CS 539 Progress Report – Toxic Posts/Comments Detection

Sean Chung 9709952090

November 2018

1 Architecture of Model

1. Rather than using every word/sentence, first convert them to index of their order.
2. Since each post/comment has various length, rectify it to l words; namely, if it is shorter than l words, append 0s at the end; if it is longer, then truncate it.
3. Embedding word and have each word become its vector (word vector), with embedding size: embed_size and the vector space size: max_feature. This can help judging if words are semantically similar.
4. I mainly use Long-Short Term Memory (LSTM) model to collect each output by feeding it with word vectors (embedding result of each word).

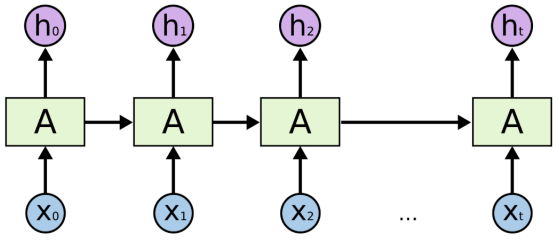

Figure 1: x_i, where $(i \in 0, 1, ..., t)$, is word vector (embedding result). A is LSTM model. h_i, where $(i \in 0, 1, ..., t)$, is corresponding result of x_i

5. Then I use these outputs as input to feed into a neural network with 6 outputs (one label for each).

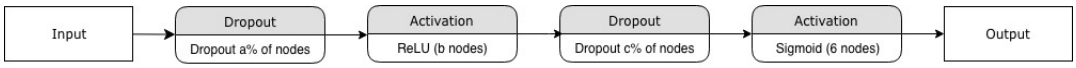

Figure 2: A brief structure of the neural network

2 Experiment Results

Here are some experiment results with different parameters described previously assignment: From the Table 1 we can see when parameters is (20000, 100, 200+,

max_feature	l	embed_size	lstm_output	a	b	c	Maximum Kaggle Score
20000	100	128	50	10	50	10	0.9775
20000	100	256	40	15	40	15	0.9719
20000	200	256	40	15	40	15	0.9617
20000	200	256	60	15	60	15	0.9644
20000	200	256	60	10	40	12	0.9615
20000	75	256	60	10	40	12	0.9657
25000	125	256	45	10	45	12	0.9668
30000	125	256	50	10	50	10	0.9650
25000	125	128	50	10	50	10	0.9663
30000	125	256	50	10	50	10	0.9650
20000	100	200	50	10	50	10	0.9773
20000	100	300	50	10	50	10	0.9780
20000	100	400	50	10	50	10	0.9732
20000	100	500	50	10	50	10	0.9770
20000	100	500	40	15	50	10	0.9758
15000	120	600	50	10	50	10	0.9731
18000	120	600	50	10	50	10	0.9762

Table 1: Results with different parameters

50, 10, 50, 10), it has higher maximum Kaggle score. However, we can also see that actually they are convergent since each result is very close to each other.

I tried different of neural networks (with more layers) and different activation functions, but they turn out to achieve lower scores.

It seems to be LSTM for this problem has reached its bottleneck. If we want to improve the score, we may need to try different models (e.g., Support Vector Machine (SVM), Naive Bayes, etc.), and ensemble them to reduce the error from any specific model.

The next step I will working on is to either develop SVM or Naive Bayes model.