Introduction

Discussing things, you care about can be difficult. The threat of abuse and harassment online means that many people stop expressing themselves and give up on seeking different opinions. Platforms struggle to effectively facilitate conversations, leading many communities to limit or completely shut down user comments. Thus, building a model to detect the negative online behaviors, like toxic comments (i.e. comments that are rude, disrespectful or otherwise likely to make someone leave a discussion) becomes meaningful.

The purpose of our project is to build a model to detect toxicity across a di- verse range of conversations, which recognizes toxicity and minimizes this type of unintended bias with respect to mentions of identities. Thus, we have built an NLP model with DNN that recognizes toxicity and minimizes this type of unintended bias with respect to mentions of identities. The dataset would be labeled for identity mentions and optimizing a metric designed to measure unintended bias.

Our model will be separated to two part:

1) Data pre-processing. In this part, we will clean the data by delete some symbols like: “ /-'?! π‘₹´”. And we will also do the word embedding by converting the words to vectors. For the word embedding part, we will use the pre-trained word vectors, GloVe which is trained by Jeffery Pennington, Richard Socher and Christopher D.Ming and FastText which is released by Facebook. Then we concatenate them to get a new word vector.

2) DNN. For the deep learning model part, we use two BLSTM as the main sequential training model and then we use two kinds of Global pooling layer to extract feature points. After pooling, we use the Res-Net to do the full connect nonlinear layer. At last, the output layer will use a sigmoid activation function

Individual work

BLSTM and other BRNN model

In this project, I first invest the possible of applying Bidirectional Recurrent Model into our problem. I came cross this model in one of the kernels. I basically solve the following problems:

- Why do we need Bidirectional model?
- Does it improve our score?
- How does it works in frameworks, what params do we have?
- Is there downside for Bidirectional model?

Starts with first question. Let's take a step a back and think why we RNN works for NLP problem. One of the reason is that RNN can "remember" the information from previous words, which is the same like how human read. But this method has its own method, one example of this problem is, suppose we have a classifier to determine every word in sentence is a name or not. Given those two sentence

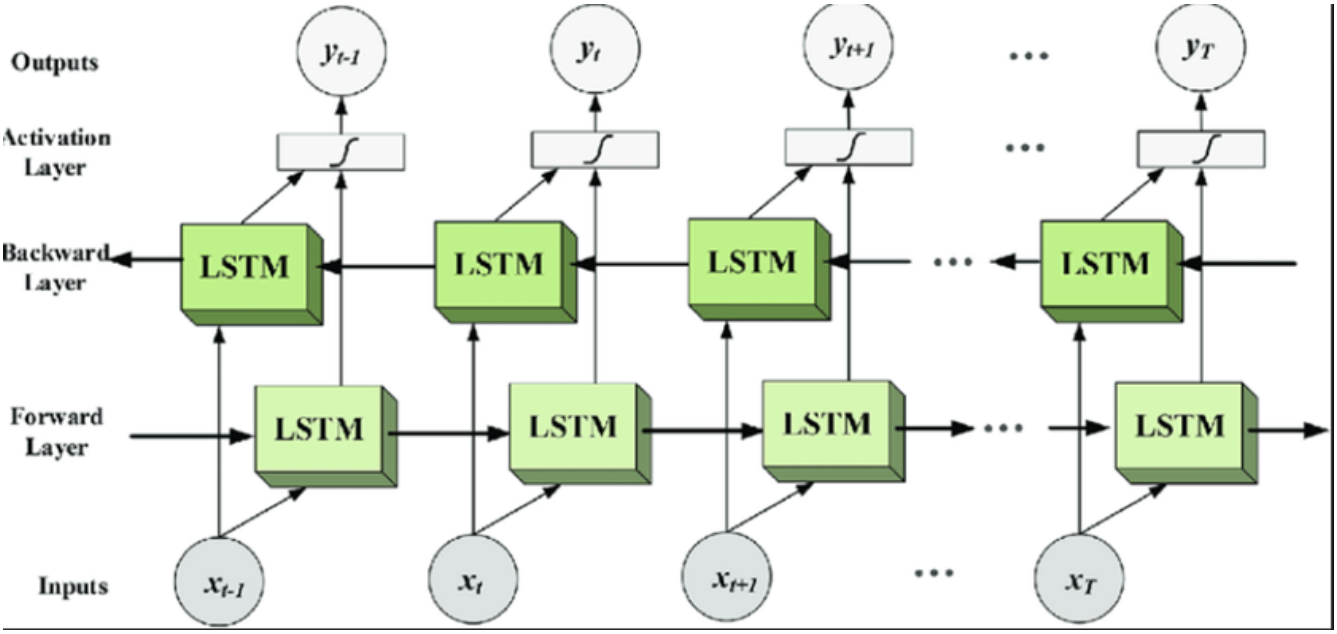
> He said, "Teddy bears are on sale!"
>
> He said, "Teddy Roosevelt was a great President"

It would be hard for an RNN to do the job, because the information for those two sentence before `Teddy` are the same. Thus, we need somehow "capture" the information after `Teddy`, That's what Bidirectional RNN can help us.

But how BRNN "capture" the information? BRNN has two RNN cells inside. One will operate on the first word of sentence, and move to the second one, ... etc. The other will operate on the last word of sentence, and move to the previous one, ... etc.

In Keras, there is a layer wrapper call Bidirectional to make a unidirectional layer to become bidirectional. It accepts the Recurrent layer(RNN, LSTM, GRU) instance as input. And there is another param to determine how will we merge the result of the two Recurrent layer. I did test over difference merge_mode and its impact over our model performance.

But to using BRNN, one thing we need is that we need whole sequence as input for the two RNN cell to read all the input before we merge them in some way and get the result.



The result of BLSTM can be expressed in such way

$$\overrightarrow{h_t} = H(W_{x\vec{h}}x^t + W_{\vec{h}\vec{h}}\vec{h}_{t+1} + b_{\vec{h}})$$

$$\overleftarrow{h_t} = H(W_{x\overleftarrow{h}}x^t + W_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{h}_{t+1} + b_{\overleftarrow{h}})$$

$$y_t = W_{\vec{h}y}\vec{h}_t \oplus W_{\overleftarrow{h}y}\overleftarrow{h}_t$$

Note that the way for us to combine the result of two LSTM cell can be different. Here we just concatenate the two vector. But we can also sum them up or do an average.

The downside of this model compared with LSTM or other RNN cells can be very simple. First, as we employ two LSTM cells in one RNN cells, the training time of the BLSTM cell should be doubled for the same situation. Also, for the LSTM cells in BLSTM to work, the whole sequence must be input, which means you can not use a word as a input in our problem, the input can only be the whole sentence.

LearningRateScheduler

LearningRateScheduler is a callback provided by keras, which main focus on adjust learning rate according to the epoch index and current learning rate. Here we use $0.001 * 0.6^{\text{epoch}}$ as our learning rate, as we can expect that the smaller learning rate will help us converge in the later epoch.

Pooling layer

We use pooling layer in our model for two reasons.

- Since the output of the BLSTM is twice of LSTM if we did not specific merge_mode, we can expect the output of the BLSTM contains lots of duplicate information, using `GlobalMaxPooling1D` and `GlobalAvgPooling1D` will help us reduce the size of tensor.
- It also helps fighting again overfitting, since it reduce the complexity of the network.

Result

1. merge_mode of BLSTM

Experiment has shown that we will get public score as `0.92836` if we sum the vector up and `0.93061` if we choose to concatenate them.

2. Number of layers of BLSTM

I also test over the influence of different layer of BLSTM, when we have 2 layer BLSTM, we have public score of `0.93010`, when we have 3 layer BLSTM, we have `0.92960`, but for 4 layer BLSTM, we have public score `0.93043`.

3. BLSTM v.s. LSTM

I also test over how BLSTM's performance differs from the LSTM's performance. In our test, we have public score of `0.93010` if we use 2 layers of BLSTM, but we will have public score of `0.92821`. However, the running time of the BLSTM version is `6983` s while LSTM version only costs us `3267` s. It is understandable as the BLSTM contains two LSTM cell, so it toke two times to train the model with BLSTM

4. Learning rate

We test over some value of the decreasing factor, when we use `0.6`, we have public score `0.93010`, but when we have `0.9`, we have public score `0.92969`

5. Output of LSTM unit

If we have 128 as the output shape of LSTM unit(which means 256 for BLSTM layer), we have public score of `0.93010`

If we have 64 as the output shape of LSTM unit(which means 128 for BLSTM layer), we have public score of `0.92308`

Clearly, the less of LSTM unit output would carry less info, thus, decreasing the performance.

Summary and conclusion

In this project, we test over some of the classical NN structure in NLP problems like LSTM, BLSTM, GRU, BGRU. We have learn, understand and experiment over the following topic:

- Relationship between Recurrent model and NLP
- Bidirectional RNN and it variant.
- Word Embedding algorithm
    - Word2vec
    - Glove
- Residual Neural Network
- Global Pooling and its application in NLP

For future improvement, we have following ideas in mind.

- Using BERT embedding, as it is the latest breakthrough made by Google recently.
- Using Attention layer to reduce the bias in the model.
- Using deeper model and better hyperparameters.
- Deal with unbalance data.

Percentage

77%

Reference

1. Yu, Z., Ramanarayanan, V., Suendermann-Oeft, D., Wang, X., Zechner, K., Chen, L., ... & Qian, Y. (2015, December). Using bidirectional lstm recurrent neural networks to learn high-level abstractions of sequential features for automated scoring of non-native spontaneous speech. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*(pp. 338-345). IEEE.
2. https://www.kaggle.com/bminixhofer/simple-lstm-pytorch-version