Jigsaw Rate Severity of Toxic Comments

EE541 Final Project

Yusong Chen(yusongc@usc.edu), Xingchen Li(xli31296@usc.edu)

December 13, 2021

Abstract

A large number of toxic comments on the Internet make people worry about making public comments, which greatly affects the efficiency of information sharing on the Internet. In order to promote the active communication of netizens and ensure a healthy and relaxed online sharing environment, it is necessary to establish a model to automatically identify and filter highly toxic comments. At present, the popular method is to use LSTM model to classify toxic comments. In this regard, this paper will complete Jigsaw Rate Severity of Toxic Comments based on the LSTM model. The main work completed in this paper includes:

1. Analysis and visualization of data sets
2. The data is preprocessed. Adopt me. Stem. Porter. PorterStemmer () function to filter stopwords, adopting Word2Vec predict the words with the same context.
3. Toxic reviews were scored using LSTM and Bidirectional LSTM, and the two models were compared.

Keywords: LSTM, Bidirectional LSTM, toxic comments, Word2Vec

1. Background Introduction

A large number of toxic comments on the Internet make people worry about making public comments, which greatly affects the efficiency of information sharing on the Internet. In order to promote the active communication of netizens and ensure a healthy and relaxed online sharing environment, it is necessary to build a model to automatically identify and filter highly toxic comments.

When evaluating the toxicity of online reviews, we can't learn anything about the critics. Everyone has their own opinions about different reviews, so it's difficult to come up with a universal way to evaluate. In order to solve this controversial situation, or to weaken the differences brought by personalized evaluation, comparative evaluation can be adopted. Choosing two different reviews, with the experts indicating which was the more toxic review, improves the consistency of the results and facilitates observation and research. The project provided 14,000 comments, and experts picked which of the two was the more toxic. The problem our model addresses is to evaluate the toxicity of two comments, which is extreme and which is slight, and the model results should be as close as possible to the expert evaluation assessments.

2. Relative Works

Multi-label classification techniques: The model is to give these comments different labels inspired by previous competitions. Divide comments into six categories, including toxic, severe_toxic, obscene, threat, insult, and Identity_hate. Based on this idea, you can customize the weights for each category. This simply quantifies the toxicity of the comments by comparing their corresponding values. The problem with this model is that the weight of each category is also set freely by the researcher, and the controversial consideration of high and low toxicity is the main reason for the error.

Algorithm adaptation methods: [1] proposes “algorithm adaptation methods” implementing multi classification solving. This approach addresses the problem of OneVsRest and Binary Relevance, which do not consider the relevance of different labels to the experimental data. In the adaptation algorithm, it is not training on classifiers like the Classifier Chains algorithm, and it also avoids the situation of the explosion of the computation degree of Label Powerset with the increasing number of labels. This algorithm takes advantage of changes in the cost function to apply the solution of single classification problem to multi-classification.

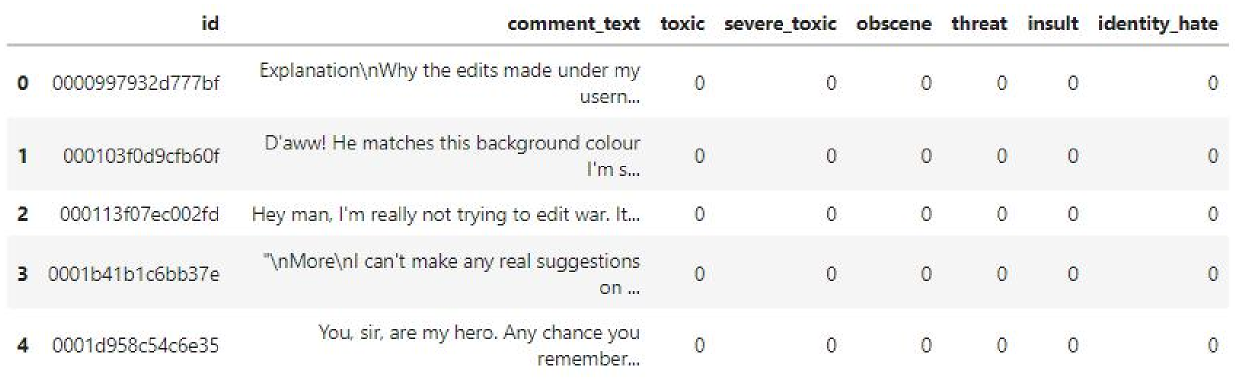
3. Implementation

3.1 Dataset and Quick EDA (Exploratory Data Analysis)

The dataset of the competition is 28.85 MB. It consists of about 14,000 comments and the corresponding ids, they are stored in comments_to_score.csv files. Each comment contains English words and punctuation marks. The goal of the competition is to predict a score that represents the relative toxic severity of the comment. Comments with a higher degree of toxicity should receive a higher numerical value compared to comments with a lower degree of toxicity; scores are relative, and not constrained to a certain range of values. Meanwhile, the competition provides validation data. validation_data.csv is pair rankings that can be used to validate models; this data includes the annotator worker id which is the same in the comments_to_score.csv file, and how that annotator ranked a given pair of comments.

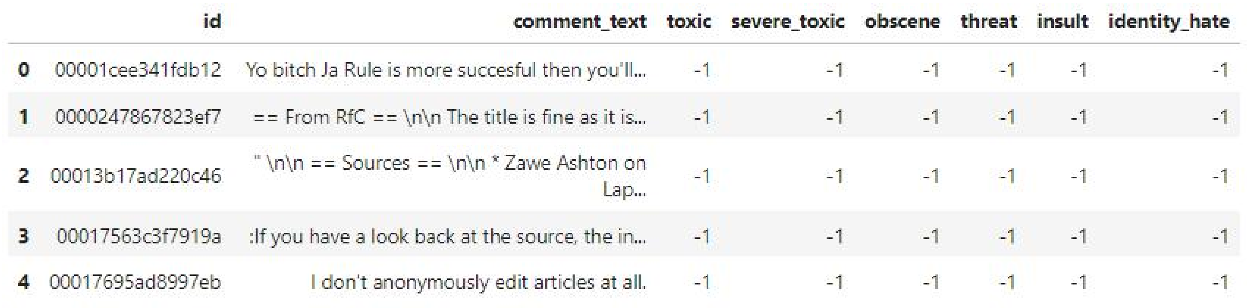
In addition, the competition data of "The problem of classification of toxic comments" from

2017 is also allowed. Including training dataset and test dataset, used for the model to classify various toxic comments. Among them, the training dataset size is 68.8MB, as shown in the train table. The test dataset is divided into test.csv and test_labels.csv, where test.csv stores id and comment_text, test_labels.csv stores the corresponding id and its category. In this project, we merge test.csv and test_labels.csv into one test data set, the size is 65.33MB, the data is shown in the Table 2: test dataset.



	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

Table 1 train dataset



	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...	-1	-1	-1	-1	-1	-1
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...	-1	-1	-1	-1	-1	-1
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...	-1	-1	-1	-1	-1	-1
3	00017563c3f7919a	:If you have a look back at the source, the in...	-1	-1	-1	-1	-1	-1
4	00017695ad8997eb	I don't anonymously edit articles at all.	-1	-1	-1	-1	-1	-1

Table 2 test dataset

Among 159,000 comments in the main database, there were about 8% of the cases that were labeled as toxic, while the remaining 92% were labeled as nontoxic. Since normally fewer comments are toxic in every general social media related data set, the distribution of classes of the data set used in this study is expected. Therefore, because the main goal here is to develop an algorithm that is more highly accurate than others using the same data set (or similarly structured data sets), this issue does not create a problem in comparing results to similar methods and algorithms. In order to assess the frequency of words size being repeated in two different categories, the overall frequency over number of words was plotted Fig.1. Interestingly, the toxic group were usually shorter than non-toxic and therefore it helped us to focus more on shorter sentences to improve algorithm

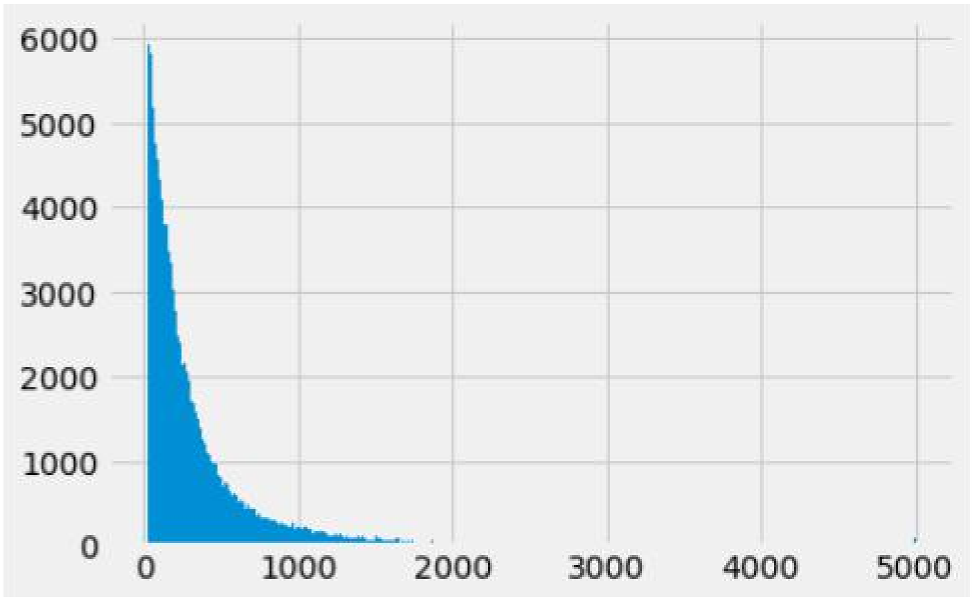
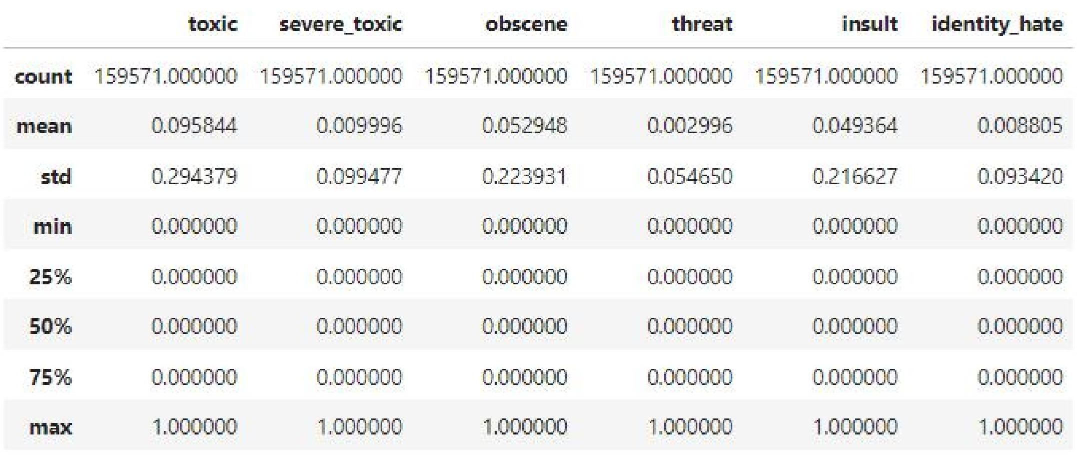


Figure 1 Histogram representation of total words in comments.



	toxic	severe_toxic	obscene	threat	insult	identity_hate
count	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000
mean	0.095844	0.009996	0.052948	0.002996	0.049364	0.008805
std	0.294379	0.099477	0.223931	0.054650	0.216627	0.093420
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Table 3 Dataset Basic Statistics

Data visualization

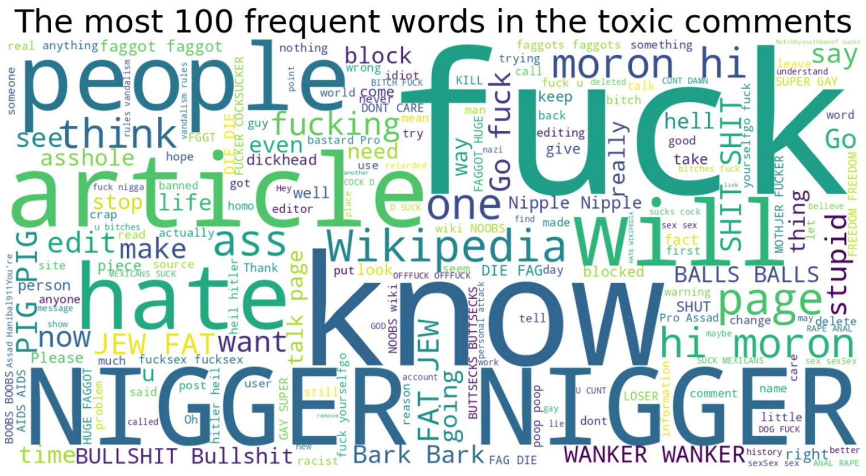


Figure 2 the most 100 frequent words in the toxic comments

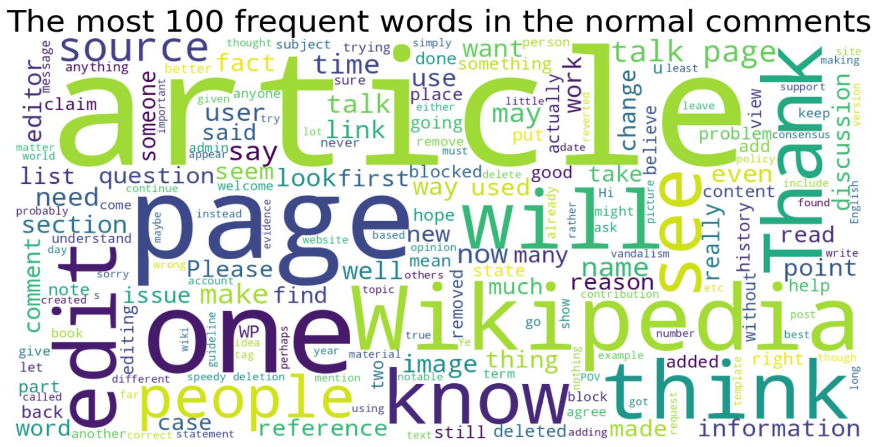


Figure 3 the most 100 frequent words in the normal comments

3.2 Text Pre-Processing

3.2.1 Cleaning and Stemming

When dealing with text, we should first do some cleaning and stemming. Cleaning and stemming is the process of removing a part of a word, or reducing a word to its stem or root. For example, there is a set of words — send, sent and sending**.** All three words are different tenses of the same root word send. So after we stem the words, we'll have just one word — send. In this section, we use the nltk.stem.porter.PorterStemmer() function.

Then, we need to filter stopwords, such as "the", "a", "an", "in". We don't want these words to take up space in our database or take up precious processing time. For this reason, we can easily delete them by storing a list of stopwords, which can be done by the nltk.corpus.stopwords() function.

3.2.2 Word Embedding (Word2Vec)

Word2Vec is a shallow word embedding model proposed by Mikolov et al. [2]. The main principle of this method is to learn law dimensional vectors from the begging. In fact, it predicts words based on their context by using one of two distinct neural models: CBOW and Skip-Gram.

Continuous bag of words (CBOW) predicts a current word based on its context. This latter corresponds to the neighboring words in the window. In the process of CBOW, three layers are used. The input layer corresponds to the context. The hidden layer corresponds to the projection of each word from the input layer into the weight matrix which is projected into the third layer which is the output layer. The final step of this model is the comparison between its output and the word itself in order to correct its representation based on the back propagation of the error gradient.

Skip-Gram is the opposite of the CBOW models. In fact, the input layer corresponds to the target word and the output layer corresponds to the context. Thus, Skip-Gram seeks the prediction of the context given a word instead of the prediction of a word given its context like CBOW. The final step of Skip-Gram is the comparison between its output and each word of the context in order to correct its representation based on the back propagation of the error gradient.

3.2 Architecture

3.2.1 LSTM

Long Short Term Memory Recurrent Neural Network [LSTM-RNN], which is a class of deep, artificial neural networks [ANN/DNN], was applied to identify the toxic comments. DNN and its applications in NLP were found to be highly capable in higher order parameters which are very vital for a specific classification [3, 4]. In this application, the higher order parameter can be a sequence of words which were labeled as a specific class. LSTM-RNN treats the comment as a set of vectored words similar to a time series trying to learn how the words are aligned in a time series attributed to a specific label [5]. The performance of the models was tested against the benchmark model. If a sentence or comment was treated like a time-series, the natural choice would use RNN (Recurrent neural network). However, for long time series and more, its applications in NLP, LSTM is highly preferred over RNN. Due to chain rule and the fact that the inverse of activation functions is usually less than one, the early layers are pruned to be treated with a very small portion of gradient descent. As a result, those early layers or deeper layers from the solution, were not trained. RNN (NLP tasks) treated the first, second, etc. words as earlier nodes as a result of Vanishing Gradient; those beginning words did not contribute as much as later words for NLP tasks. In order to solve this problem, LSTM was designed in a way to have a more capacity for remembering long term memories. The idea behind LSTM is in fact simple. Rather than each hidden node being simply a node with a single activation function, each node is a memory cell that can store other information. Specifically, it maintains its own cell state. Normal RNNs take in their previous hidden state and the current input, and output a new hidden state. An LSTM acts similarly, except it also takes in its old cell state and outputs its new cell state. LSTM has a way to control the flow of information through its hidden nodes. In other words, standard RNNs (Recurrent Neural Networks) suffer from vanishing and exploding gradient problems. LSTMs deal with these problems by introducing new gates, such as input and forget gates, which allows a better control over the gradient flow and enables a better preservation of long-range dependencies.

Our LSTM model takes a sequence of words as input. An embedding layer transforms one-hot-encoded words to dense vector representations. To process the sequence of word embedding, we use an LSTM layer with 128 units, followed by a dropout of 20%. Finally, a dense layer with a sigmoid activation makes the prediction for the multi-label classification and a dense layer with softmax activation makes the prediction for the multi-class classification.

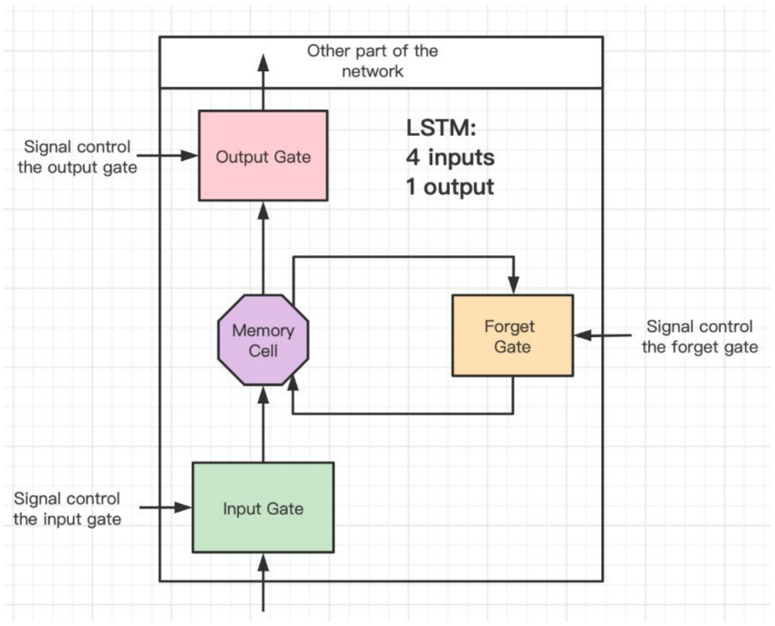

Other part of the network
Signal control the output gate
Output Gate
LSTM: 4 inputs 1 output
Memory Cell
Forget Gate
Signal control the forget gate
Signal control the input gate
Input Gate

Figure 4 LSTM

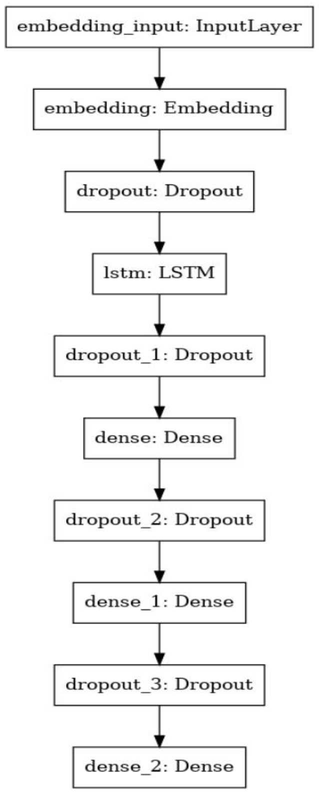


Figure 5 LSTM model architecture

3.2.2 Bidirectional LSTM

RNNs can compensate for certain errors on long range dependencies. In contrast to the standard LSTM model, the bidirectional LSTM model uses two LSTM layers that process the input sequence in opposite directions. Thereby, the input sequence is processed with correct and reverse order of words. The outputs of these two layers are averaged. We use layers with 128 units. All other parts of the neural network are inherited from our standard LSTM model. As a result, this network can recognize signals on longer sentences where neurons representing words further apart from each other in the LSTM sequence will process more likely together.

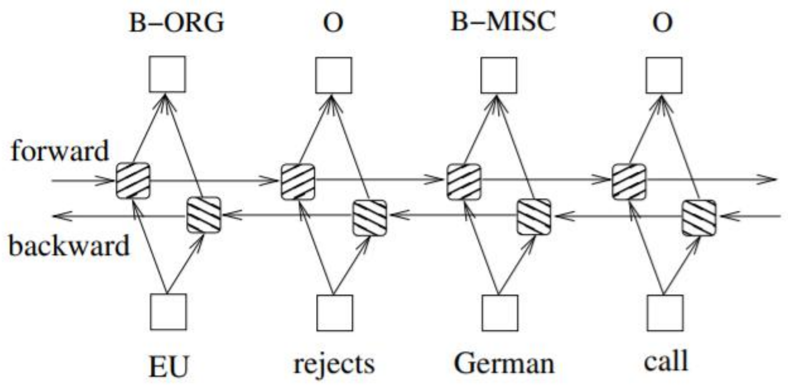


Figure 6 A bidirectional LSTM network

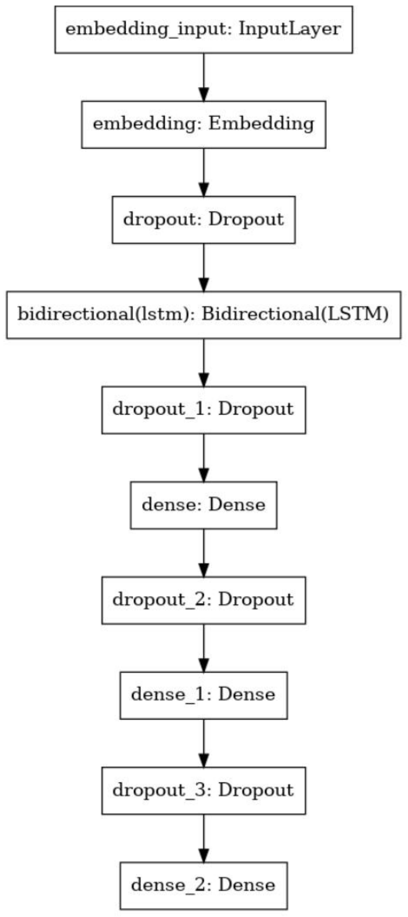


Figure 7 bidirectional LSTM architecture

3.2.3 Loss Function

We use Cross Entropy to measure how close the network output vector is to the label. The calculation formula of cross entropy is:

$$cross\_entropy = -\sum_{k=1}^{N}(p_k * logq_k) \tag{3-1}$$

Where p represents the true value, expressed in one-hot encoding; q is the predicted value, because the one-hot encoding of p only has 0 and 1, and it is a multiplication operation, so only the probability value corresponding to 1 needs to be paid attention to, and it will be ignored The probability value corresponding to 0.

3.2.4 Optimization algorithm

In order to improve the learning rate of machine learning and speed up the model convergence speed, this article chooses the Adam optimizer. Compared with the traditional first-order optimization algorithm, the stochastic gradient descent algorithm of this algorithm maintains a single learning rate, and the learning rate will not change after the weight is updated in the training process, which may make the loss function trapped in the local minimum, resulting in difficult to find the global optimal solution. The Adam optimizer designs the independent adaptive learning rate by calculating the first and second moments, and the formula is as follows:

$$m_t = \mu \times m_{t-1} + (1-\mu) \times g_t \tag{3-2}$$

$$n_t = v \times n_{t-1} + (1-v) \times g_t^2 \tag{3-3}$$

$$\hat{m}_t = \frac{m_t}{1-u^t} \tag{3-4}$$

$$\hat{n} = \frac{n_t}{1 - v^t} \tag{3-5}$$

$$\Delta\theta_t = -\frac{\hat{m}}{\sqrt{\hat{n}} + \varepsilon} \times \eta \tag{3-6}$$

Among them, m_t is the estimation of the first-order moment and the second-order moment of the gradient respectively, that is, the estimation of the expectation. $\hat{m}_t$ and $\hat{n}$ are corrections to m_t, n_t. μ is the momentum factor, η is the learning rate, and g_t is the gradient. This can be approximated as an unbiased estimate of expectations. It can be seen from this that the direct moment estimation of the gradient has no additional requirements for memory, and can be dynamically adjusted according to the gradient, and $-\frac{\hat{m}}{\sqrt{\hat{n}} + \varepsilon}$ forms a dynamic constraint on the learning rate and has a clear range.

4. Result

Due to the simple structure of LSTM network, the increase of training epoch will lead to overfitting. The overfitting problem will lead to the decrease of the generalization ability of the model, which will lead to the decrease of the identification accuracy of the model.

Our approach is to use the EarlyStopping() function to prevent overfitting. The principle of EarlyStopping() is to divide data into training sets and verification sets. After each epoch ends, test results will be obtained from the verification set. With the increase of epoch, if test error is found to increase in the verification set, training will be stopped and the weight after stopping will be taken as the final parameter of the network.

In the process of LSTM training, the Batch size was 256, and the learning rate was 0.001.

When it was around the 10th epoch, the model was obviously over-fitting. Even though loss continued to decrease, VAL_ACC began to decrease, and val_loss began to increase. The optimal VAL_ACC occurred in epoch=9, which was 0.875, and the model derived from this epoch was regarded as the optimal model.

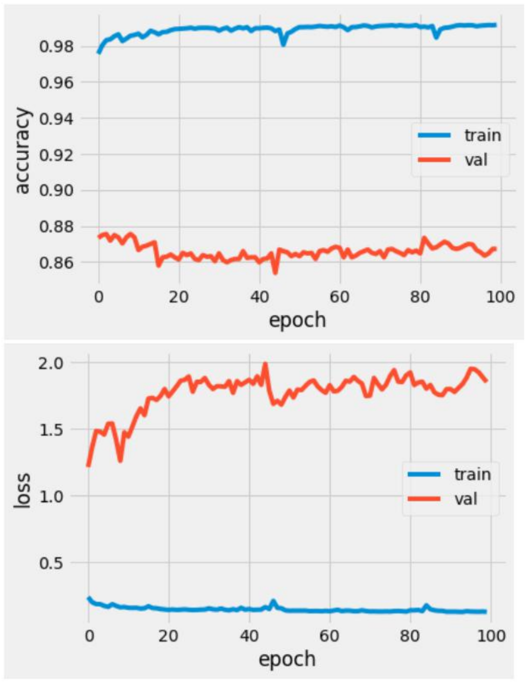


Figure 8 Training process of LSTM

In the process of Bidirectional LSTM training, the batch size was 256 and the learning rate was 0.001. Around the 5th epoch, the model produced overfitting. The Bidirectional LSTM performs better in directional training, generating optimal models faster and with better accuracy than LSTM. The optimal val_acc occurred in epoch=5, which was 0.896, and the model derived from this epoch was regarded as the optimal model.

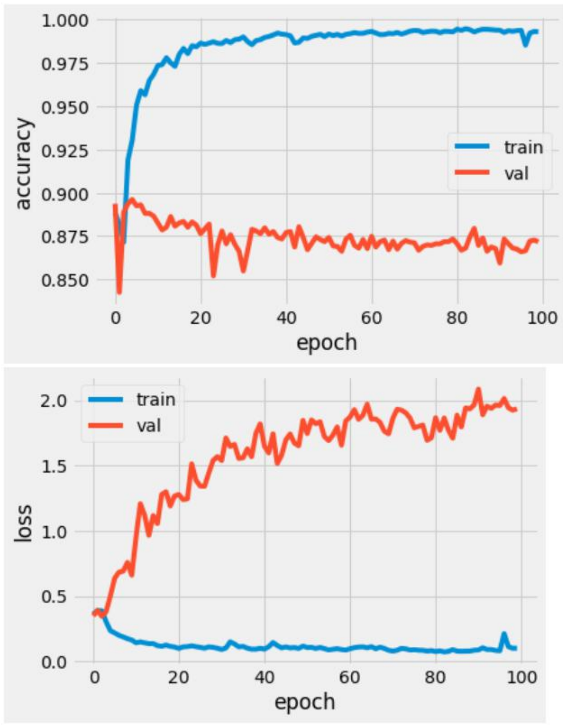


Figure 9 Training process of bidirectional LSTM

5. Conclusion

Our project aims to train an LSTM based model for rating toxic comments. As our basic model, LSTM, can basically achieve the classification and rating of toxic reviews, but the effect is not very ideal. In order to improve the score of the model, we choose bidirectional LSTM, the bidirectional LSTM model uses two LSTM layers that process the input sequence in opposite directions. As a result, this network can recognize signals on longer sentences where neurons representing words further apart from each other in the LSTM sequence will be triggered together. Our comparative experiments demonstrate that bidirectional LSTM has better results in this project. We also made some innovations in data preprocessing to improve the accuracy of the model, and on the basis of the participle, we applied Word2Vec to strengthen the connection with the same contextual words, which made it better to distinguish similar words. However, there are still defects in our model. In the future, we can use more different hyper parameters to train the model and consider adding better models.

6. Reference

[1] Nooney, K. (2019, February 12). Deep dive into multi-label classification..! (with detailed case study). Medium. Retrieved December 13, 2021, from https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff.

[2] Mikolov T, Chen K, Corrado G, Dean J. Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR, arXiv; 2013. p. 1301-3781.

[3] Nicola Michielli, U Rajendra Acharya, and Filippo Molinari. Cascaded lstm recurrent neural network for automated sleep stage classifification using single-channel eeg signals. *Computers in biology and medicine*, 106:71–81, 2019.

[4] Weicong Kong, Zhao Yang Dong, Youwei Jia, David J Hill, Yan Xu, and Yuan Zhang. Short-term residential load forecasting based on lstm recurrent neural network. *IEEE Transactions on Smart Grid*, 10(1):841–851, 2017.

[5] Rui Yan, Yiping Song, and Hua Wu. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 55–64, 2016.