11-695: AI Engineerng
Assignment 3: Unsupervised Text Style Transfer

Spring 2019

Abstract

In this assignment, you will implement a small text style transfer system in TensorFlow [Abadi et al., 2015], in Eager Execution mode. You will get familiar with training using Professor Forcing model [Lamb et al., 2016]. The starter code is provided for you, with all the data loading mechanisms and training driver implemented. Your job is to understand the starter code and implement the correct training routine for the current model, and extend it using cycle consistency loss.

1 Introduction

Style Transfer is an active research area at the moment, especially in computer vision [Karras et al., 2018, Liu et al., 2017, Ma et al., 2018, Zhu et al., 2017]. Unlike computer vision, however, there have been much less people working on text style transfer partly because text domain is discrete, and thus requiring a different approach for optimization. Nonetheless, unsupervised text style transfer has recently gleaned more attention with some remarkable pioneering work [Fu et al., 2018, Hu et al., 2017, Lample et al., 2019, Shen et al., 2017]. In this assignment, you are required to finish the implementation based on the work by Shen et al. [2017] in Eager Execution mode.

In more detail in terms of notation, given two datasets $S_1 = \{(x_1, y_1)_i\}_{i=1}^{n_1}$ and $S_2 = \{(x_2, y_2)_j\}_{j=1}^{n_2}$ where each dataset consists of multiple (x, y) pairs of text x with associated style y. Given a sentence x_1 from style y_1, the goal in text style transfer is to generate coherent text $\hat{x}_2$ that preserves the content from the original sentence x_1 but stylistically similar to sentences in style y_2, as follow:

$$(x_1, y_1) \xrightarrow{+y_2} \hat{x}_2, \tag{1.1}$$

$$(x_2, y_2) \xrightarrow{+y_1} \hat{x}_1 \tag{1.2}$$

The reconstruction loss will be the difference between two pairs: $(\hat{x}_1, x_1)$, $(\hat{x}_2, x_2)$. In this assignemnt, we only have one style which is sentiment (either positive or negative). Likewise, given a positive sentence, you are required to transfer it to a negative one with perserving most of it content. For example: given a sentence "The food is so good" ($y = 1$), the good transferred counterpart is "The food is so bad" ($y = 2$).

To achieve such objective, one solution is to use an Encoder-Decoder model based on Seq2Seq [Sutskever et al., 2014], and train it adversarially using 2 discriminators, each of which is for one style. Figure 1 illustrates this base model. For more information, you can refer to the original paper [Shen et al., 2017].

2 Code and Dataset

Install TensorFlow. If you have not already, please navigate to TensorFlow's site and follow their instructions to install the framework. The instructions are at

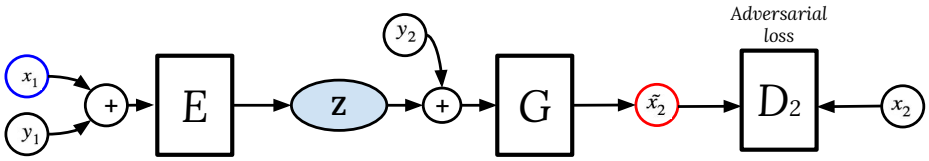

Figure 1: Text Style Transfer base model [Shen et al., 2017] includes one RNN encoder E and one RNN decoder (generator) G; both constitute a Seq2Seq translation model for text. The input type of both RNN is the concatenation of the text and style (for encoder), or the hidden variable and style (for decoder). The encoder encodes the input with its same style into a latent variable z before transferring to another text style. The discriminator D_2 tries to distinguish between the style transferred text $\tilde{x}_2$ and the ground truth text x_2. The other discriminator D_1 is omitted from this figure.

https://www.tensorflow.org/install/.

Their instructions should be sufficient to install TensorFlow and all of its dependencies on your system. It is possible to complete this assignment without using any GPU, so you do not need to install CUDA or anything related to GPU programming. However, if you wish to, you are more than encouraged to install TensorFlow GPU, which will make your implementation much faster. Please use at least the version **1.13** for Eager Execution and other related features using in the skeleton code.

Datasets. There are 2 datasets provided to you along with the code file: Yelp Review Dataset and another tiny version of it.

1. Yelp Review Dataset [Shen et al., 2017]: This dataset contains reviews separated into 2 groups: negative and positive ones. Because this dataset is unparalleled so this task is unsupervised. The dataset is located under the directory `data/yelp`, and is separated into such 2 categories for each train, val and test segments. In detail, the filename ends with 0 denotes negative reviews (*e.g.* `sentiment.train.0`) and vice versa for 1 ending.

2. Tiny Yelp Review Dataset: This dataset is just a random sampled from the dataset above, with a much smaller scale for you to easily debug your program. The filenames, consequently, start with the prefix `small`, *e.g.* `small.sentiment.dev.0`. The rest is the same.

Starter code and Datasets. Due to the storage quota limit on our website, the starter code for this assignment is instead located in our **`Resources`** section on Piazza. This writeup is, however, still located in our normal place:

http://www.cs.cmu.edu/~htpham/11695/assignment.html

After downloading and unzipping the code and the data, you can navigate to your code directories, namly `code`. Similar to the last assignments, you will see the `TODOs` and hints which suggest what you should or should not do. You will be mainly working one 2 files: `style_transfer_eager_cycle.py` and `models.py` and no need to change the other ones (but you are also encouraged to explore).

In our opinion, this assignment is more difficult than the previous ones in 2 perspectives. First, understanding the model will take a little more time for it is not as simple as in other assignments. Second, it's an unsupervised setting and there is adversarial training, which is much trickier as shown in our lectures. Despite of those challenges, this assignment requires less effort in terms of number of code lines, as long as you understand the model's components and how they interact with each other in details.

3 Requirements

The only entry point of the program is in the file `style_transfer_eager_cycle.py`. From this file, relevant modules will be called. It is your responsibility to read the code and figure out all the relevant points. We also provided some `TODOs` and hints, however, to make your tasks less challenging. The breakdown of the your work *of coding* includes 3 tasks: A, B, and C as boldfaced below.

1. Because the base model is not as straightforward, we have already implemented for you. In detail, it includes 3 main components: EncoderDecoder, Discriminator0 and Discriminator1. The encoder RNN encodes both content and content to the latent features. After that, it will drop the current style component and concatenate a new style, both of which will be fed into the generator (decoder RNN) to have the style-transferred text. Remeber, as in machine translation, we need to have a lookup procedure to decode from logits to a word in vocabulary. In this code base, we have provided to you both Greedy Search and Beam Search classes taking care of that already–you don't have to change anything in those components.

   Now to the main part, your first job for the baseline model includes 2 tasks:

   - **TASK A** Training part: Finish the training step code located in the method `train_step(model, batch, args)` which will make use of `tf.GradientTape`, as straightforward as shown in our lectures. After this step, you can begin to train this base model using the script provided, namely `small_train.sh`.
   - **TASK B** Evaluation (validation or testing) part: you are required to finish the method `transfer(model, decoder, args, vocab, data0, data1, out_path)`. This method will basically make use of the existing components, and will write the transferred texts into files, one per each. In more detail, after epoch `n`, the validation file `data/yelp/small.sentiment.dev.0` (negative) will be transferred to `output/small.sentiment.dev.epoch.0.tsf`, which is supposed to have positive sentiments. Similarly for the other sentiment.

   Hint: You can refer to our Eager Execution lecture, GAN implementation lecture, and the original code repository by the author to help you with those two tasks.

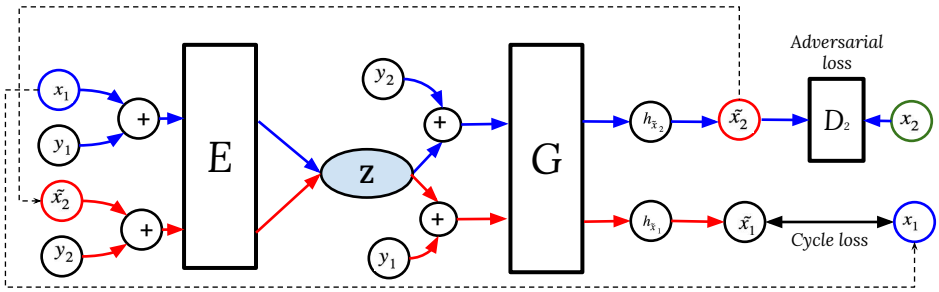

Figure 2: Cycle Text Style Transfer model uses the same Seq2Seq to transfer forward to another style, before transferring it backward to the original style. It applies the cycle consistent loss after these two-phase transfers to train our model. The notation is similar to Figure 1.

2. **TASK C** Adding the cycle phase of style transfer: inspired from the popular work CycleGAN [Zhu et al., 2017], our model can be augmented using the cycle consistency loss. In detail, it can be explained in the pair of equations below:

$$forward : (x_1, y_1) \xrightarrow{+y_2} \hat{x}_2, \tag{3.1}$$

$$backward : (\hat{x}_2, y_2) \xrightarrow{+y_1} \hat{x}_1. \tag{3.2}$$

Note the change that we keep using $\hat{x}_2$ to pass though the EncoderDecoder one more time. Likewise, the singe process in this case includes both equations above, which is for style 1; we will have the similar 2 equations for style 2. As a result, except for the reconstruction loss, we have a new (cycle) loss which is the difference $(\hat{x}_1, x_1)$, and similar for the other style. This task is more difficult and requires you to understand in more depth of the model. Figure 2 should be helful to you for understading more about this process. The original mentioned CycleGAN paper will be useful as well.

Hint: To finish this task, you will need to edit the class `EncoderDecoderCycle`, particularly and only the parents-overriding method `call(self, batch)`. For training this model, you need to provide `is_cycle true` as shown in the respective driver script `small_train_cycle.sh`

4 Assessment

You are required to train your model to achieve a reasonable performance on the full Yelp dataset. The Tiny Yelp is for debugging purpose only, and will not be enough for a reasonable performance. In particular, we will read your transferred files, namely `sentiment.test.{0,1}.tsf` and also the corresponding validation (dev) files. So make sure you submit the real results along. Except for this human evaluation, you are required to report 2 other metrics:

- BLEU score: You can run as follows for both styles (See HW2 for more reference):

  ```
  cat small.sentiment.test.1.tsf | sacrebleu ../data/yelp/small.sentiment.test.1
  cat small.sentiment.test.0.tsf | sacrebleu ../data/yelp/small.sentiment.test.0
  ```

- Sentiment Accuracy: You are provided the pre-trained classifier and the only job is to call it directly to test. The format is similar to the two scripts `sentiment_test.sh` and `sentiment_test_cycle.sh`. **Note:** you only need to change the `--test` and `--vocab` parameters only.

Those are the **baselines**:

1. BLEU score: you need to pass BLEU **5.0** for both models for non-cycle (base) and cycle models, for each sentiments. Likewise, you will have 4 BLEU scores.
2. Sentiment Accuracy: You need to pass **60.0%** for non-cycle base (slightly better than random) and **70.0%** for your cycle model.

Note: Implementation of cycle model is not straightforward, so we will give some credits if you can implement it succesfsully, regardless the results. We also give less points to this model, but strongly encourage you to finish this part.

5 Reports

As usual, you are required to report the numbers *w.r.t* Section 4 and your analysis, *e.g.* comparison between cycle and non-cycle. We will give bonus credits if you have a good quality one.

6 Gradings

The grading breakdowns are as follows with totally 110 points:

1. Pass the BLEU for the base model: **35 points**

2. Pass the BLEU for the cycle mode: **5 points**
3. Pass the Sentiment for base model: **45 points**
4. Pass the Sentiment for cycle model: **5 points**
5. Report: **10 points**
6. Extra credits: up to **10 points** . As usual, we will not limit any creative ideas from you; feel free to apply any ideas that you see fit.

7 Submission

Submit the following to our shared instructor email at: cmu.11695.sp2019@gmail.com

- Your folder of code **with results** included.
- Your real log of running each of your models. You can export log of running simply by using this syntax: `python3 style_transfer_eager_cycle.py 2>&1 | tee log.txt`.
- Your report.

Note: Your code must be runnable directly with Python3 and Tensorflow. Any extra packages might be super useful but is prohibited. If you want to use any of new packages on the internet, check with the instructors first. Furthermore, if your code has a special instruction to run, please detail it. We will give you zero for each part where your code is not runnable. The TAs will be running each of your code so please be considerate to them as well.

8 Bonus

As circulated, we give extra credits for top 3 performers for this assignments (total best BLEU + Sentiment Accuracies combined for 2 sentiments). We also reserve our right to change this bonus scheme based on your real submissions.

As usual, we urge you to start early and acquire help from instructors soon.

9 Academic Integrity.

As normal, you are encouraged to discuss with your friends and the instructors. Anything they tell you, you can use. However, looking at other's code should not happen at all cost.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. Style transfer in text: Exploration and evaluation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596, 2017.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2018.

Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609, 2016.

Guillaume Lample, Sandeep Subramanian, Eric Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. Multiple-attribute text rewriting. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1g2NhC5KQ.

Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, pages 700–708, 2017.

Shuang Ma, Jianlong Fu, Chang Wen Chen, and Tao Mei. Da-gan: Instance-level image translation by deep attention generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5657–5666, 2018.

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

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.

Revision

1. Apr 20: First version
2. Apr 22: Fixed some typos.