NLP Final Course Project
Report

Unsupervised Text Style Transfer

Submitted by

Name	Student No.
Saeed Hedayatian	97100292
Ali Marashian	97102441
Maryam Gheysari	98209178

Course Instructor
Ehsaneddin Asgari

Department of Computer Engineering
Sharif University of Technology

Fall 2021

Abstract

Style transfer is identified as a central problem in sequence to sequence modelling and in particular, text generation domains. Not having access to a parallel dataset for training (i.e., unsupervised style transfer) adds another layer of difficulty to this already complicated problem. In this project we implement a strategy for solving this problem that is inspired by Generative Adversarial Networks (GANs). We also discuss some of the challenges that we faced along the way and how they could be countered with.

Contents

Chapter 1

Introduction

1.1 Motivation

In the recent years, neural style transfer has experienced a lot of progress and various methods have been proposed. However, most of these efforts have focused on the domain of images. In our opinion, text style transfer has not received enough attention, despite being a central problem in sequence-to-sequence modelling. By carefully defining what constitutes the *style* of a text, one can view a large class of text generation problems as instances of style transfer. Simplifying or modernizing complex sentences, summarizing a large body of text, and even machine translation can all be viewed as changing the style of given input text. Thus, studying the style transfer problem in its general form will help in many other areas of natural language processing as well.

In this project, we focused on an even more general and difficult problem, unsupervised style transfer. The main difference is that in the unsupervised case, we only assume access to un-parallel corpora of different styles. That is, we would like to design a model that is able to change the style of an input to an style of our choosing by learning from a set of disjoint and possibly unrelated corpora, each containing texts with a particular style. This restriction on our training dataset makes the problem significantly more difficult as traditional supervised learning techniques can not be used in this setting.

In the remainder of this section, we will concisely define the problem that we are trying to solve. In the next chapter, we will introduce our method for solving this problem and discuss the particular challenges that we faced. We will then discuss the results that we were obtained on a sentiment transfer task and elaborate on the challenges of evaluating a style transfer

model. Finally, we will finish with a discussion on possible avenues for future research.

1.2 Problem Definition

We shall now formally define the unsupervised style transfer problem. But before we begin, notice that there is an inherent vagueness in the definition of *style* and *content*. While there have been efforts to formalize these notions, we don't intend to do so as we believe this vagueness is what gives style transfer methods their generality. It suffices to say that style can be either a very clear aspect of a text, for instance it's sentiment, or a very complicated attribute of it, for instance the particular writing style of poets in a specific era.

Assume we have identified n styles, $s_1, \cdots, s_n$ for texts. For the sake of definition, assume that there exists a perfect style classification oracle, $\mathcal{O}_s$, that can perfectly determine the style of a given text, and a perfect content examination oracle, $\mathcal{O}_c$, that outputs 1 if its input sentences convey the same meaning and 0 otherwise (we don't have access to either of these oracles). In the unsupervised style transfer problem, we assume access to n corpora of text, namely $D_1, D_2, \cdots, D_n$, each containing sentences of a particular style. That is, for any sentence x in D_i, $\mathcal{O}_s(x) = s_i$.

A style transfer model, $\mathcal{M}$, should take as input a sentence x and a designated style s_i and output a new sentence, y that has the same content as x, but with the style s_i:

$$\mathcal{M}(x, s_i) = y \text{ s.t. } \mathcal{O}_c(x, y) = 1, \mathcal{O}_s(y) = s_i.$$

Notice that we have access to neither O_c nor O_s, which complicates the evaluation of the success rate of $\mathcal{M}$. We will further discuss this issue in the evaluation chapter.

1.3 Datasets

We used two datasets to test our model on. First, this collection of reviews and comments that people wrote on SnappFood, an Iranian food delivery service. This dataset was originally created for sentiment analysis, as each comment is labeled with either a "Happy" or "Sad" label. We defined a sentence's sentiment to be its style. Our model therefore tries to turn positive comments into negative ones without changing their overall content and vice versa. We identified some sentences that were mislabeled in this dataset,

and also some sentences that were neither happy nor sad. We would expect better results if we use a more accurate dataset with possibly more classes than merely using happy and sad. We used the original train/validation/test split that was proposed.

We also prepared another dataset, composed of poems from different Persian poets (in particular, we used the works of Saadi, Khaghani, and Sanaie, which we thought exhibited different poetic styles). These poems were scrapped from Ganjoor and we downloaded them from here. This dataset was also prepared for our purposes and is available at our project's GitHub, though we did not have enough time to run experiments on this dataset.

We used some minimal pre-processing in form of normalization on these datasets and then trained our own byte pair encoder [4] to tokenize the text before feeding them to our neural modules.

Chapter 2

Method

2.1 Overview

Our work is mainly inspired by [5], though there are some differences. The idea is to remove the style from a given text, leaving only its bare content, and then regenerate it in the desired style. First, the encoder takes in the input and generates contextualized representations of the input, but we want the output of the decoder to be free of the initial style. Given the target style, the decoder generates tokens in that style. In order to make the encoder learn to generate representations that have minimal hints of the initial style, we use the signal from another model, the discriminator: if the encoder is to successfully empty the text from its style, it must be hard for a discriminator to guess that initial style. Figure 2.1 provides a schematic view on how training is done in our method, while figure 2.2 illustrates how the model is used for inference.

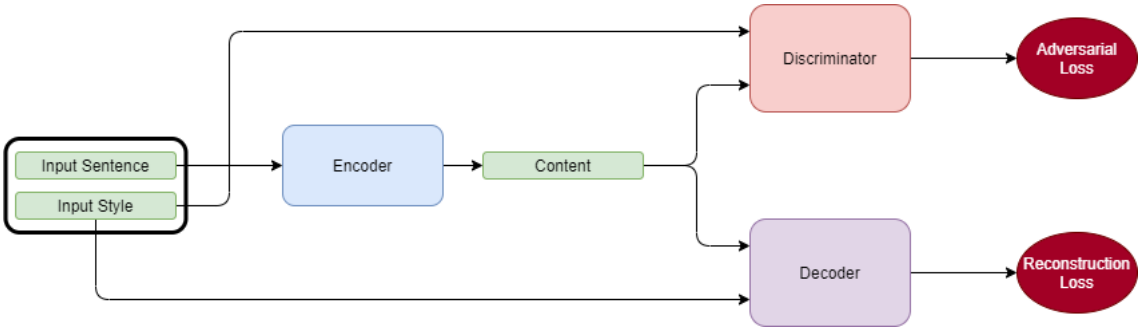

Figure 2.1: Flow of data during training of model. The encoder receives loss signals from both the discriminator (adversarial loss) and decoder (reconstruction loss).

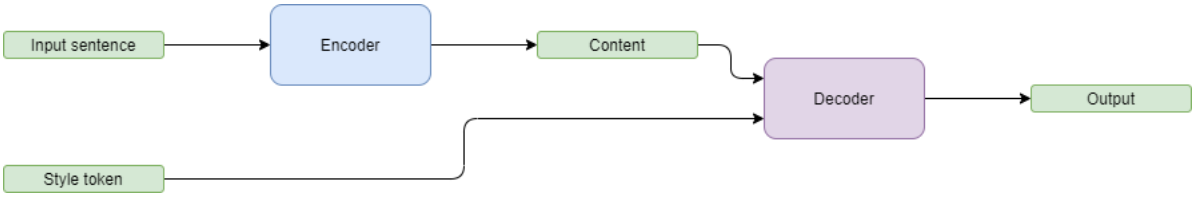

Figure 2.2: How the model is used during inference to change the style of a given sentence.

2.2 Models

Our approach comprises three models, all of which we will describe in detail.

2.2.1 Encoder

The encoder is a normal Transformer [7] encoder. Because our data is rather limited, we decided to have only a few layers of encoder layers (in our experiments, we usually had about 2 layers). This module, trained properly, would map the input to a representation solely based on its content; i.e. different styles would be close in the latent space, so as to be indistinguishable for the discriminator.

2.2.2 Decoder

The decoder is a normal Transformer decoder. The goal of the decoder is to generate -with the strategy of choice- an output in the desired style. Starting with the style token, and using the output of the encoder, we iteratively generate the next token(s), until we generate the EOS token. In learning, we use a hybrid method: first, freezing the decoder, we generate the tokens. having access to the ground truth (the original input), we then mix that ground truth and the generated tokens and consider that the new ground truth in hopes of having a faster convergence.

2.2.3 Discriminator

The discriminator is a Convolutional Neural Network (CNN). Note that for classifying the style, it makes sense to consider spans of tokens. We use the negative of the gradients of discriminator in back-propagation when it passes the encoder; this way encoder learns how to ”fool” the discriminator by representing the input a style-free fashion. We employ different strategies

in updating encoder and discriminator which we will not get into here. Our code is available here.

2.3 Decoding Strategies

A major design choice regarding almost any sequence-to-sequence model is the decoding strategy used at inference time. Our model generates the output sentence token by token. At each time step t, our model proposes a distribution $\Pr[w_t|w_{1:t-1}]$ for the next token. This distribution is defined over the entire vocabulary. We must somehow choose the next token according to this distribution and then feed it to the model to produce the next token and so on. We implemented three different decoding strategies: greedy decoding, beam search, and top-k and nucleus sampling. In our experiments, we found that beam search decoding with a beam width of 10 works best, and so we used that in the evaluation phase. Next, we shall discuss the merits of each one of these methods briefly.

2.3.1 Greedy Decoding

Greedy decoding is perhaps the simplest approach one can take to generate the output sequence. In this method, at each time step we choose the most probable token according to our model's output distribution.

$$w_t = \operatorname{argmax} \Pr[w_t|w_{1:t-1}].$$

As one can imagine, the most probable token is not always the best choice, and in our experiments we found the sentences generated using greedy decoding to be of poor quality.

2.3.2 Top-k and Nucleus Sampling

Instead of choosing the most probable token, we could choose the next token by sampling from the proposed distribution. Top-k sampling tries to further improve the quality of sampled token by limiting the sampling domain to only the top k most probable tokens. Notice that greedy decoding can be thought of as a top-1 sampling. Nucleus sampling, or top-p sampling [2] is very much similar, except that it attempts to dynamically choose k. Essentially, instead of sampling only from the most likely k words, in top-p sampling we choose the smallest possible set of words whose cumulative probability exceeds the probability p, and sample from them. In our experiment, we observed that

top-p sampling generates diverse sentences, although its generated sentences generally have less quality in comparison to those generated by beam search.

[!insert image]

2.3.3 Beam Search Decoding

In this strategy, at each time step we will compose a list of K sequences, that we hope contain more optimal sequences that their greedily-generated counterparts. In fact, as was the case with out previous strategy, one can think of greedy decoding as beam search decoding with $K = 1$. The algorithm is as follows:

$sequences \leftarrow \{([style_i], 0) | 1 \leq i \leq K\}$
while ending criteria not met **do**
  $temporarysequences \leftarrow \{\}$
  for *sequence*, *sequencescore* in *sequences* **do**
    if last token of sequence is EOS **then**
      Add (*sequence*, *sequencescore*) to *temporarysequences*
      continue
    end if
    generate the distribution of the next token
    Add the top K most probable tokens
    update the scores of these K new sequences
    Add these K new sequences and their scores to *temporarysequences*
  end for
  $sequences \leftarrow K$ highest-scored sequences in *temporarysequences*
end while
return *sequences*

Note that the ending criteria for ending the while loop can be a threshold for length of generated sequences or else depend on all the sequences being finished (containing the EOS token at their end). In practice, for decoding, we are just interested in the best generated sequence of the returned sequences(that with highest cumulative score). This strategy outperformed other introduced decoding strategies.

2.4 Challenges

When we started training the model, we encountered a number of challenges. The fact that our model is relatively complicated and is composed of multiple competing components, along with its large set of hyper-parameters, make

debugging even more challenging. In particular, we identified three key difficulties that we suspect may be common to most style transfer methods that leverage adversarial ideas.

2.4.1 Overfitting

Over-fitting is a major issue across deep learning. In our model, without carefully balancing adversarial and reconstruction loss, we can easily over-fit and only learn to auto-encode the input sentence. The two main tools that we used to combat this issue were a constant λ that weights the contribution of each one of errors, and decreasing the frequency of auto-encoder updates. In the former, the total loss for the encoder is calculated as

$$\mathcal{L}_{\text{encoder}} = \mathcal{L}_{\text{rec}} + \lambda \mathcal{L}_{\text{adv}}.$$

Of course, we also utilized common regularization techniques such as drop-out, adding noise to discriminator's input, etc.

2.4.2 Exposure Bias

Sequence-to-sequence models are usually trained with teacher forcing. Instead of using the predicted output as the input at the next step, the ground truth output is used. This will significantly improve convergence rates and allow for parallelization of training procedure, resulting in much faster training.

However, teacher forcing is not without it's drawbacks. Noticeably, it causes a mismatch between the models input at training and inference time. During training we always know the previous ground truth but not during inference. Because of this, it is common to see a large gap between error rates on a held-out set evaluated with teacher forcing versus true inference. This phenomena, known as exposure bias, when combined with the imperfections of decoding mechanisms (in our case, beam search) can lead to poor performance at inference time despite the model having a low training and evaluation loss.

A variety of approaches have been proposed to mitigate the exposure bias induces by teacher forcing. We employed a variant of scheduled sampling [1]. The idea is to select the previous predicted output instead of the ground truth output with probability p. This will slow-down training as we would need to pass data through our decoder multiple times, but we empirically observed significant improvements from using it.

2.4.3 Moving Targets

There is an inherent tension in our adversarial training procedure, as the discriminator tries to distinguish the outputs of the encoder while, at the same time, the encoder is trying to fool the discriminator. If our discriminator is bad, the encoder will not receive an informative signal and will not improve either. The fact that each one of these two modules are trying to fool the other while using the other's output is reminiscent of a problem known as moving targets in reinforcement learning and Q-learning literature. Inspired by techniques such as DQN, we freezed one when trained the other. For instance, we used five consecutive mini-batches to train the discriminator, then freezed the discriminator, and used the next five mini-batches to train the encoder. This approach got us some mixed results!

Chapter 3

Results

Here, you can see some (cherry-picked!) examples from our model. Overall, the model seems to perform better when the desired label is "Happy". We also monitored the accuracy of our discriminator on a held-out test set. Ideally, we would like this accuracy to be around 50 percent, indicating that our encoder has been successful in removing stylistic elements from the input sentence. This was the case for most the models we tested, although in some cases the accuracy was higher, around 60 percent. This means that those models can still improve and we should consider using different hyper-parameters for them. Some of our training logs, which include this accuracy can be found under the logs directory in our code.

initial review	desired label	generated review
پیک خیلی بد اخلاق بود واقعا متاسفم	Happy	پیک خیلی خوب بود ممنون از پیک اسنپ فود فوق العاده راضی بودم
غذا فوق العاده بدمزه بود	Happy	غذا فوق العاده خوش مزه بود
خوب بود نون. سیر خیلی خوشمزه بود.	Sad	خوب بود نون لواش ته دیگ خوب نبود.
قارچ و پنیر بسیار کم بود	Happy	قارچ سوخاری فوق العاده خوب بود
خوشم اومد خیلی سریع آورد	Sad	دیر رسید سفارش داده بودم
کروسان اصلا تازه نبود	Happy	کروسان خوشمزه بود.

Chapter 4

Future Work

There are multiple approaches we could take to further improve the performance and efficiency of our method. First, to reduce the effects of exposure bias aside from scheduled sampling that we have used, numerous other methods have been suggested. professor teaching [3] is one such method that we would have liked to use, had we had more time.

In terms of evaluation, we could do a lot more. Firstly, a thorough and systematic human evaluation using the output of our model on a held-out test set is currently the best approach. As is argued in [6], using BLEU scores or other conventional metrics used in sequence-to-sequence modelling tasks is not ideal for style transfer, and can even be at odds with the human evaluation. Particularly in our current setting, a model that learns to auto-encode the input sentence without changing its sentiment would achieve high scores in n-gram based metrics such as BLEU, without changing the style. Having said that, creating a dataset comprised of paired positive-negative sentences and reporting BLEU scores using it could still be illuminating. Unfortunately, we did not have such a dataset and we could not create one in time.

Conducting a systematic search for hyper-parameters and using larger models (maybe even pre-trained models such as BERT) is another venture for future work. Our current models are fairly small (particularly our discriminator) and could be replaced with more sophisticated ones. Also, due to a lack of time and the large number of hyper-parameters, we did not systematically search for best hyper-parameters, rather we used our intuition to test configurations that we thought would be ideal. This of-course is not the right way to conduct research, and we would like to systematically search for them.

Finally, we would like to test our model on different tasks. We prepared another dataset using persian poems of different poets who we thought had

different writing styles. We did not test our model on this dataset, but one could imagine that our model would benefit from having to learn multiple styles as it would return a stronger loss signal. We also identified some elements of poetic style that we could use in addition to our vanilla discriminator to provide and even stronger adversarial signal. The fact that a single binary cross entropy loss is currently the only signal that our encoder receives is a serious bottleneck and having more than one style, multiple discriminatory losses and even providing a token-wise loss (which we had discussion on, but did not mention here) could help tremendously.

References

[1] Samy Bengio et al. “Scheduled sampling for sequence prediction with recurrent neural networks”. In: *Advances in neural information processing systems* 28 (2015).

[2] Ari Holtzman et al. “The curious case of neural text degeneration”. In: *arXiv preprint arXiv:1904.09751* (2019).

[3] Alex M Lamb et al. “Professor forcing: A new algorithm for training recurrent networks”. In: *Advances in neural information processing systems* 29 (2016).

[4] Rico Sennrich, Barry Haddow, and Alexandra Birch. “Neural machine translation of rare words with subword units”. In: *arXiv preprint arXiv:1508.07909* (2015).

[5] Tianxiao Shen et al. “Style transfer from non-parallel text by cross-alignment”. In: *Advances in neural information processing systems* 30 (2017).

[6] Alexey Tikhonov et al. “Style transfer for texts: to err is human, but error margins matter”. In: (2019).

[7] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).