Natural Language Processing

Instructor - Prof Dr. G Srinivasaraghavan

Anish Rajan
IMT2018009
IIIT Bangalore

Gaurav Goel
IMT2018025
IIIT Bangalore

Problem Statement

Offensive language is the offense of using **language** in a way that could cause offense to a reasonable person.
This problem persists in many social media networking sites today where many people try to express their views or opinions. The problem is that people try to emphasize their views by using foul language. Removing all such comments or text is a fairly good solution but the content is lost in some way.
Instead, comments could be presented in a polite way such that their meaning is preserved and no one would feel offended.

Introduction

We have trained a model to transfer offensive language to non-offensive language with unsupervised text style transfer.
Text style transfer is the task of converting the style of a given sentence from one style to a target style. This has many applications such as text style transfer from positive to negative sentiment. We have particularly done the task of change a style from offensive to non-offensive language.
Unsupervised text style transfer is the task of changing from one style to another without parallel data. Parallel data from each sentence from one style to another is hard to obtain. For this reason, we have used a particular training technique proposed in Fighting Offensive Language on Social Media with Unsupervised Text Style Transfer by Cicero Nogueira dos Santos, Igor Melnyk, Inkit Padhi at IBM Research.

The important aspects of this technique are that:

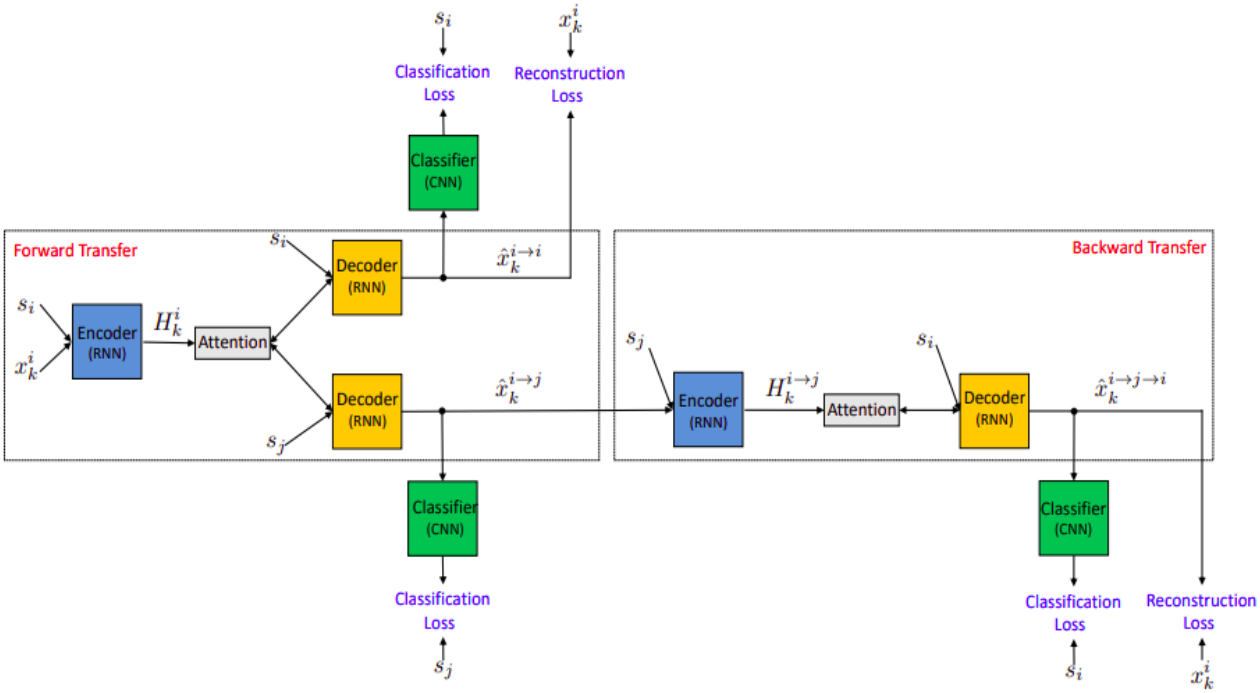
1. This is not targeted at converting from one style to another particular style.

2. This can be used to convert from any style to any other style.
3. Parallel data is not required due to the peculiar training method.
4. A classifier is also trained in the training process to detect the style.

Dataset

The researchers in the dataset used a Reddit Political Dataset and Twitter Dataset. We could not find that dataset particularly but we found very similar datasets and combined those. We used a combination of two datasets namely Hate Speech and Offensive Language Dataset and The Offensive Language Identification Dataset (OLID) datasets. The combined dataset has 35998 entries out of which 23393 entries are offensive and 12605 are non-offensive.
Preprocessing has been done by removing multiple spaces. A lot of the text had mentioned in them such as @Anish or @Gaurav. This has been removed by replacing all the mentions with @USER.
Some of the texts were Retweets and had an RT prefix before them which has been removed. Finally for the PyTorch data loader, <start> and <end> tokens have been added to the beginning and end of each text.

Model Architecture and Details of Training



As can be seen from Figure, the encoder (a GRU RNN) takes as input a sentence x together with its style label s_i and outputs H, a sequence of hidden states. The generator (also a GRU RNN) takes as input the previously computed H and a desired style label s_j and outputs a sentence $\hat{x}$, which is the original sentence but transferred from style s_i to style s_j. The hidden states H are used by the decoder in the attention mechanism (Bahdanau attention), and in general, can improve the quality of the decoded sentence.

1. When i = j, the decoded sentence $\hat{x}$ is in its original style si (top part of Figure).

2. When i != j, the decoded/transferred sentence $\hat{x}$ is in a different style sj (bottom part of Figure). Denote all transferred sentences as $\hat{X}$

3. The classifier (a CNN), then takes as input the decoded sentences and outputs a probability distribution over the style labels.

4. By using the collaborative classifier our goal is to produce a training signal that indicates the effectiveness of the current decoder on transferring a sentence to a given style.

5. The top branch of Figure can be considered as an auto-encoder and therefore we can enforce the closeness between $\hat{x}$ i→i and by using a standard cross-entropy loss.

6. However, for the bottom branch, once we transferred X to $\hat{X}$ (forward-transfer step), due to the lack of parallel data, we cannot use the same approach. For this purpose, we propose to transfer $\hat{X}$ back to X (bank transfer step) and compute the reconstruction loss between $\hat{x}$ i→j→i and xi.

7. There are 5 loss functions as indicated in the figure which are all cross-entropy loss. So, finally, we need to minimize the sum of all these losses.

Implementation Details

1. The encoder and decoder everywhere in the architecture are the same. The encoder and decoder are also of the same configurations. Both are GRU RNN's with a hidden state of size 200.

2. The embeddings for the text are an embedding layer of randomly initialized vectors of size 100.
3. The classifier is a single layer CNN with a set of filters of width 1, 2, 3 and 4, and size 128.
4. The attention is Bahdanau attention also known as additive attention.
5. Also, during the training from one style i to a different style j where i != j, I have put a **Beam search decoder** because of lack of parallel data. This will give the sentence with minimum negative log-likelihood. This is a modification of what I knew as this was nowhere mentioned in the paper.

Training Details

This model has been implemented in PyTorch with GPU as a backend for the model to train. The model has been trained for 5 epochs with a batch size of 32 and an Adam Optimizer with a learning rate of 0.0005 on Kaggle Kernel.
Teacher forcing has been used in the decoder for faster convergence of the model.

Results and Further Directions

The results were good but not very satisfactory. The accuracy of the classifier was 92%. The accuracy of the generated sentences was 67% and the content preservation score was 30%.

This summer, I am planning to modify this architecture with pre-trained LM's. I plan to use **Bert as an encoder and decoder** in this training method. Probably, the model was not able to converge well in the RNN as it was trained from scratch.

Conclusions

This paper provides a way to train on a non-parallel corpus which is very important nowadays. Our implementation would be the **first implementation** that would be pushed to Github to be open source.