Toxic Comment Classification Challenge

ABDULSALAM BANDE

TABLE OF CONTENTS

NOTES

NOTE 1 : I used Pytorch For my implementation because that is what I am most comfortable with, However, I could I have used Keras but I did not because I had less time and my proposed model takes Hours to train and finetune.

NOTE 2 : You might see some bad and foul sentences in the code or in the presentation. I totally apologize, they are just meant for testing.

My new youtube Channel explaining mathematics behind Deep Learning techniques : https://www.youtube.com/channel/UCK_6ta2hzY9ly4gRRRYEYdw

Data Preparation

- Stop words like “I” that occur very frequently are removed from the text corpus.
- Text like “!”, “@” is also removed.
- A tokenizer is then used to separate words into distinct words in a sentence and also give the words indices for embedding.
- A pretrained Embedding which is fasttext is used.
- Dimension of the embedding is 300.
- A bucket iterator is also used to limit excess padding which can lead to faster performance.

Model

- In order to make the result more explainable, a directional Long Short Term Memory (lstm) that incorporates attention is used.
- Below is the pseudo code used for the attention after the lstm layer

Attention_weights = (outputs x hidden values)

Alphas = Softmax(Attention_weights)

New_hidden_values = (outputs x Alphas)

(outputs and hidden values are outputs of the bidirectional lstm)

Model

- The dimension of ***alphas*** corresponds to the length of the input text after processing plus end token (<eos>).
- So if an input sentence has 5 words. Alphas will have a dimension 6 (with the end of sentence token). And each value in the alphas represents how much attention to be paid to the input.
- Index of the largest alpha represents index of the the word to pay most attention to.
- Sum of alphas is 1 because a Softmax function is used.

Model

- I did not explain lstm because the focus here is on the attention mechanism
- But close to lstm is RNN . My Youtube video https://youtu.be/4p9SXDSwxqA explains RNN in pytorch in detail. You can check it out
- The end of the model contains a linear layer to predict 6 outputs
- The model uses dropout with probability of 0.3. You can check my Youtube video that explains Dropout https://youtu.be/Jjapw_aB6RQ

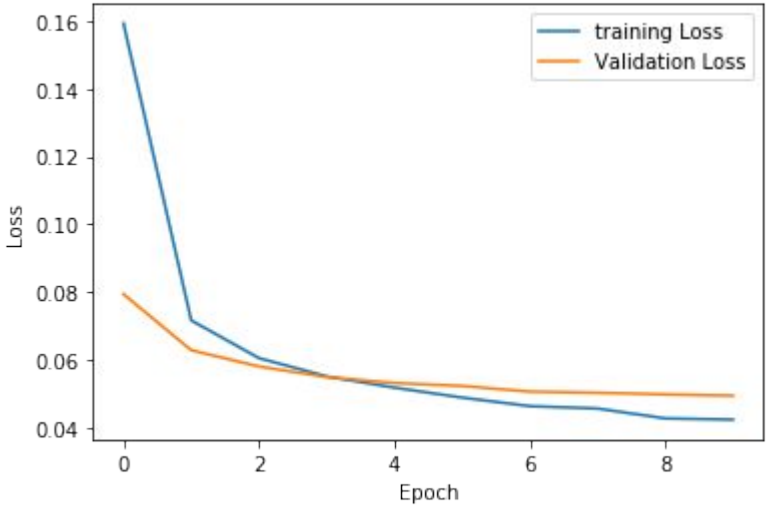
Training

- Loss function used is Binary Cross Entropy with logits

$$l_n = -w_n \left[y_n \cdot \log \sigma(x_n) + (1 - y_n) \cdot \log(1 - \sigma(x_n)) \right],$$

  - Sigma is Sigmoid
  - -w[n] represents the summation over all training examples then averaged
- The sigmoid function is in the loss function not the output layer of the model to take advantage log-sum-exp trick for numerical stability.
- Receiver Operating Characteristic (Roc) score metric used.
- Epoch is 10. Computational cost did not allow more experiments on epoch.
- Learning rate is 1e-4. And Adam Optimizer used.

Results



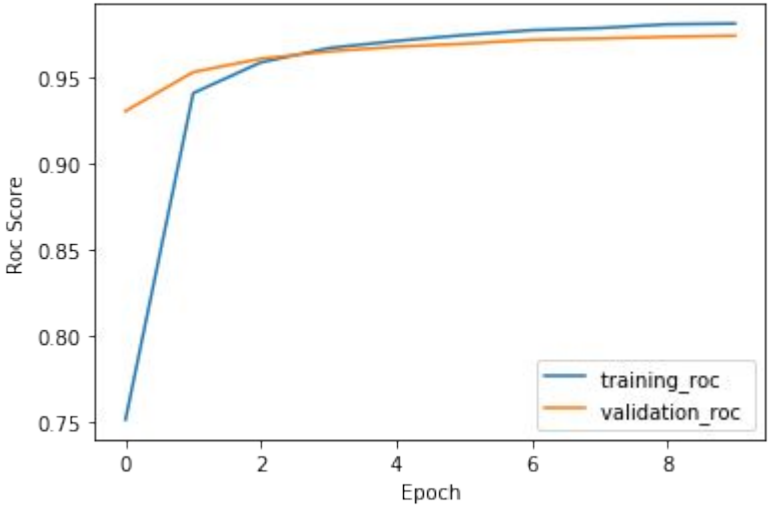


Figure on the left is the Loss Vs epoch. While Figure on the right is Roc score Vs epoch

Result Analysis

- After submitting the test set result on kaggle, I got a public score of 0.94982
- The loss drastically decreased during the first few epochs but still slowly and steadily decreased afterwards.
- After epoch 3 , the Roc score did not increased much as it was approaching 1
- The dimension of the hidden rate was also experimented on. For example hidden of the lstm 120 made the loss worst.
- Hidden dimension of 100 was the one that resulted in both the lowest loss and the highest roc score for 10 epochs.