpus. Then iteratively, the two most common symbols are combined into a new token, until some upper limit for vocabulary size or lower limit for token frequency is reached. Then during test time (or on the validation split) this procedure is repeated with the exception that the merge operations are applied as they were learned during train time. We use an implementation from Github [2].

5.2 Metrics

5.2.1 Transfer strength

To evaluate how strong the output sentence has assumed the desired style, we use the metric *transfer strength*, introduced in [Fu et al., 2017]. It is defined as the ratio of output sentences $\tilde{x}$ classified as the target style t. More formally, we have a style classifier $C_{Ev}(\tilde{x};\ _{C_{Ev}})$. Then we have transfer strength

$$tf(\tilde{X}) = \frac{1}{j\tilde{X}j} \sum_{x2\tilde{X}} predict(\tilde{x}) \tag{5.1}$$

where

$$predict(\tilde{x}) = \begin{cases} 1; & \text{if } p(tj\tilde{x};\ _{C_{Ev}}) \quad 0{:}5 \\ 0; & \text{otherwise} \end{cases} \tag{5.2}$$

As our C_{Ev} we train an RNN to classify the style of the input sentence. It has a 200-dimensional input word embedding initialized with pretrained Glove vectors, followed by one bidirectional hidden GRU-layer of size 73, followed by a softmax. [3]. On the yelp dataset, it reaches a near-perfect validation accuracy of 97.9% which is almost the same as what [Shen et al., 2017] achieved with a CNN.

[2] https://github.com/rsennrich/subword-nmt/blob/master/subword_nmt/learn_joint_bpe_and_vocab.py
[3] http://nlp.stanford.edu/data/glove.twitter.27B.zip

5.2.2 Content preservation

Since we aim to generate a sentence similar to our input sentence, we need a metric to capture the similarity of two sentences. For this, we use *content preservation*, also introduced in [Fu et al., 2017]. It computes the minimum, maximum and average word embedding in both sentences and then takes the cosine similarity, that is a scaled dot-product, between the two. So if w_{x_i} denotes the embedding of word x_i, content preservation cp for sentences $X; Y$ is calculated as follows:

$$v^X_{min} = \min_{x2X} \; w_x$$

$$v^X_{mean} = \text{mean}_{x2X} \; w_x$$

$$v^X_{max} = \max_{x2X} \; w_x$$

$$v = [v_{min}; v_{mean}; v_{max}]$$

$$cp(X; Y) = \frac{(v^X)^T v^Y}{kv^X k \;\; kv^Y k}$$

$$= \frac{v^X_{min} v^Y_{min} + v^X_{max} v^Y_{max} + v^X_{mean} v^Y_{mean}}{kv^X k \;\; kv^Y k}$$

Note that the $cp \; 2 \; [\; 1; 1]$, with high values indicating high similarity. To get a random baseline of this metric, we compute the average cp between 10000 random sentence pairs from two subsets of our yelp validation set. We get the following values for different choices of subsets: 0.866 for (valid-negative, valid-negative), 0.864 for (valid-positive, valid-positive) and 0.86 for (valid-positive, valid-negative). For more intuition, the sentence pair *Hello, how are you?* and *My dog got sick.* has a cp of 0.851, indicating that very dissimilar sentences can still receive a high cp.

5.2.3 Fluency

Content preservation doesn't explicitly measure how fluent, i.e. in line with a sentence produced by a native speaker, the sentence is. For this purpose, we train a language model LM on pre-

dicting the next word given all previous words: $p(x_i|x_{i-1},\ldots,x_1;\ LM)$. It has a 300-dimensional randomly initialized word embedding, followed by two LSTM layers with 300 dimensions each, followed by a softmax. On our yelp dataset, it achieves a validation perplexity of 24.44. As our fluency score, we add the log probabilities of the words in the sentence and divide them by the length of the sentence:

$$fl(x) = \frac{1}{|x|} \sum_{x_i \in x} \log p(x_i|x_{i-1},\ldots,x_1;\ LM) \tag{5.3}$$

Higher fl-values indicate a higher fluency. For intuition purposes, we run fl on 1000 sentences from the yelp validation set and get an average of -4.622 for negative sentences and -4.076 for positive sentences.

5.3 Adversarial Autoencoder

The results of our models and two reference models are shown in Table 5.2. Our model configurations are as follows:

large-batch We use training option *alternate-batch*. We set $d_{model} = 512$, $d_{inner} = 1024$, 2 encoder and decoder layers each, $d_c = 50$ and we use bottleneck option *avg-bottleneck*. As weights we use $\lambda_{rec} = 1$ and $\lambda_{adv} = 4$. No BPE is used.

small-batch We use training option *alternate-batch*. We set $d_{model} = 128$, $d_{inner} = 256$, 2 encoder and decoder layers each, $d_c = 25$ and we use bottleneck option *avg-bottleneck*. As weights we use $\lambda_{rec} = 1$ and $\lambda_{adv} = 4$. BPE is used.

small-epoch We use training option *alternate-epoch*. We set $d_{model} = 128$, $d_{inner} = 256$, 1 encoder and decoder layers each, $d_c = 5$ and we use bottleneck option *no-bottleneck*. As weights we use $\lambda_{rec} = 1$ and $\lambda_{adv} = 1$. BPE is used.

small-rev We use the training option *grad-rev*. We set $d_{model} = 128$, $d_{inner} = 256$, 2 encoder and decoder layers each, $d_c = 25$ and we use bottleneck option *no-bottleneck*. As weights

Model	Transfer strength	Fluency	Content pres
Cross-align ([Shen et al., 2017])	0.818	**-4.22**	0.93
MultiDecoder ([Fu et al., 2017])	**0.924**	-4.77	0.9015
large-batch	0.398	-4.772	**0.976**
small-batch	0.76	-5.124	0.92
small-epoch	0.157	-4.692	0.972
small-rev	0.276	-4.436	0.971
baseline	0.24	-4.32	0.984

Table 5.2: Comparison of different adversarial autoencoder models we used

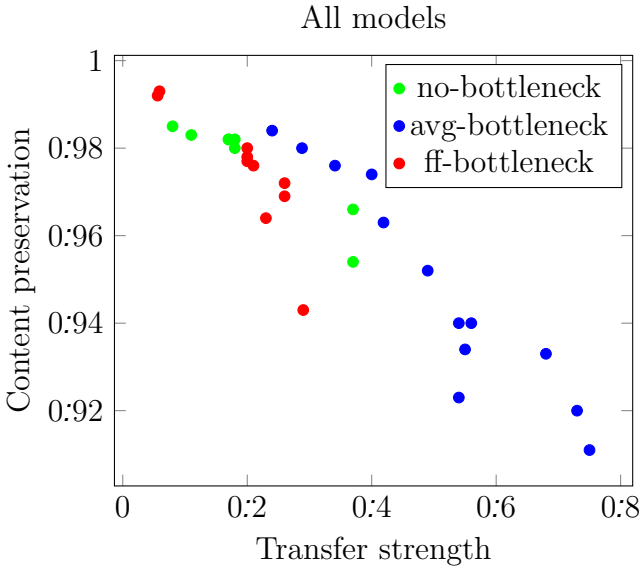


Figure 5.1: These adversarial autoencoders with different parameters show a trade-off between transfer strength and content preservation. Since *avg-bottleneck* often has a smaller size than the other two bottleneck options, the classifier's work becomes easier so the transfer strength is higher.

we use $_{rec} = 1$ and $_{adv} = 4$. BPE is used.

baseline Same as *large-batch* except $_{adv} = 0$ which means that no adversarial training is taking place.

5.4 Classifier-guided Revision

We set the number of layers in E and D to 2, $d_{model} = 512$, $d_{inner} = 1024$ and $d_C = 50$ for all models. We also experiment with bottleneck options *no-bottleneck* with $= 200$ and *avg-bottleneck* with $= 2:3$ (see section 4.1.1). We use $it = 40$ for both options. BPE is applied to

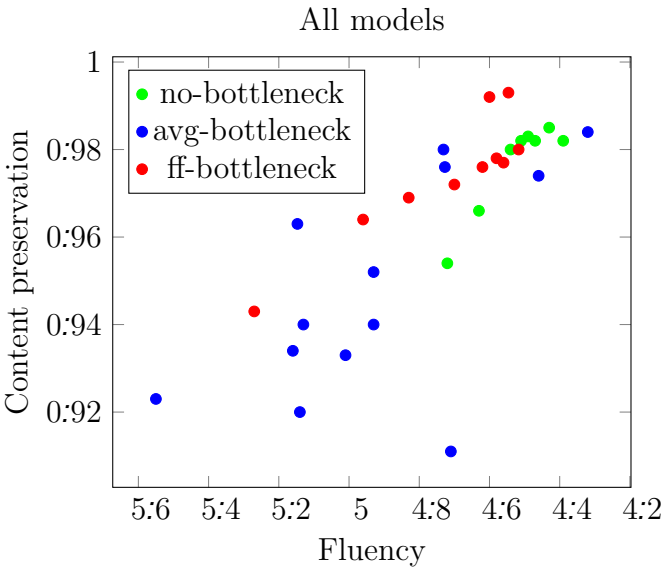


Figure 5.2: We can see that for adversarial autoencoders, fluency strongly correlates with content preservation, although some models with the same content preservation have significantly better fluency.

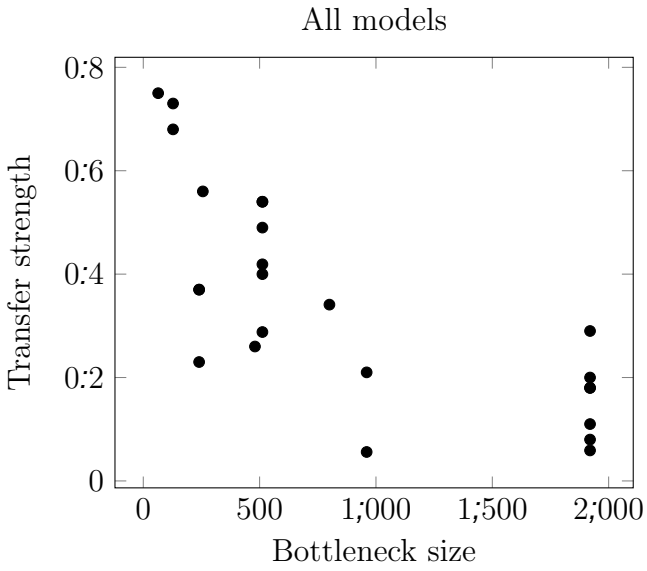
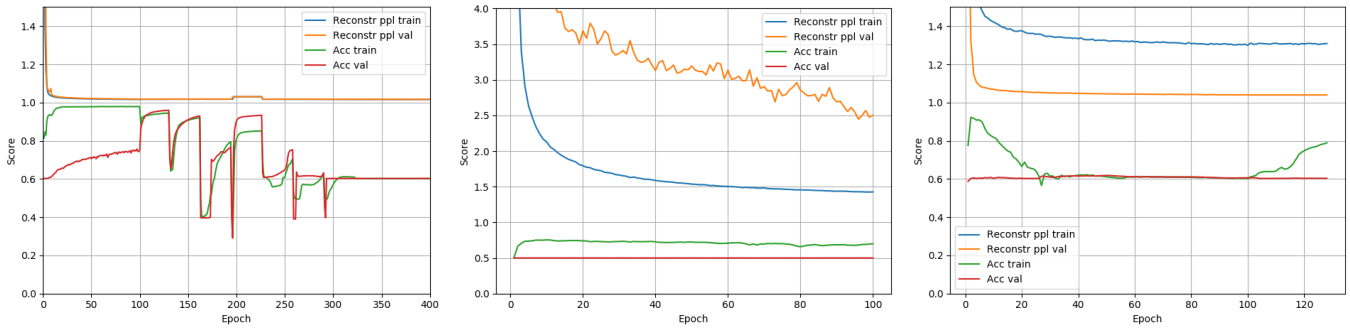


Figure 5.3: The smaller the bottleneck size in adversarial autoencoders (for all three bottleneck options), the stronger transfer strength tends to be. A possible explanation for this is that with a large bottleneck size, it is difficult for the classifier to achieve a high accuracy so the adversarial training signal is too small, thus the latent representation still contains the original style. At the same time, it is more difficult for the network to perfectly reconstruct the input sentence.

rec	adv	Transfer strength	Fluency	Content pres
4	1	0.398	-4.772	0.976
40	1	0.288	-4.731	0.98
4	0	0.24	-4.32	0.984

Table 5.3: Comparison of *avg-bottleneck, no bpe* models with different -parameters for adversarial autoencoders. When $_{rec}$ is very high, transfer strength decreases and content preservation increases. The same thing happens when $_{adv}$ is set to 0, which means the model is only trained on reconstruction.



(a) *small-epoch* (b) *large-batch* (c) *small-rev*

Figure 5.4: Training curves for three different adversarial autoencoders. On the left, we see that the classifier accuracy gets lower and lower after each adversarial phase until it doesn't improve at all, whereas the reconstruction error barely rises. In the center, we see that the validation accuracy of the classifier doesn't improve at all. On the right, the train accuracy goes up and then down due to the annealing, but the validation accuracy barely goes up.

the data as preprocessing. We experiment with two training schedules: we call *alternate* the schedule where only L_{rec} is trained first and then only L_{class} is trained, which is the default unless specified otherwise; we call *simultaneous* the schedule where our loss is $L_{rec} + L_{class}$ first and in the end only L_{class} again. We report classifier accuracy on the validation set at the end of training, in addition to the metrics from section 5.2.

1dec Uses bottleneck option *avg-bottleneck* and only one decoder. We use $it = 40$ and = 2:3.

1dec-nobn Uses bottleneck option *no-bottleneck* and only one decoder. We use $it = 40$ and = 200.

2dec Uses *avg-bottleneck* and two decoders that are specific to the target label.

2dec-simult Unlike all the other models, we use training schedule *simultaneous*. Otherwise the same as *2dec*.

Model	Transfer strength	Fluency	Content pres	Classif accuracy
1dec	0.649	-5.175	0.95	92.4%
1dec-nobn	0.3	-6.067	0.925	92.4%
2dec	**0.844**	-4.781	0.948	86.8%
2dec-simult	0.631	-4.856	0.931	94.4%
2dec-nograd	0.397	**-4.546**	**0.966**	86.8%
2dec-random	0.571	-5.596	0.923	86.8%

Table 5.4: Results for all classifier-guided revision models.

2dec-nograd No gradient ascent on z is performed, so the unmodified z is simply decoded by style-specific decoder D_t. Uses option *avg-bottleneck*.

2dec-random Since we want to be sure that C provides a useful signal, we have a baseline where a random perturbation is applied to $r_z P$ in each gradient ascent step. We implement this by multiplying each gradient dimension by a factor sampled from a uniform distribution in the range [2; 2]. Two decoders and option *avg-bottleneck* is used.

5.5 Deleting Salient Words

We set the number of layers in E and D to 2, $d_{model} = 512$ and $d_{inner} = 1024$. Neither bottleneck nor byte pair encoding is used. We report multiple models in Table 5.5:

statistical We use the statistical deletion approach and follow the recommendation of [Li et al., 2018] for a similar Yelp-review dataset of choosing = 1 and = 15. We also use = 7 for a different model.

gradient For the gradient-based approach, we report our model with sal_2 as our saliency definition.

random As a baseline we train a model where for each sentence, k (according to our heuristic) random words have been deleted.

bt Back-translation with parallel data generated from the specified model. Since each model can only handle one direction, the numbers shown are the average of two models.

Model	Transfer strength	Fluency	Content pres	del negative	del positive
DeleteAndRetrieve	**0.895**	**-4.377**	0.934		
DeleteOnly	0.869	-4.652	0.937		
statistical-15	0.706	-5.114	0.94	29%	42%
statistical-7	0.82	-5.188	0.911	42%	55%
gradient	0.639	-5.079	0.928	27%	28%
random	0.294	-4.864	0.96	27%	28%
bt-statistical-15	0.718	-4.765	**0.956**	0	0
bt-gradient	0.692	-4.766	0.954	0	0

Table 5.5: Results for delete-salient-models. We specify the three metrics from section 5.2 as well as how many percent of the words were deleted in each dataset. We notice that statistical-15 outperforms the gradient-model in both transfer strength and content preservation. However, the gradient-model is still superior to random deletions by a large margin, indicating that the gradient can give us a useful signal for saliency. Both back-translation models significantly greatly improve fluency and content preservation and slightly improve transfer strength, compared with the model that generated their data. We also show the results of the *DeleteOnly* and *DeleteAndRetrieve* models from [Li et al., 2018], the former being related to *statistical-15*.

Model	Transfer strength	Fluency	Content pres
MultiDecoder ([Fu et al., 2017])	**0.924**	-4.77	0.9015
large-batch (section 4.1)	0.398	-4.772	**0.976**
2dec (section 4.2)	0.844	-4.781	0.948
DeleteOnly ([Li et al., 2018])	0.869	**-4.652**	0.937
bt-statistical-15 (section 4.3)	0.718	-4.765	**0.956**

Table 5.6: Comparison of best models from each section with baselines from other papers.

All metric results are the average of both targets, *negative* and *positive*.

5.6 Comparison

As a summary, we compare the best models from each section in Table 5.6. The models from section 4.2 and section 4.3 could be judged as much better than the ones from section 4.1 due to a better tradeoff between transfer strength, fluency and content preservation.

Chapter 6

Discussion

6.1 Adversarial Autoencoder

Our results show that it is at least very difficult to achieve both a high transfer strength and content preservation with this kind of model. This isn't simply an issue of our implementation, since the original implementation shows similar results.

It is problematic that our classifier tries to classify a latent representation, since the Encoder can be trained to represent a function that makes this task very hard for the decoder, with the decoder still being able to decode almost perfectly. This is shown by the training curve of *small-epoch* in Figure 5.4, for example.

Other research on this topic, e.g. [Prabhumoye et al., 2018] and [Shen et al., 2017], mostly uses a classifier that receives sampled words as input instead of a latent representation, which seems to work better. This makes it more difficult for the classifier to be tricked since its input have to be actual words.

6.2 Classifier-guided revision

Since the model *bn-1dec* is already fairly successful at applying the target style, we can deduce the the backprop method with the classifier alone is quite powerful. This is further shown by the drastic gain of *2dec* compared to *2dec-nograd* in transfer strength. *1dec-nobn* likely didn't work because of the high dimensionality of the hidden representation. Furthermore, we can show that the gradient ascent is more than just a random perturbation of z because *2dec-random* is strictly dominated by *2dec*.

Another observation is that the two-decoder model is superior to the one-decoder model in both transfer strength and fluency. The former is obvious and an explanation for the latter could be that z' is closer to hidden representations that the style-specific decoder was trained on, therefore it can be decoded better.

Why the model trained with *alternate* outperforms the one trained with *simultaneous* is unclear.

6.3 Deleting Salient Words

The statistical deletion approach outperforms the gradient approach in both transfer strength and content preservation, which means that the definition sal_1 is a better guidance than sal_2 as to which words contain style, at least with the hyperparameters that were used. One possible explanation for this is that gradient descent on the input does not always align with human judgement. For example, a particular word change might change the classifier output but not human judgement. This has been studied with the goal of tricking the classifier, for example in [Ebrahimi et al., 2017]. Furthermore, it may be that improving the k-metric is responsible for the inferiority of the gradient approach. The k could be computed with a statistical approach instead of a heuristic one, for example by just using the number of deleted words with the statistical approach in that sentence. Alternatively, words could be deleted iteratively from high to low saliency until the classifier gives the source label low enough probability. Lastly, sal_3 looks promising on paper and on a few examples and can hopefully be modified in some

way to work well on most of the dataset. We leave these things for future work. Nevertheless, the gradient approaches beat the random baseline by a huge margin in terms of style strength, indicating that the gradient can give us a useful signal for saliency. The relatively high content preservation of the random model can be attributed to the fact that easy-to-retrieve words such as *and, but, or, I*, etc are deleted more often.

Back-translation greatly improves both fluency and content preservation and slightly improves transfer strength. This can be explained by the fact that the back-translation model is trained and evaluated in exactly the same way, namely producing the opposite style. By contrast, the unsupervised model is trained generating the same style but evaluated (inference) on generating sentences of one style given a sentence of the other style. The latter has style words removed but this can never completely align the distributions for the two styles. An additional explanation is that the unsupervised model doesn't receives a less direct training signal, since some of the input-words have been deleted.

One flaw with the current back-translation model is that the unsupervised model doesn't produce very diverse del-replacements as training data for it, mostly with the *negative* target, where it replaces with *num* very often. More diverse outputs of this model are desirable because it allows the supervised model to learn more mappings from input to output words.

Compared with the baselines from [Li et al., 2018], our models score worse in transfer strength and fluency but better in content preservation. The only difference between *DeleteOnly* and our *statistical-15* is that the latter uses Transformer networks, in addition to other possible implementation differences.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this work, we experimented with three different methods to perform unsupervised style transfer. The adversarial autoencoder achieved only very moderate success, however classifier-guided revision and deleting salient words were able to produce many grammatically sound sentences that fulfilled the target style. This is interesting because we can control for a particular style without having explicit examples of how to do this (parallel data) which makes data collection much easier. We also showed how these can be used with the Transformer architecture instead of with RNNs, leading to faster training times.

7.2 Future Work

An interesting direction of research would be to train these models not only on applying a style in the narrow sense such as emotions but on controlling other aspects of the input such as certain content or grammar attributes. This would be possible with no major modification to our methods because our training objective is sufficiently general, since we are trying to control for one attribute, which in this work is style, while leaving the sentence as unchanged

as possible. As a toy example, one could train the model to conjugate all present tense verbs into past tense, or to only use indefinite articles instead of definite ones.

Appendix A

Sample Output

We provide sample outputs for the models listed in Table 5.6. The original sentences are from the validation split of the Yelp dataset.

Original	we had the shrimp with vegetables and shrimp fried rice - both lovely .
MultiDecoder	we ordered the sauce and num people and chicken was our waste ?
large-batch	we had the shrimp with vegetables and shrimp fried rice - both vegetables very lovely
2dec	we gave it the shrimp and shrimp with vegetables and shrimp said them was flat .
DeleteOnly	we had the shrimp with vegetables and shrimp fried rice - both dry and cold .
bt-statistical-15	we had the shrimp with vegetables and shrimp fried rice - both tasteless .
Original	high quality food at prices comparable to lower quality take out .
MultiDecoder	poor food too flavor and room was a bad money .
large-batch	high quality food prices , comparable to lower quality take out num back .
2dec	low quality food at prices to lower quality to take num minutes out .
DeleteOnly	food at prices were too high to comparable quality lower to take our order .
bt-statistical-15	below average food at prices comparable to lower quality take out .
Original	i will be going back !
MultiDecoder	i will be going back !
large-batch	i will not be going back !
2dec	i will not be going back !
DeleteOnly	i do n't think i will be going back !
bt-statistical-15	i will be going elsewhere .
Original	my appetizer was also very good and unique .
MultiDecoder	my food did n't good was extremely disappointed .
large-batch	my appetizer was also very good and very salty and cold flavor .
2dec	my appetizer was was also very dry and tasteless .
DeleteOnly	i gave up the only two stars if my my appetizer here was not good .
bt-statistical-15	my appetizer was mediocre at best and bland .
Original	i was just one of those lucky enough to find this one !
MultiDecoder	i was really num stars , this place had a num minutes .
large-batch	i was just one of those lucky enough to find this one to this one !
2dec	i was just one of those standing enough to tell this to find this one !
DeleteOnly	i was just all the long number lucky to pretend one out this same !
bt-statistical-15	i was just one of those lucky enough to find this one !

Table A.1: positive *!* negative

Original	the $ num minimum charge to use a credit card is also annoying .
MultiDecoder	the best chinese food my favorite food i went is recommended !
large-batch	the $ minimum charge card to use a credit card is also so credit .
2dec	the $ num to use a credit card is also very nice also .
DeleteOnly	the $ num minimum charge to use a credit card is also great !
bt-statistical-15	the $ minimum minimum charge to use a credit card is also addictive .
Original	my goodness it was so gross .
MultiDecoder	my recommend it was always delicious .
large-batch	my goodness it was so delicious .
2dec	my goodness it was so tender and delicious food .
DeleteOnly	i always love my goodness my delicious it goodness it was delicious .
bt-statistical-15	my goodness it was so delish !
Original	if i could give them a zero star review i would !
MultiDecoder	if you are a best best to anyone the salon ! you ever !
large-batch	if i could give them a five star review i would recommend them !
2dec	if i could give them a great star review and i ' m !
DeleteOnly	if i had a great experience that i review would !
bt-statistical-15	if i could give them a tremendous star review i would !
Original	it was super dry and had a weird taste to the entire slice .
MultiDecoder	it was super friendly and a very nice for a great here .
large-batch	it was super dry and had a weird taste to the las vegas slice .
2dec	it was super tasty and had a great taste to the entire slice of the entire .
DeleteOnly	it was very filling and had a real taste at the slice .
bt-statistical-15	it was super juicy and had a delicious taste to the entire slice .
Original	sorry but i do n't get the rave reviews for this place .
MultiDecoder	will always a friendly is the best nice here for a nice here .
large-batch	sorry but i do get to the rave reviews for about this place .
2dec	anyway , i ' ll get the great reviews for this is the place .
DeleteOnly	i love dr. rave reviews for this place .
bt-statistical-15	sorry but i do n ' t get the rave reviews for this place .

Table A.2: negative *!* positive

Bibliography

[Ebrahimi et al., 2017] Ebrahimi, J., Rao, A., Lowd, D., and Dou, D. (2017). HotFlip: White-Box Adversarial Examples for Text Classification. *ArXiv e-prints*, page arXiv:1712.06751.

[Fu et al., 2017] Fu, Z., Tan, X., Peng, N., Zhao, D., and Yan, R. (2017). Style Transfer in Text: Exploration and Evaluation. *ArXiv e-prints*.

[Ganin and Lempitsky, 2014] Ganin, Y. and Lempitsky, V. (2014). Unsupervised Domain Adaptation by Backpropagation. *ArXiv e-prints*.

[Gatys et al., 2015] Gatys, L. A., Ecker, A. S., and Bethge, M. (2015). A Neural Algorithm of Artistic Style. *ArXiv e-prints*.

[Gupta et al., 2017] Gupta, A., Agarwal, A., Singh, P., and Rai, P. (2017). A Deep Generative Framework for Paraphrase Generation. *ArXiv e-prints*.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

[Li et al., 2015] Li, J., Chen, X., Hovy, E., and Jurafsky, D. (2015). Visualizing and Understanding Neural Models in NLP. *ArXiv e-prints*, page arXiv:1506.01066.

[Li et al., 2018] Li, J., Jia, R., He, H., and Liang, P. (2018). Delete, Retrieve, Generate: A Simple Approach to Sentiment and Style Transfer. *ArXiv e-prints*.

[Liu et al., 2017] Liu, M.-Y., Breuel, T., and Kautz, J. (2017). Unsupervised Image-to-Image Translation Networks. *ArXiv e-prints*.

[Luan et al., 2017] Luan, F., Paris, S., Shechtman, E., and Bala, K. (2017). Deep Photo Style Transfer. *ArXiv e-prints*.

[Mallinson et al., 2017] Mallinson, J., Sennrich, R., and Lapata, M. (2017). Paraphrasing revisited with neural machine translation. pages 881–893.

[Mueller et al., 2017] Mueller, J., Gifford, D., and Jaakkola, T. (2017). Sequence to better sequence: Continuous revision of combinatorial structures. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2536–2544, International Convention Centre, Sydney, Australia. PMLR.

[Prabhumoye et al., 2018] Prabhumoye, S., Tsvetkov, Y., Salakhutdinov, R., and Black, A. W. (2018). Style Transfer Through Back-Translation. *ArXiv e-prints*.

[Rao and Tetreault, 2018] Rao, S. and Tetreault, J. (2018). Dear Sir or Madam, May I introduce the GYAFC Dataset: Corpus, Benchmarks and Metrics for Formality Style Transfer. *ArXiv e-prints*.

[Sennrich et al., 2015] Sennrich, R., Haddow, B., and Birch, A. (2015). Neural Machine Translation of Rare Words with Subword Units. *ArXiv e-prints*.

[Shen et al., 2017] Shen, T., Lei, T., Barzilay, R., and Jaakkola, T. (2017). Style Transfer from Non-Parallel Text by Cross-Alignment. *ArXiv e-prints*.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need. *ArXiv e-prints*, page arXiv:1706.03762.

[Yang et al., 2018] Yang, Z., Hu, Z., Dyer, C., Xing, E. P., and Berg-Kirkpatrick, T. (2018). Unsupervised Text Style Transfer using Language Models as Discriminators. *ArXiv e-prints*.