CS584: DISASTER TWEETS DETECTION

Tianyi Li
Stevens Institute of Technology
tli51@stevens.edu

ABSTRACT

This is an application project of NLP to predict which Tweets are about real disasters and which ones are not. In addition to daily social interaction, Twitter can also be used to predict disasters. Therefore, this predicted information will help relevant agencies to monitor and take measures. In this project, we will use general NLP methods to pre-process the tweets text data, and then use different RNN models to predict whether it is a real disaster and try to use ways to improve the accuracy of the prediction. At the end, we will compare the effectiveness and efficiency of different models.

1 Introduction

This is an ongoing competition on Kaggle. According to reports, Twitter has become an important communication channel in times of crisis. Smartphones are ubiquitous, allowing people to announce emergencies that they are observing in real time. Therefore, more organizations are interested in programmatically monitoring Twitter (i.e. disaster relief organizations and news organizations). Therefore, such predictions become very important. In this project, the challenge is to predict which Tweets are about real disasters and which one's are not.

2 Background

Detecting twitter is challenging. Because It is always bewildering whether the words written by a person really declare a disaster.
For example, the case mentioned in this Kaggle competition: There is a tweet that says 'On plus side LOOK AT THE SKY LAST NIGHT IT WAS ABLAZE'. The author uses the word "ABLAZE" explicitly, but expresses it metaphorically. This is obvious to humans, especially with visual aids. But it is not clear to the machine.
In this competition, we're challenged to build a model that predicts which Tweets are about real disasters and which one's are not.

3 The approach

3.1 Point of view of the problem

The goal of this project is to predict whether a given tweet is about a real disaster or not. So this is a binary classification problem. The prediction result is a 0 or 1 number and the input data is the text of each tweet.

3.2 Selection of model

Among the existing solutions, using the RNN model is a good choice since RNN is more suitable for processing sequential and different length text information.
Here, we consider two ways to build our models: LSTM, GRU.
The LSTM is easier to stores long-term information and learns long-distance dependencies, so it's better than simple RNN model.
And GRU is similar , but it is quicker to compute and has fewer parameters than LSTM, because it lacks an output

gate. So we can make a comparison here.
Theoretically, compared with RNN, LSTM will have a slightly better accuracy rate, but the speed will be slightly slower. In general, the two models have similar performance. Here we make both models, and finally verify the success rate and efficiency of each model through the divided test set to choose a better model.

3.3 Metrics of evaluation

Regarding the metrics of the model, we use the F1 score because this is the metrics stipulated in this competition. The F1 score is an indicator used to measure the accuracy of a two-class classification model. It takes into account the precision and recall of the classification model.
On the other hand, although efficiency of the model is not a requirement for the competition, we can also use the time of training to compare the differences between models.

3.4 Data analysis and pre-processing

Before training the data, we also need to analyze the data first, so that we can better pre-process it. So at this stage, I used visualization processing to analyze the data, and remove the stop words and symbols to clean the data.
In addition, in order to enable the data to be effectively input to the model, I also completed tokenization and padding after data pre-processing.
Finally, for a more effective model, we processed the word vector of the input data, and used the pre-trained GloVe vector downloaded from the Internet to assign the word and create word embedding as input.

4 Experimental design

Language: Python
Software: Jupyter Notebook
Software library: keras, sklearn, tqdm, matplotlib, numpy, pandas, pandas profiling

4.1 Data sets

We can get tree files from the website. The train file, test file and sample submission file.
The size of training set: 7613 lines and 5 attributes.
The size of test set: 3263 lines and 4 attributes.
And Each sample in the train and test set has the following information:

- The text of a tweet
- A keyword from that tweet (although this may be blank)
- The location the tweet was sent from (may also be blank)

The meaning of each column names in the data set:

- id - a unique identifier for each tweet
- text - the text of the tweet
- location - the location the tweet was sent from (may be blank)
- keyword - a particular keyword from the tweet (may be blank)
- target - in train.csv only, this denotes whether a tweet is about a real disaster (1) or not

Regarding the submission file:

- 1 represents a real disaster
- 0 represents not a real disaster

4.2 Data Analysis

- Finding and deleting missing values
- Analysing length of words
- Data visualization by using library pandas profiling

4.3 Preprocessing the data

- Removing Common stopwords
- Removing urls
- Removing HTML tags
- Removing punctuations

4.4 Splitting data set

We divide the data set into train, test and validation. The validation set can be used to adjust the hyper-parameters later. And the test set can be used to verify the accuracy.
After Splitting data set, we have 4872 lines in train, 1523 lines in test and 1218 lines in validation.

4.5 Tokenization and Padding

Software library: keras.Tokenizer
Tokenization turns each word into an unique number.
In padding, we set the alignment length to 15.

4.6 Using embeddings

Before building the model we use pre-trained GloVe model to represent our words.
And it has many versions. We will try this verson here: glove.6B.100d.txt, which has 6 billion token and 100 features and create a embedding matrix.

4.7 Modeling

We consider two ways to build our models: LSTM, GRU

- For LSTM:
  software: keras
  setting the dropout layer to prevent overfitting
  loss function: binary cross-entropy
  activation function: sigmoid
  training epoch: 8
  optimizer: adam
  structure: 1 embedding layer, 1 LSTM layer, 1 dense layer
- For GRU:
  software: keras
  setting the dropout layer to prevent overfitting
  loss function: binary cross-entropy
  activation function: sigmoid
  training epoch: 8
  optimizer: adam
  structure: 1 embedding layer, 1 GRU layer, 1 dense layer

4.8 Scores And Comparing

Software: Sklearn.metrics.classification report
Metrics: Results are evaluated using F1 between the predicted and expected answers.
F1 is calculated as follows:
$F1 = 2 * \frac{precision * recall}{precision + recall}$
where:
$precision = \frac{TP}{TP+FP}$
$recall = 2 * \frac{TP}{TP+FN}$

4.9 Submission

We get the predict labels from the test set by using LSTM model.
And, we save the result as a csv file and submit it.

5 Experimental results

- Finding and deleting missing values
  By data analysis, we found Both training and test set have missing values in keyword and location. However, text and target don't have any missing values, so we don't need to deal with them here. Because currently I only use the text and target.

- Loss, accuracy and running time in each epoch in LSTM model
  Epoch 1/8 3s 14ms/step - loss: 0.6218 - accuracy: 0.6376 - val loss: 0.4994 - val accuracy: 0.7660
  Epoch 2/8 1s 11ms/step - loss: 0.4949 - accuracy: 0.7664 - val loss: 0.5281 - val accuracy: 0.7668
  Epoch 3/8 1s 11ms/step - loss: 0.5014 - accuracy: 0.7627 - val loss: 0.4937 - val accuracy: 0.7709
  Epoch 4/8 1s 11ms/step - loss: 0.4851 - accuracy: 0.7696 - val loss: 0.4836 - val accuracy: 0.7759
  Epoch 5/8 1s 11ms/step - loss: 0.4692 - accuracy: 0.7845 - val loss: 0.4966 - val accuracy: 0.7800
  Epoch 6/8 1s 11ms/step - loss: 0.4805 - accuracy: 0.7789 - val loss: 0.4817 - val accuracy: 0.7775
  Epoch 7/8 1s 11ms/step - loss: 0.4636 - accuracy: 0.7902 - val loss: 0.4793 - val accuracy: 0.7841
  Epoch 8/8 1s 11ms/step - loss: 0.4502 - accuracy: 0.7922 - val loss: 0.4666 - val accuracy: 0.7874

- Loss, accuracy and running time in each epoch in GRU model
  Epoch 1/8 3s 15ms/step - loss: 0.6524 - accuracy: 0.6063 - val loss: 0.5191 - val accuracy: 0.7553
  Epoch 2/8 1s 11ms/step - loss: 0.5322 - accuracy: 0.7512 - val loss: 0.4794 - val accuracy: 0.7800
  Epoch 3/8 1s 11ms/step - loss: 0.4836 - accuracy: 0.7813 - val loss: 0.4795 - val accuracy: 0.7759
  Epoch 4/8 1s 11ms/step - loss: 0.4987 - accuracy: 0.7599 - val loss: 0.4780 - val accuracy: 0.7726
  Epoch 5/8 1s 11ms/step - loss: 0.4802 - accuracy: 0.7851 - val loss: 0.4749 - val accuracy: 0.7783
  Epoch 6/8 1s 11ms/step - loss: 0.4781 - accuracy: 0.7812 - val loss: 0.4813 - val accuracy: 0.7816
  Epoch 7/8 1s 11ms/step - loss: 0.4635 - accuracy: 0.7843 - val loss: 0.4890 - val accuracy: 0.7775
  Epoch 8/8 1s 11ms/step - loss: 0.4646 - accuracy: 0.7899 - val loss: 0.4770 - val accuracy: 0.7783

- F1 score of LSTM Model

	precision	recall	f1-score	support
0	0.86	0.80	0.83	931
1	0.72	0.79	0.75	592
accuracy			0.80	1523
macro avg	0.79	0.80	0.79	1523
weighted avg	0.80	0.80	0.80	1523

- F1 score of GRU Model

	precision	recall	f1-score	support
0	0.88	0.79	0.84	968
1	0.69	0.82	0.75	555
accuracy			0.80	1523
macro avg	0.79	0.80	0.79	1523
weighted avg	0.81	0.80	0.80	1523

6 Analysis of the results

For F1 score, a larger value means better model performance
We can also see the recall, precision score for each category in the test set and these two model has similar performance. Through the loss display of each epoch, it can be found that the loss value of the two models will hardly become smaller after a few epochs, and the prediction accuracy has not been greatly improved. So this model does not need too much training, too much training will cause overfitting. And we can see that the training time of each epoch is very short.

The F1 scores of both models are around 0.8, it's not too bad so they have almost the same performance.

7 Conclusion and future work

7.1 Conclusion

The F1 scores of both models are around 0.8.
The LSTM and GRU model have almost the same performance in terms of F1 score.
As for training time, both have only a short time.
This result is not too bad. We can use the results of the LSTM model to generate the prediction results. We expect to have an 0.8 accuracy to predict whether the tweet indicates a real disaster.

7.2 Future work

- Using BERT model
  Generally, BERT will have better performance. We can try to use this model to improve the effect later.
- Using Cross-validation
  In this project, we found that the entire data set is not large, only nearly 10,000 pieces of data. So if we use cross-validation, we can use more data more efficiently to improve accuracy.
- Tuning hyper-parameters
  In the above implementation, we still lack some hyper-parameter settings. If the adjustment is accurate, I believe it will be more effective to make the effect even further.
- Using FastText
  FastText is another word vector model, we can compare its effect with the GloVe used in the project, and choose a better one.

References

https://www.kaggle.com/c/nlp-getting-started/overview