Unintended Bias in Toxicity Classification

2016147005 박상재
2016147016 신우현
2020147051 장 건

1. Introduction

Recently, malicious conversation problems have intensified in online. To prevent such malicious conversations, several sites are taking steps to sanction abusive language. However, several victims came out due to the wrong judgment of the malicious conversation censorship system. To prevent such damage, the purpose of this project was to accurately classify malicious text through deep learning using bidirectional LSTM.

2. Method

2.1 Data Description

Jigsaw/Conversation AI open data was used of this project was to accurately classify malicious text in this project. This open data consists of 1.8 million train data and 97,000 test data. Train data consists of a total of 45 columns, including "id", "target", and "comment_text" but in the case of test data, it consists of only two columns id and "comment_text" except target data. As test data in open souce doesn't consist of any information of target data that is essential columns for training the model, test data was not used in this project.

id	comment_text	target
239576	It was a great show. Not a combo I'd of expected to be good together but it was.	0.0
239578	Wow, that sounds great.	0.0
id	**comment_text**	**target**
316949	Another fool pipes in.	1.0
317986	Nutter!	1.0
318641	Stupid collage students... YOU'RE NOT GOING TO EARN A LIVING MAKING COLLAGES, IDIOTS.	1.0

Fig 1. Comment_text with target

To obtain the degree of toxicity of comment, 10 annotators were asked to choose the ratings of comments. As the result, degree of toxicity was aggregated through weight in each answer and target value comes up by this calculation. As shown in Figure 1, higher value of target is considered as more toxic text and the range of target value is 0.0~1.0.

2.2 EDA

In data source, 24 columns have 1,399,744 missing values and 1 column has 778,646 missing values. These 25 columns are identity information contained in the comment. As these columns contains many missing values, these columns were useless in training the model. Therefore 25 indentity information of comments were not used in project.

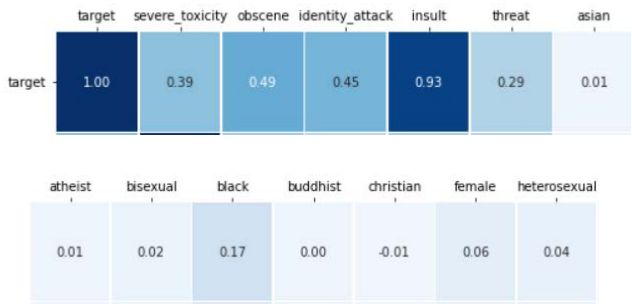

Fig 2. Correlation coefficient of target

As shown in Fig 2, toxic degree and subtype such as "severe_tosicity", "obscene", "identity_attack", insult that the correlation coefficient value was 0.3 or higher. Otherwise, correlation coefficient of other columns was so low such as 0.01. Data excluding the toxic degree and subtype were not used in

training due to low correlation coefficient. The correlation coefficient of the toxic subtypes was high, but these were also not used in training because it was not data that could be obtained. As the result, only "ID", "comment_text", and "target" columns were used in training the model.

2.3 Data Preprocessing

Fig 3. Imbalance of target data

Data preprocessing was performed to classify the toxicity of data through deep learning. Since the target data was subjectively determined, the data with '0' was removed. Then under sampling was performed because data was unbalanced[Fig 3]. Out of a total of 1,804,874 data, 288,668 data were used after preprocessing.

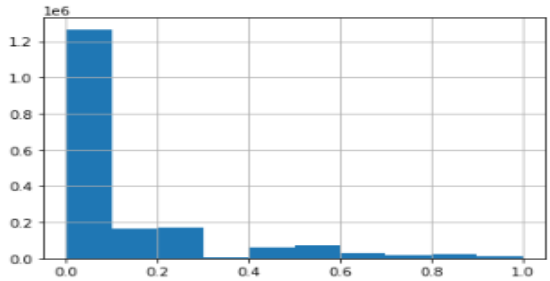

Fig 3. Imbalance of target data

To improve learning speed and accuracy, meaningless stopwords and punctuation were removed. The data form from which stop words and punctuation have been removed is shown in Fig.4

Fig 4. Remove stopwords and punctuation

The data was divided by the ratio of Train: Validation: Test=56:24:20. There were 161,654 Train data and 69,280 Validation data used for model learning, and 57,734 test data were used for performance verification. To use text data for model learning, it was converted into sequence data through Keras' Tokenizer. The process is as follows

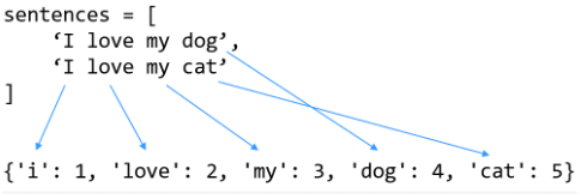

Fig 5. Text tokenizing

After tokenizing the text as shown in Figure 5, an index is assigned to each word.

Fig 6. Text to sequence

As shown in Figure 6, it is converted into sequence data using the index of each word. For smooth learning, zero padding is applied to the sequence to match the length equally.

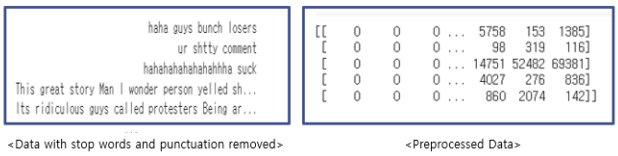

Fig 7. Preprocessed data

After preprocessing, the data was converted into sequence data as shown in Figure 7.

3. Modeling

3.1 Bidirectional LSTM

RNN and LSTM are suitable for time series prediction because they store historical data information in the hidden layer. However, there is a limitation in that the results converge based on the previous order because the inputs are made in chronological order. Bidirectional RNN is proposed to address these shortcomings. Bidirectional RNN learns through two separate cyclic neural networks, forward and reverse. Performance is generally improved by adding a reverse direction to the existing forward direction and adding it to the hidden layer.

However, Bidirectional RNN has the disadvantage of losing past information when the data is long and the layers are deep. To overcome these shortcomings, the proposed algorithm is Bidirectional LSTM. Therefore, Bidirectional LSTM was used to properly learn the interword meanings in sentences in modeling. The structure of Bidirectional LSTM is shown in Figure 8.

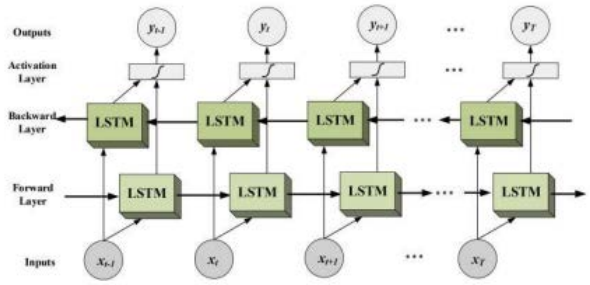

Fig 8. Structure of Bidirectional LSTM

3.2 Network Architecture

The sentence entered into the Embedding layer is converted into a vector of word units. Next, the meaning between words is learned through a separate cyclic neural network in the forward and reverse directions through the Bidirectional LSTM layer. This process is repeated twice and then passes through the GlobalMaxPooling1D layer, which is mainly used for classification, and the Global AveragePooling1D layer, which is mainly used for regression. The extracted two features are concatenated to form a fully connected network input layer. And the target value is predicted through the hidden layer and the output layer. We also use Dropout and Batch Normalization in the model to prevent the model from overfitting and stabilize the learning. The configuration of the basic model is shown in Figure 9, and the two cases of weight sum and sigmoid function were considered as the activation function of the output layer.

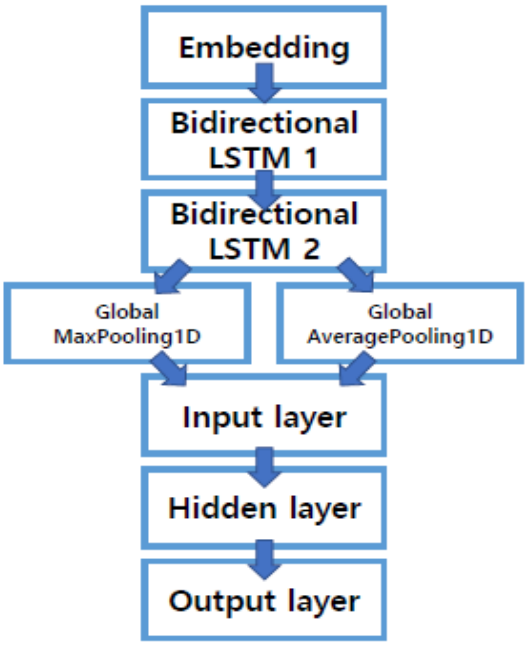

Fig 9. Base Model

The reason why the two things are considered as the activation function of the output layer is because of the characteristics of the target value.

The target value is a ratio of whether people think it is a malicious sentence, so it has a value between 0 and 1. Therefore, sigmoid function was also considered because it is possible to predict unrealistic values less than 0 or greater than 1 when predicting the target value with weighted sum.

In addition, the regression model was constructed using Adam as the optimizer of the model and MSE(mean squared error) as the loss function. In addition, if learning does not improve using Keras' callback function, overfitting is prevented by adjusting the learning rate, ending learning early, and storing weights with minimal validation loss.

3.3 Hyperparameter Tuning

Hyperparameters are as follows. Dropout rate, Batch normalization, Activation function, Learning rate, Batch size. For Hyperparameter tuning, a group of candidates was constructed by referring to the paper or commonly used values for each Hyperparameter. In addition, in order to tune one Hyperparameter, the value of the remaining Hyperparameters was arbitrarily fixed, and the value with the best learning result was selected. Figure 10 shows the results of comparing the performance of the Hyperparameter candidate group and each value.

Dropout rate	0.4	0.5	0.3	
Batch normalization	O	X		
Activation function	ReLU	Leaky ReLU	ELU	
Learning rate	0.001	0.0005	0.0001	
Batch size	256	512	1024	2048

Fig 10. Hyperparameter tuning result

4. Experiments

4-1. model1 (weighted sum)

Learning ended early in the 10th epoch. The learning process is shown in Figure 11 and Figure 12.

```
Epoch 1/20
632/632 [==============================] - 77s 109ms/step - loss: 0.1182 - val_loss: 0.0409 - lr: 1.0000e-04
Epoch 2/20
632/632 [==============================] - 68s 108ms/step - loss: 0.0366 - val_loss: 0.0304 - lr: 1.0000e-04
Epoch 3/20
632/632 [==============================] - 68s 108ms/step - loss: 0.0295 - val_loss: 0.0286 - lr: 1.0000e-04
Epoch 4/20
632/632 [==============================] - 67s 105ms/step - loss: 0.0264 - val_loss: 0.0373 - lr: 1.0000e-04
Epoch 5/20
632/632 [==============================] - 68s 108ms/step - loss: 0.0247 - val_loss: 0.0278 - lr: 1.0000e-04
Epoch 6/20
632/632 [==============================] - 66s 104ms/step - loss: 0.0234 - val_loss: 0.0333 - lr: 1.0000e-04
Epoch 7/20
632/632 [==============================] - 68s 107ms/step - loss: 0.0221 - val_loss: 0.0281 - lr: 1.0000e-04
Epoch 8/20
632/632 [==============================] - 66s 104ms/step - loss: 0.0203 - val_loss: 0.0280 - lr: 1.0000e-05
Epoch 9/20
632/632 [==============================] - 66s 105ms/step - loss: 0.0200 - val_loss: 0.0280 - lr: 1.0000e-05
Epoch 10/20
632/632 [==============================] - 66s 104ms/step - loss: 0.0198 - val_loss: 0.0280 - lr: 1.0000e-06
```

Fig 11. Training of model1

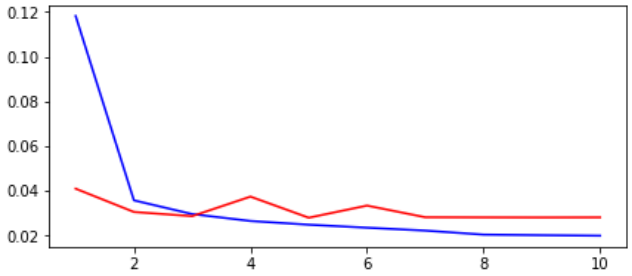

Fig 12. Train loss and validation loss of model1

After learning, the MSE for the test set was calculated to be about 0.0279. In addition, performance indicators as a classification model were calculated by setting 0.5 to threshold, assuming that it is a regression model but is used as a classification model according to predicted values and thresholds when it is actually used online. AUROC is about 0.8748 and other values are shown in Figure 13.

	precision	recall	f1-score	support
0	0.73	0.89	0.80	28890
1	0.86	0.67	0.75	28844
accuracy			0.78	57734
macro avg	0.79	0.78	0.78	57734
weighted avg	0.79	0.78	0.78	57734

Fig 13. Classification evaluation of model1

4-2. model2 (sigmoid)

Learning ended early in the 10th epoch. The learning process is shown in Figure 14 and Figure 15.

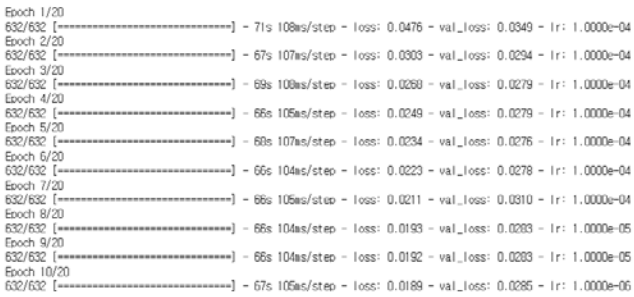

Fig 14. Training of model2

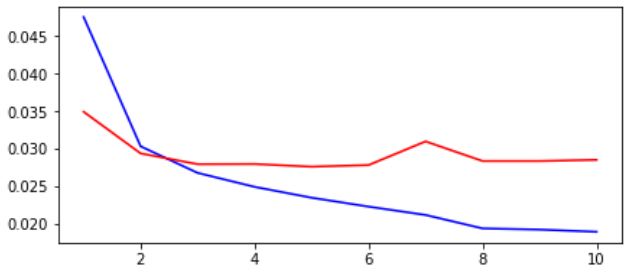

Fig 15. Train loss and validation loss of model2

After learning, the MSE for the test set was calculated to be about 0.0275. In addition, performance indicators as a classification model were calculated by setting 0.5 to threshold, assuming that it is a regression model but is used as a classification model according to predicted values and thresholds when it is actually used online. AUROC is about 0.8752 and other values are shown in Figure 16.

	precision	recall	f1-score	support
0	0.71	0.91	0.80	28890
1	0.87	0.63	0.74	28844
accuracy			0.77	57734
macro avg	0.79	0.77	0.77	57734
weighted avg	0.79	0.77	0.77	57734

Fig 16. Classification evaluation of model2

5. Coclusion

When comparing Model 1 and Model 2, it was confirmed that AUROC could be used as a classification model with good performance at about 0.875 for both models. As for the difference between the two models, MSE and AUROC are very slightly ahead of Model2, but there is little difference. However, if you look at the recall of label 0 (nontoxic), you can see that Model 2 is about 0.02 higher. This can be interpreted as a 2%p decrease in the case of misclassifying nontoxic sentences into toxic sentences. The most important thing in classifying malicious sentences is that general sentences are misclassified into malicious sentences so that there are no unfair cases. Therefore, it can be seen that it is more appropriate to use model 2 among model 1 and model 2.

Summarizing this project, we learned a regression model using deep learning techniques and confirmed that it becomes a meaningful classification model. Although only 15% (290,000 / 1,800,000) of the original dataset was used to solve the imbalance problem, the loss of information was minimized by fully using the target value held in each sentence. In addition, since we constructed a regression model that predicts target values rather than when learned from the beginning with a

binary classification model through labels, we also left open the possibility of comparing values at various thresholds.

The most important part to improve is that there is a limit to the current model as the performance of model 1 and model 2 is almost similar and the misclassified sentences are similar. Of course, it may be a problem that the target value of dataset is subjective, but if you look at the sentences that are misclassified, it seems that the meaning between words is not properly learned due to the length of the sentence being too long, proper nouns, and abbreviations. Therefore, when tokenizing a word, it is expected that the performance will improve if the meaning of the word can be contained. We also think that increasing the learning data through oversampling in the current model will be helpful.

6. Reference

[1] Jigsaw/Conversation AI, "Jigsaw Unintended Bias in Toxicity Classification" Kaggle.com. https://www.kaggle.com/competitions/jigsaw-unintended-bias-in-toxicity-classification/overview

[2] Park Ho-yeon and Kim Kyoung-jae, "Sentiment Analysis of Movie Review Using Integrated CNN-LSTM Mode", Journal of Intelligence and Information Systems, vol. 25 Issue 4, pp. 141-154, 2019.

[3] Shin Dong-Won and Lee Yeon-Soo, Jang Jung-Sun, Rim Hae-Chang, "Using CNN-LSTM for Effective Application of Dialogue Context to Emotion Classification", Annual Conference on Human and Language Technology, vol. 28 Issue 25, pp. 141-146, Oct. 2016.