Project Deliverable 3

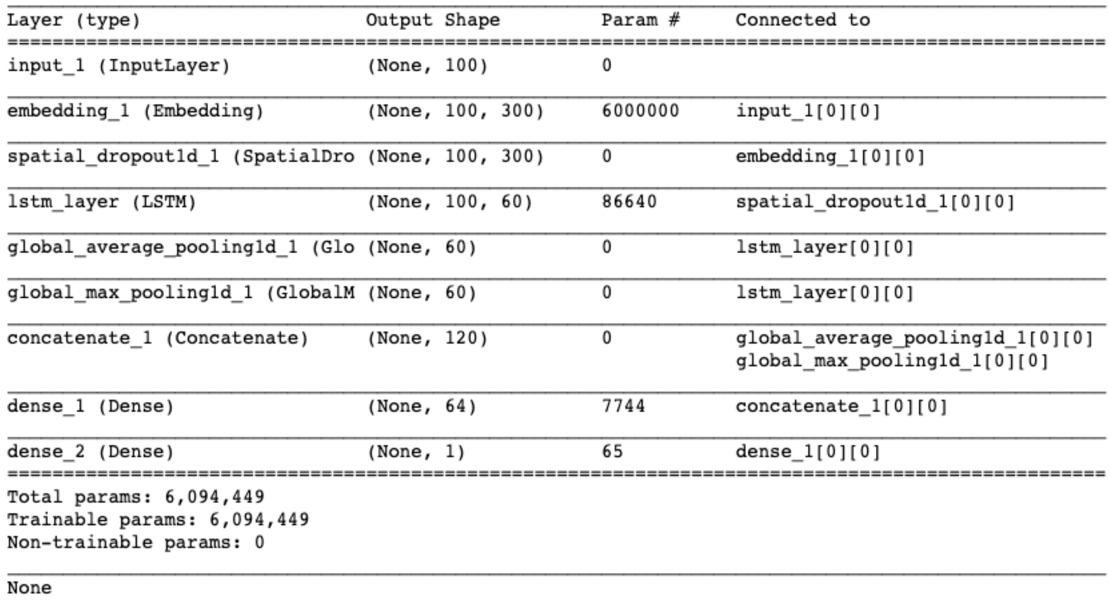
Ze Dian Xiao

March 12, 2019

§1 Final Training Results

§1.1 Preliminary Model

In my first deliverable, I have chosen to use a simple architecture of the following shape:



```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            (None, 100)          0
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 100, 300)     6000000     input_1[0][0]
__________________________________________________________________________________________________
spatial_dropout1d_1 (SpatialDro (None, 100, 300)     0           embedding_1[0][0]
__________________________________________________________________________________________________
lstm_layer (LSTM)               (None, 100, 60)      86640       spatial_dropout1d_1[0][0]
__________________________________________________________________________________________________
global_average_pooling1d_1 (Glo (None, 60)           0           lstm_layer[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_1 (GlobalM (None, 60)           0           lstm_layer[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 120)          0           global_average_pooling1d_1[0][0]
                                                                 global_max_pooling1d_1[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 64)           7744        concatenate_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 1)            65          dense_1[0][0]
==================================================================================================
Total params: 6,094,449
Trainable params: 6,094,449
Non-trainable params: 0
__________________________________________________________________________________________________
None
```

however, this didn't give me quite good results as I obtained an f1 score of 0.98 for the sincere ones, but an f1 score of only 0.55 for the insincere ones, which is exactly what we are trying to see here. The reason why we are using the f1 score is because the f1 score is the average of the precision(True Positive/Predicted) and the recall(True Positive/Actual) in terms of the insincere class. Whereas the accuracy would be a faulty measure as there is class imbalance.

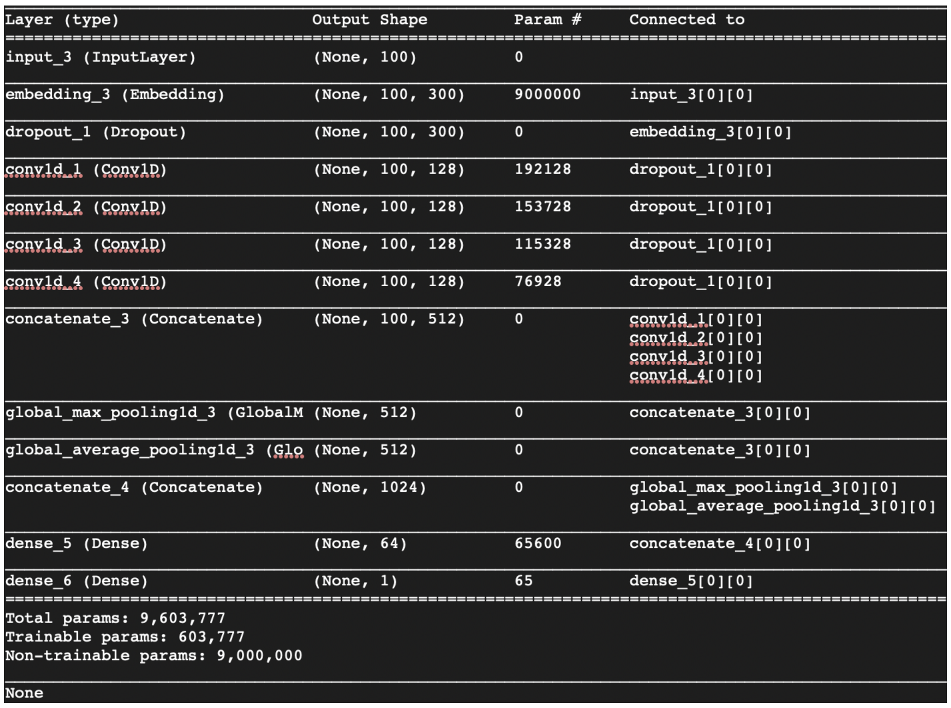
In my last deliverable, I didn't have time to properly include the different pre-trained word embeddings, hence I simply trained the word embeddings on my own data, I also used an LSTM layer since we are dealing with text data (we want to obtain the word dependencies) and finally both global average and max pooling which is another regularization method.

§1.2 Some Models I have tried

While looking for a better solutions, I have both included the GloVe embeddings, wikinews embeddings as well as the Paragram embeddings. I have tried each of those with models to obtain the accuracies of each and have decided to make an ensemble of the GloVe and Paragram embeddings. I have tried to different preprocessing by taking more vocabulary, however it didn't really change as expected since i only let some stop words remain.

§1.2.1 CNN

In one of the models I have explored, instead of using the traditional LSTM layer, I have used 4 one-dimensional convolutions which each had a different kernel size. The architecture was as follows:

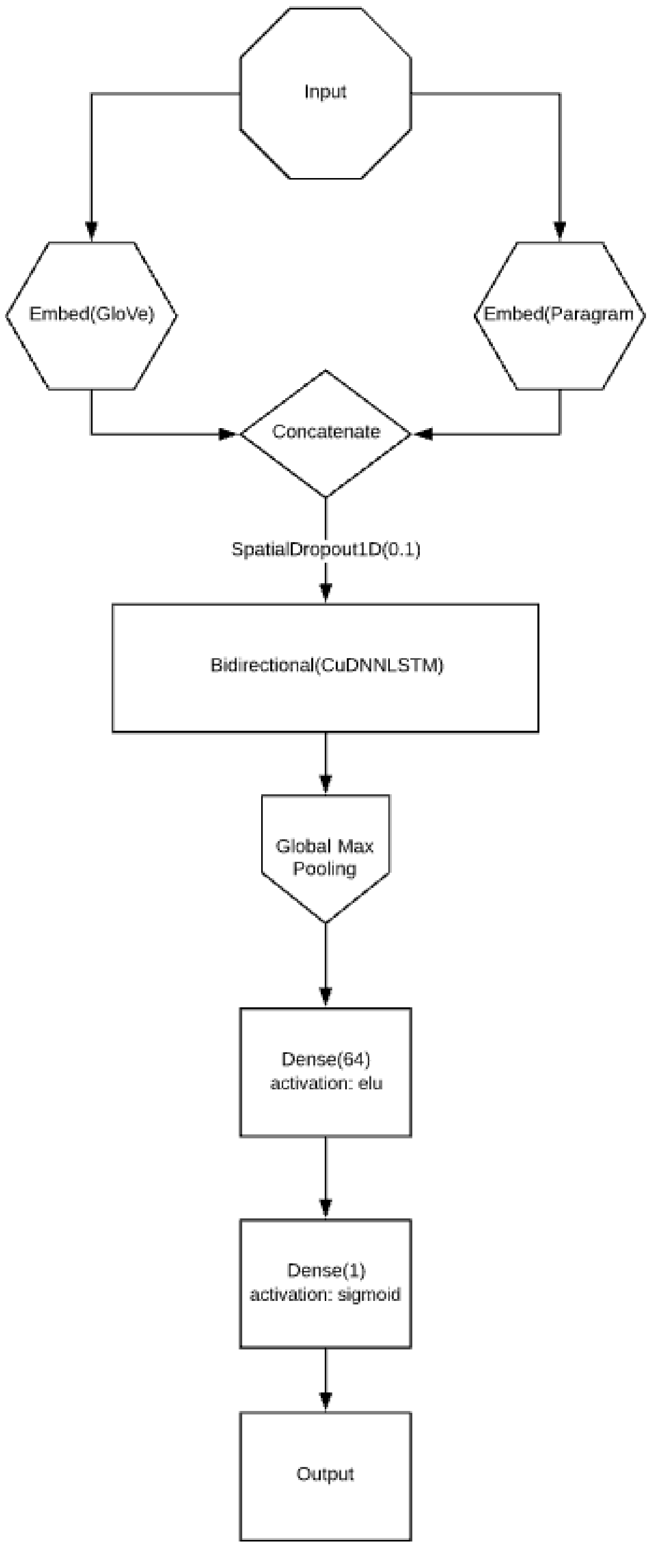


```
Layer (type)                     Output Shape         Param #     Connected to
==================================================================================================
input_3 (InputLayer)             (None, 100)          0
__________________________________________________________________________________________________
embedding_3 (Embedding)          (None, 100, 300)     9000000     input_3[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)              (None, 100, 300)     0           embedding_3[0][0]
__________________________________________________________________________________________________
conv1d_1 (Conv1D)                (None, 100, 128)     192128      dropout_1[0][0]
__________________________________________________________________________________________________
conv1d_2 (Conv1D)                (None, 100, 128)     153728      dropout_1[0][0]
__________________________________________________________________________________________________
conv1d_3 (Conv1D)                (None, 100, 128)     115328      dropout_1[0][0]
__________________________________________________________________________________________________
conv1d_4 (Conv1D)                (None, 100, 128)     76928       dropout_1[0][0]
__________________________________________________________________________________________________
concatenate_3 (Concatenate)      (None, 100, 512)     0           conv1d_1[0][0]
                                                                  conv1d_2[0][0]
                                                                  conv1d_3[0][0]
                                                                  conv1d_4[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_3 (GlobalM (None, 512)          0           concatenate_3[0][0]
__________________________________________________________________________________________________
global_average_pooling1d_3 (Glo (None, 512)          0           concatenate_3[0][0]
__________________________________________________________________________________________________
concatenate_4 (Concatenate)      (None, 1024)         0           global_max_pooling1d_3[0][0]
                                                                  global_average_pooling1d_3[0][0]
__________________________________________________________________________________________________
dense_5 (Dense)                  (None, 64)           65600       concatenate_4[0][0]
__________________________________________________________________________________________________
dense_6 (Dense)                  (None, 1)            65          dense_5[0][0]
==================================================================================================
Total params: 9,603,777
Trainable params: 603,777
Non-trainable params: 9,000,000
__________________________________________________________________________________________________
None
```

This one gave me decent results although I was only using Paragram's embeddings. I had an f1 score of 0.6, however I wasn't very satisfied and through testing a few models, while fine tuning parameters, I opted for my final model which consists of the following:

§1.2.2 Bidirectional LSTM

Here, I used both embeddings, a bidirectional LSTM layer and only global max pooling as I observed it performed better than both GMP and GAP together. Here, the most important hyper-parameters were the embedding layers, as they provide a huge jump from the initial 0.55 f1 score, the LSTM is important in this case since we are dealing with text data, and also the regularization layers. We use the binary cross-entropy loss since it is most efficient at capturing the a definite distinction between the 2 classes. I believe that although an f1 score of 0.65 seems low, it isn't that bad. What's left are the tfidf vectorizer for preprocessing, and possibly doing a k-fold validation and testing my prediction of the test data on the kaggle competition.


Input
Embed(GloVe)
Embed(Paragram
Concatenate
SpatialDropout1D(0.1)
Bidirectional(CuDNNLSTM)
Global Max Pooling
Dense(64) activation: elu
Dense(1) activation: sigmoid
Output

§2 Final Demonstration Proposal

§2.1 Application

I would like to make a web page using React or Angular, but I have no experience with either. So, if I do figure out, I'm probably going to keep it simple with a section where the user can input text which sends a request to a flask web app, converting the input using the same method as my preprocessing and using the same tokenizer.

§2.2 Poster Presentation

Here, I would like to simply outline the process of my preprocessing/ model selection and about the potential usage of the model in industry. (filtering online blogs etc.)