Text Style Transfer: Dual Reinforcement Learning

Ahmet Ş. Yener Deniz Zağlı

Abstract

Today, art is one of the indispensable parts of our lives. We are all interested in at least one branch of art. There is an artist that we are inspired or influenced by these art branches that we are interested in. The part we are impressed is actually the style of these artists. The artists put forth the same content using their own style. For example, Vincent Van Gogh puts the deformed reality in the foreground with enormous brushstrokes. The spirals applied in his paintings have a golden ratio. Ludwig Van Beethoven revealed his own style with a unique style in the transition from the classical period to the romantic period. One of the representatives of realism, Fyodor Dostoevsky wrote his masterpieces in his own style. What we are trying to talk about here is that they create similar content that distinguishes artists from other artists with their own style.

Figure 1. Image Style Transfer

Disentangling of style and text has many use cases such as anonymity, realistic review bots, author style transfer with high real world impact. The case of disabundant parallel data has been a bottleneck in achievements in the area. This lack of parallel data problem has been improved upon in machine translation, especially in languages where the available data is sparse for neural network training.

1. Introduction

Text style transfer is the domain inherent to human understanding. We separate the style from the form of the object in computer vision, we can talk about a topic in formal, informal languages considering the social context we are currently in. We can deduce the author from the textual style, the words he uses [1] which is a great threat to online anonymity with increasing surveillance and effect on day to day lives. The style and content which seems intuitive to humans is hard to decipher for machines, and harder to replicate.

In this work we utilize reinforcement learning, which does not directly aim to decouple style and content as representations but as reward signals considering the loop-ack nature of textual style transfer task. Great data precedes breakthroughs in architecture and the textual data open to public increases day by day and facilitates greater research into the topic, mirroring the sharp increase in fidelity and capabilities of computer vision algorithms. The data collected by yelp by its nature is great for the aim of the work which is sentiment transfer between works. Yelp is an online forum where users can rate their experiences with a public place, mainly restaurants and also give a number, star, between 1 and 5 ranking their satisfaction. The sentiment being numerized helps the dataset immensely giving automatic sentiment classification to the comment. We take 3 stars below to be a negative sentiment and above to be a positive sentiment about the place. While there are common words in negative sentiments such as “bad”, “worse”, “bland”; not all sentiments are explicitly stated in the comment making semantic exploration of data required.

Transformers have lately been extensively used in state of the art models, beating out LSTM architectures of the past. They require bigger amounts of data but their self-attention mechanism helps them attend linguistic relations bigger than the exploration windows of the LSTM’s which degrade the longer the text being processed is. We apply transformer architectures in this seq2seq problem and explore their results compared to bilstm encoder decoder types proposed in the original paper.

While the data source lacks any non-annotated parallel data, unsupervised seq2seq especially in neural machine translation has only progressed lately with techniques such as back-translation being deployed to production. We also explore pseudo-parallel data creation and their effects on the training of models, dealing with the cold start problem of Reinforcement Learning and the inherent lack of parallel

data found in the problem aimed to be solved in this work.

2. Related Work

The area of style transfer seldom has parallel data for content in differing styles apart from hand created datasets. Nonparallelity of data has been dealt with in differing ways.

[2] Study shows that machine translation removes the style of the author, but by the nature of the translation task keeps the content intact. [3] Leverages this work by The encoding of the content is done by using another language as the latent representation of the sentence. This translation is then fed into style specific decoder that outputs the final transformed version of the original content in the required style. The fitness signal to the decoders is then provided by an style classifier which as an input gets the output of the style specific decoders ,global attention is used to help the decoders learn non-local meaning relationship existing in different languages . The discrete tokens created by the language translator are picked using softmax from the output of the bidirectional lstm at each time step. This non-differentiable softmax layer is replaced with an softmax approximation which is based on the work of [4].

[5] This work deals with the discreet nature of the tokens by cross-aligning the generator sentence with the original sentence, the encoder representation of the original example from the style is aligned by a discriminator with the generator's feed the previous layer logits generation style. In the work a min-max game is played by the encoder, and generator trying to minimize and the generator trying to maximize. The distributions of the generator sentences must match with the original data and this creates a valid constraint for the style change and content preservation. This work is generally accepted to be a benchmark in the style transfer domain. An auto-encoder is preferred as the encoder and generator over the VAE method since the latent space is inferred as apriori a normal distribution in VAE's while this means content cannot be reconstructed from the latent space, most of the work of representation the content lies in the encoder. Auto-encoders provide a rich latent space and the reconstruction error of the auto-encoder forces the latent space to carry as much information about the content of x without including style specific information. This latent space is created by two style specific encoders and are then used by two style specific generators to create content preserving but style changed sentences.

3. Model

3.1. Model

We adopt the training technique and the information flow proposed in [6] to train our network. The problem in its nature is a sequence to sequence problem [7]. We would like to have a content of one sequence but change the sentiment of the sequence to the opposite (pos-to-neg, neg-to-pos).

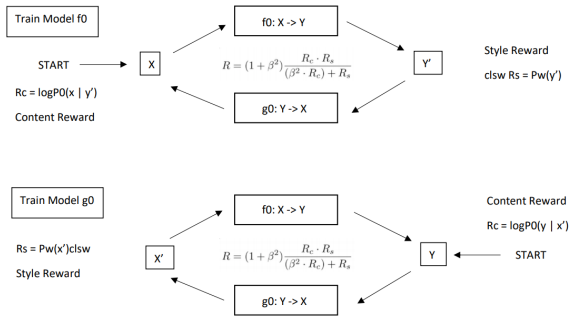

Figure 2. Dual Reinforcement Learning

The two transfer models in the baseline model are composed out of a Bidirectional LSTM [8] to encode the text, and the decoder is attention based rnn decoder.[9]

Our model uses the Transformer architecture proposed in [10]. We train word2vec vectors over the entire corpus first and get the word embeddings. Unlike the RNN's there are no explicit information about the location of the word in the relevant sentence so as in the original paper Modular mathematic coupled with the sine and cosine function is used to give positional encodings to the vectors and these vectors are input to the transformer encoder.

Multi-headed Attention in the encoder layer has what's called "Self-Attention" This mechanism allows the layer to associate any input token to any other input token. Giving weights according to the strength of the relatedness these two tokens contain. The multi headed part allows the network to learn different linguistic relations among the tokens. We feed the input vectors to 3 distinct Linear Layers giving as key,query,value values which will be denoted as k,q,v. K,q,v is split into different self attention layers which work as described.

$$q * k^t$$

(q dot k$\hat{t}$) gives us the scores where how each word should attend to other words in the input sequence. We normalize the score matrix by the

$$\sqrt[2]{dimension\_k}$$

to prevent any imbalances and for faster learning. Applying softmax over the scaled score matrix we get the probability of words association between each other. This probability matrix then multiplies the value vectors and gives the output vectors to a fully connected layer which outputs the final vector representation of the attention head layer. We then feed the attention heads outputs to a fully connected layer

giving the final vector representation of the whole multi-headed attention layer.

Transformer encoder has 6 layers of Multi-Headed Attention, added with the input vectors for the skip connection and normalized, then this normalized vectors into a feed-forward layer which also has a skip connection added and normalized. After layering 6 of these we have the encoder input representations.

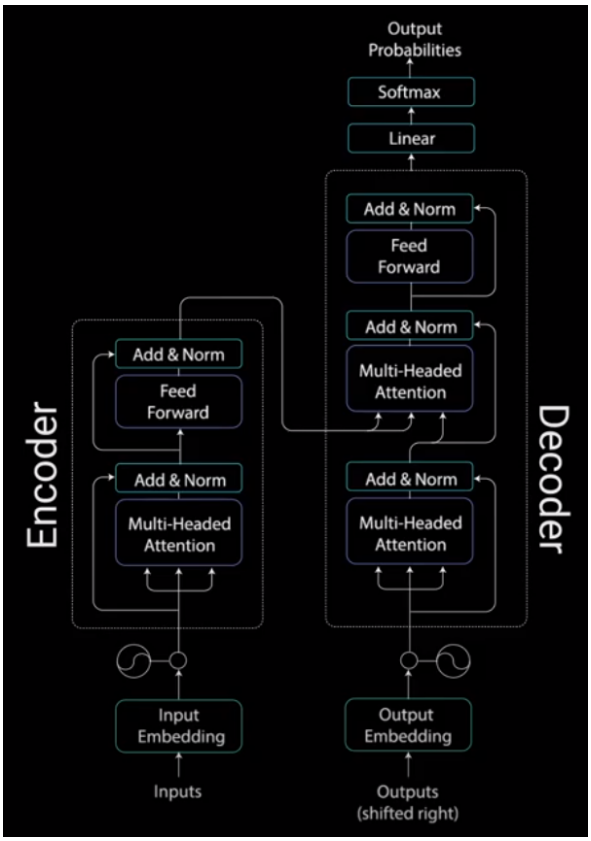

Figure 3. Transformers

Transformer decoder is auto-regressive. It takes in the previous outputs as inputs as well as the encoder outputs that contain the attention information. Decoder stops when end of string token is reached.

Transformer decoder embeddings also contain positional embedding information. This is fed into a multi-headed attention layer. However to stop the decoder from attending to the words that will be generated later we apply masking with a matrix which containts -inf in the upper diagonal. These will be turned to 0 in the softmax operation effectively stopping the decoder from attending to the words later generated. This at the later levels takes information from all encoder layers. The final output is the linear classifier fit on top of the decoder layer, which gives scores for each word in the vocab, applying softmax and picking the word with the highest probability as the output of the decoder.

3.2. Training

Two encoder-decoder couples are learned, each one mapping one sentiment to the opposite one. F model learns to translate X Y and G model learns to translate Y X Since non-human annotated parallel data is hard to come by for this task we use generated pseudo-parallel data in two ways. To give a warm start to the models we apply [11] a template based style transfer model is used to generate pretraining data for (I) X Y' ,(II) Y X'. We train the model F with (II) and model G with (II). The models are trained on these template based models to converge to a RL trainable way.

3.3. RL Training

To encourage keeping the content while the style is changed there is two distinct steps, one based on teacher-forcing which is specifically modified and a pure RL based training.

RL based training depends on the looping aspect of the task at hand where the transfer from both sides to the other side is required.

One way flow explained:

- We sample a sentence and feed it through model F

- Model F gives a style changed sentence

- Compute Reward from style classifier (I)

- Model G is fed the sentence

- The original sentence's token's probability is calculated (II)

- Harmonic mean of (I) and (II) is calculated and used as the signal to update model F

This flow is done for both models iteratively and changing in preset intervals.

3.4. Teacher Forcing

In testing time the output of the previous time is given into the decoder at the next time step as the input While in the teacher-forcing technique at the training time the ground truth sentences true token is fed into the decoder. This may result in over-exposed and brittle networks in the test time. This problem is solved by using the annealing strategy. Annealing strategy means the interval of using teacher forcing increases with more intervals effectively using it less while still reaping the benefit.

It has been noted that the quality of the template based pseudo-parallel data is not enough at later time in the training so the parallel data created by the previous checkpoint of the network has been used to update the model. As long

as the models get better at creating parallel data at later intervals of the training it has been shown to increase the fidelity of the models output.

4. Experiments & Results

4.1. Dataset

We used the dataset specially prepared by YELP for natural language proccessing studies. "yelp.com" is a platform with reviews and user reviews about businesses. There are user comments in a specially prepared dataset. These user comments are divided into positive and negative. This project is suitable for dataset style transfer as we aim to transfer the style of negative sentences to positive sentences and to transfer the style of positive sentences to negative sentences. There are 6 files in our dataset. Files with an ".0" extension contain positive sentences and sentences with ".1" extension contain negative sentences. The reason for having 6 files is that the dataset was separated from the beginning as train, dev, test. As a result, there are "train.0, train.1, dev.0, dev.1, test.0, test.1" files in the dataset. There are 384938 sentences in Train files. There are 1000 sentences in Dev files. In the test files, there are 3476 sentences in total. The number of sentences mentioned is the sentences that pass the threshold in the pre-process stage. The reason why other sentences cannot pass through the threshold is that the window, which is the linear classifier model parameter, requires a minimum of 5 words. Since one of our goals for this project is to transfer styles for long sentences, sentences containing 5 or more words will be more suitable. In the pre-process phase, we first converted all the letters in all sentences to lowercase. Then, we wrote the articles written in conjunction with the words in these sentences as separate words. For example, "re" turned into "are", "n't" turned into "note". Article is organized in this way. The "_num_" token has been added to the numbers in the sentence. Sample 3 positive and negative sentences resulting from these pre-process steps are given below:

Negative Sentences Samples:

- "walked out of this place after _num_ min of no service ."

- "maybe that is why the inside of the restaurant was so dead ."

- "drink was served in a cheap plastic cup ."

Positive Sentences Samples:

- "everything always seems to be really fresh and the prices are reasonable !"

- "staff was very attentive and friendly , and the restaurant was extremely inviting ."

- "for great service they deserve _num_ stars !"

The dataset required for the Classification section is produced with the datasets mentioned above. Using a dataset where positive and negative sentences are found separately, a new dataset is produced that combines positive and negative meaningful sentences. We create the labels required for Supervised Learning to correspond to positive and negative sentences and write these meanings in a new file. As a result, "train.txt, train_labels.txt, dev.txt, dev_labels.txt, test.txt, test_labels.txt" files are produced. Negative sentences are indicated with "neg" and positive sentences with "pos" token inside the label files.

4.2. Evaluation Metrics

We used two separate evaluation metrics in our project. The first one is Binary Accuracy, which we use for the classifier part. It is binary classification to classify the sentences in the dataset as positive or negative. For this reason, the best evaluation metric for the classifier part is Binary Accuracy.

The second one is Bleu Score. With created an n-gram vocabulary for that sentence. Then we calculate how many of these phrases in the vocabulary are in the sentence we want to calculate the BLEU score. Let's call this number we calculate x. Then we examine the sentences given as reference according to the n parameter given at the beginning. We calculate how many times the vocabulary groups in the vocabulary pass in the sentences given as reference. Let's say y to the number we get as a result of the calculation. In this case, we find the BLEU score by x / y calculation.

We use the above evaluation metrics to interpret the experimental results we made. It is the Binary Accuracy result for classifer in the numerical data we examined in the experiment results. The result for Dual Reinforcement Learning is the result of Bleu Score.

4.3. Hyperparameters

In the section Result, it is not written which hyperparameters are used for the experiments. In this section we talk about hyper parameters for the best model. Hyper parameters used for Classifier, NMT Pre-train, Dual Reinforcement Learning are discussed separately.

4.3.1. Classifier

For the Classifier part, we have experimented with both batch size and pre-trained word embedding dimension parameters. As a result of these experiments, the most suitable values are 16 batch sizes and 100 dimensions. Input dimension varies according to the vocabulary dimension. Since the vocabulary size is 9046 in the last case of the Classifier, the input dimesion value is 9046. A total of 100 filters were used. We used 3, 4, 5 as filter size. The reason we don't start the filter size from 1 is because we aim for style transfer

in long sentences. Output dimesion is 1 because the model will give a positive and negative result as a result.

4.3.2. Pre-Train NMT Models With Transformers Using Pseudo-Parallel Data

We kept the number of epoch at 5 in the pre-train phase. Validation loss and validation accuracy remain constant after epoch value 3. For this reason, we have determined the most appropriate epoch value as 5. As Batch size, we have determined 32. As the learning rate, we chose a low value of 0.001. We used bilstm as the encoder decoder type. Word embedding dimension is 300.

4.3.3. Dual-Reinforcement Learning

We used 10 epoches for Dual Reinforcement Learning. 10 epoch gave enough results to improve the results. We kept the number of epoch high and the learning rate high. For this reason, we have determined 0.00001 as the learning rate. We have set 16 as Batch size. We had to set the batch size 16 because 64 Batch did not hold on the GPU even though it gave you better results.

4.4. Results

Under the heading of Result, we primarily show our baseline results. The results of the separate topics are examined separately. Each examined results are given with the parameters used. Baseline models and models we created have worked with the same pre-process dataset.

4.4.1. Baseline Results - Classifier

This model is written in the TensorFlow library and the average run time is 2 hours and 20 minutes.

The parameters used for Classifer are given below:

- Epoch: 10

- Batch Size: 16

- Dropout: 0.5

- Filter Sizes: 1, 2, 3, 4, 5

- Learning Rate: 0.0005

With the Baseline classifier, we obtained 88.60 test accuracy and 89.00 dev accuracy using the above parameters. The graph with baseline classifier trainn losses is given in figure 4. When we examine the graph, we observe a rapid loss decrease in the first epoch. However, we observe that the loss remains constant after the first epoch. We developed our own classifier model using the PyTorch library to improve results based on baseline classifier models and parameters.

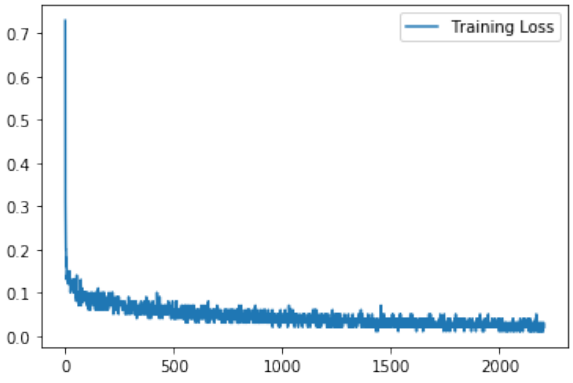

Figure 4. Baseline Classifier Train Loss Graph

4.4.2. Baseline Results - Pre-train NMT Models Using Pseudo-Parallel Data

Using this model and the parameters mentioned below, we have performed the pre-train operation of our model. The train loss and validation loss graphs calculated in this pre-train phase are given in figure 5 and figure 6. The reason for having 2 different graphics is that we perform 2 separate pre-train operations as positive-negative and negative-positive.

The parameters used for Pre-Train NMT Models Using Pseudo-Parallel Data are given below:

- Optimizer: Adam Optimizer

- Decay Type: Exponential Decay

- Decay Rate: 0.9

- Decay Steps: 10000

- Clip Gradients: 1.0

- Epoch: 5

- Batch Size: 32

- Max Iteration: 100

- Learning Rate: 0.001

- Encoder Decoder Type: Bilstm

- Encoder Units: 256

- Decoder Units: 256

- Layer size: 1

- Decode Type: Greedy

- Decode Width: 10

When we examine both graphs, we see that they are very close to each other. In both graphs, we see that the train loss and validation loss values remain constant after the second epoch. There is neither a decrease nor an increase.

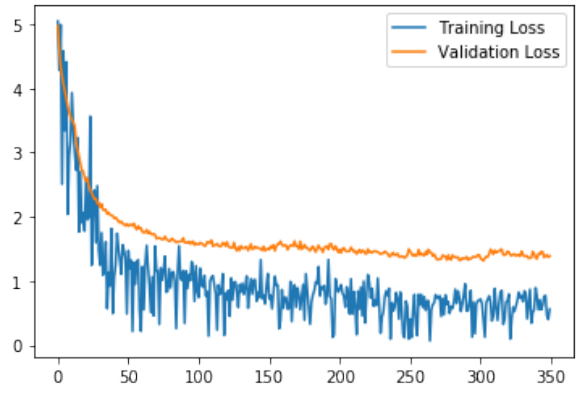

Figure 5. Baseline Negative to Positive Pre-train NMT Loss and Validation Graph

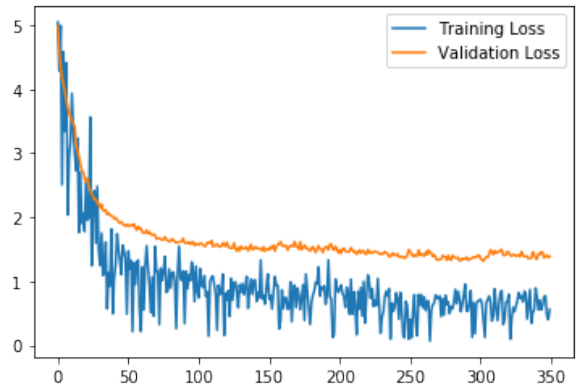

Figure 6. Baseline Positive to Negative Pre-train NMT Loss and Validation Graph

4.4.3. Results - Classifier

We created our own classifier, both to increase the run time and to apply different parameters. We developed our classifier we created in PyTorch. The run time of our newly created classifier is about 25 minutes. We achieved 97.54 in the test set and 97.10 in the validation set. Our Baseline model was using 100-size pre-trained word embedding. We developed our developed classifier with 50 dimension, 100 dimension, 200 dimension word embedding and showed the results in figure 7, figure, 8, figure 9 charts. Our baseline model batch size was 16. We tried 8, 16, 32 batch sizes and tried to find the most suitable batch size. Train loss and validation loss graphs created with different batch size are shown in figure 10, figure 11, figure 12.

The parameters used are shown below:

- Input Dimension: 9642 (Vocabulary Size)

- Filters: 100

- Filter Size: 3, 4, 5

- Output Dimension: 1

- Dropout: 0.5

Figure 7. Classifier Loss and Validation Graph - Embedding Dimension: 50

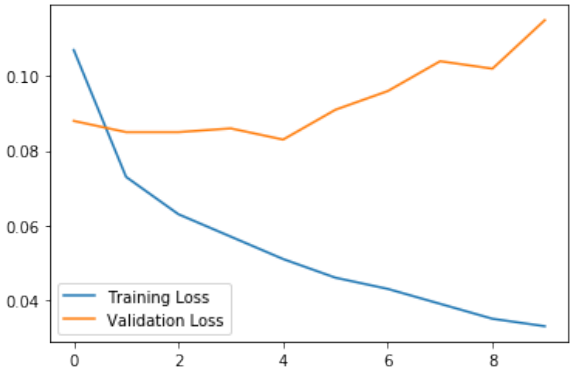

Figure 8. Classifier Loss and Validation Graph - Embedding Dimension: 100

When we examined both batch size experiments and pre-trained word embedding dimesion experiments, we determined 16 as batch size and 100 as pre-trained word embedding dimension. Compared to the Baseline model, both the runtime is significantly shortened and we see a nice improvement in test accuracy.

4.4.4. Results - Pre-train NMT Models with Transformers Using Pseudo-Parallel Data

In this section, all parameter values we use for baseline are the same. Differently, we used 6 layer transformer. Runtime duration is 18 hours for positive negative and 18 hours for negative positive. There is a 36-hour runtime in total. The train and loss graphs obtained for Pre-Trained NMT Model with Transformer Using Pseudo-Parallel Data are shown in figure x and figure y.

When we examine the results, we see that the train loss

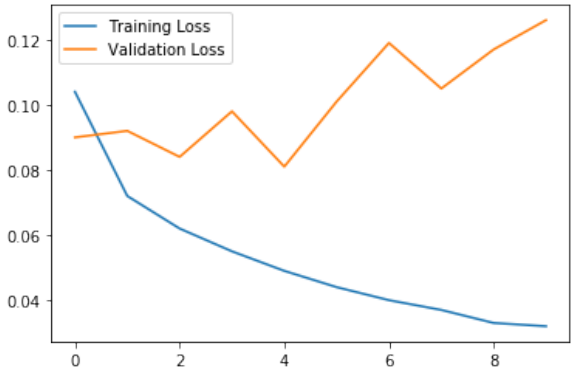

Figure 9. Classifier Loss and Validation Graph - Embedding Dimension: 200

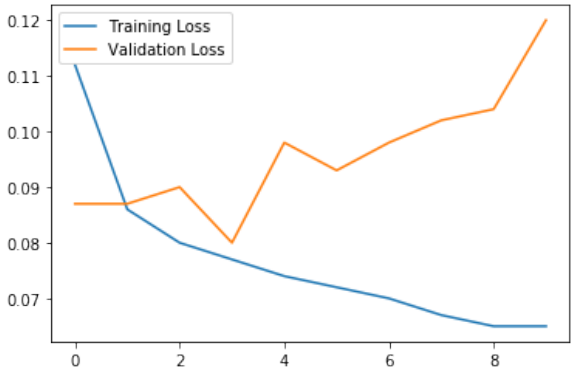

Figure 10. Classifier Loss and Validation Graph - Batch Size: 8

Figure 11. Classifier Loss and Validation Graph - Batch Size: 16

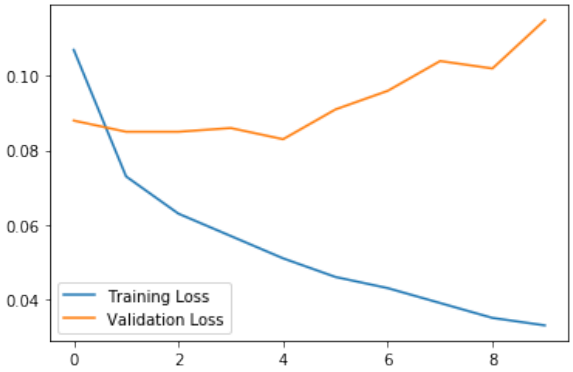

Figure 12. Classifier Loss and Validation Graph - Batch Size: 32

values are very variable. Validation loss value is fixed at a higher value than our baseline model. However, as a result of the dual reinforcement learning stage, we achieved a higher bleu score thanks to this pre-trained model.

4.4.5. Dual Reinforcement Learning

Original Sentence:

definitely disappointed that i could not use my birthday gift !

Baseline Model:

definitely happy that i could definitely use my birthday gift !

Ours:

definitely happy that i get to use my birthday gift !

Here we see an example of transformer architectures benefit over the lstm's where a fitting phrase could be picked, the attention layer of the decoder could build a better and more fluent description of the content.

Original Sentence:

we sit down and we got some really slow and lazy service .

Baseline Model:

we sit down and we got some really cool and great service .

Ours:

we sit down and we got some really fast and quick service .

While the baseline model scores better on the sentiment classifier because of the words that are frequently used in the positive sentiment class our model actually captures the relationship of antonym between slow and fast.

Original Sentence:

otherwise a great experience and we will go again .

Baseline Model:

otherwise a great experience and we will not go again .

Ours:

otherwise a terrible experience and we will go again .

Here we see our model performing worse than the baseline, our model's attention weights show us that there exists

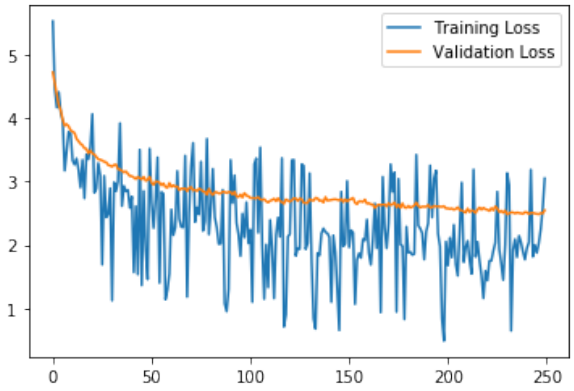

Figure 13. Classifier Loss and Validation Graph - Batch Size: 8

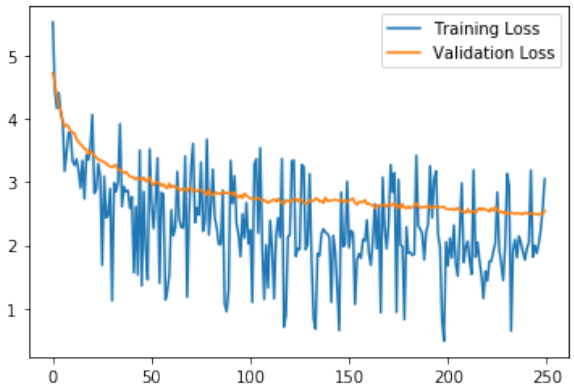

Figure 14. Classifier Loss and Validation Graph - Batch Size: 16

a highly correlated weights between again and otherwise where our model could not see that the “and” token compertimalizes the sentence and changes the meaning away from otherwise. This could be a reason from lack of training or the abundance of changing modes of speak following a “and” token in the training data set.

Baseline Bleu Score: 45.6

Our Model Bleu Score: 46.1

5. Conclusion

The weakest part of the model we created is run-time. Baseline Pre-trained NMT is trained in about 16 hours, while the pre-trained NMT in our model is trained in about 32 hours. Since the training process of Dual Reinforcement part is 8 hours, it usually takes 400 hours to train the model. We used Google Colab Pro as a cloud base. Google Colab Pro’s continuous uptime was 24 hours, so continuous model backup was required. The network-based errors experienced in this case caused the training period to be extended or to return to the last registered model. One of the important problems on the dataset side is the low number of parallel datasets. Apart from a few popular datasets, there are no other datasets in Text Style Transfer. It requires a lot of pre-process work when it is found.

References

[1] Gómez-Adorno, Helena, et al. ”Automatic authorship detection using textual patterns extracted from integrated syntactic graphs.” Sensors 16.9 (2016): 1374.

[2] Ella Rabinovich, Shachar Mirkin, Raj Nath Patel, Lucia Specia, and Shuly Wintner. 2016. Personalized machine translation: Preserving original author traits. In Proc. EACL.

[3] Style Transfer Through Back- Translation. Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, Alan W Black. ACL 2018.

[4] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In Proc. ICML, pages 1587–1596

[5] Shen, Tianxiao, et al. ”Style transfer from non-parallel text by cross-alignment.” Advances in neural information processing systems. 2017.

[6] https://export.arxiv.org/pdf/1905.10060

[7] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. ”Sequence to sequence learning with neural networks.” Advances in neural information processing systems. 2014.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Proceedings of ICLR, 2015.

[9] Vaswani, Ashish, et al. ”Attention is all you need.” Advances in neural information processing systems. 2017.

[10] Vaswani, Ashish, et al. ”Attention is all you need.” Advances in neural information processing systems. 2017.

[11] Juncen Li, Robin Jia, He He, and Percy Liang. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In Proceedings of NAACL, 2018.