Assignment 2

{DDA4220} Assignment Report

Huihan YANG (120090438)
SME
Chinese University of Hong Kong, Shenzhen
huihanyang@cuhk.edu.cn

1 Introduction

1.1 Problem

Twitter is an important tool for communication during emergency situations. With the widespread use of smartphones, people can quickly share information about emergencies in real-time. As a result, many organizations, such as disaster relief agencies and news outlets, are interested in monitoring Twitter for relevant information.

However, it is not always easy to determine whether a tweet is actually referring to a real emergency. For example, a tweet may use words like "ABLAZE" metaphorically, which can be easily understood by humans but not by machines.

This dataset provide 10,000 tweets to train a machine learning model that can clarify the real and fake emergency.

1.2 Methods

Data Preprocessing Methods

Several data preprocessing techniques like removing special but meaningless characters and URLs are adopted to improve the performance.

Details will be given in 3.1.2.

Training Methods

Gate recurrent unit, LSTM and pre-trained model(BERT) are used to train the models.

Details will be given in Section 2.

1.3 Challenges

- A number of sentences and words from Twitter tend to be wordy and mis-odered, which makes harder for machine to learn and make judgement.
- The usage of words largely depend on personal emotions and hobbies, which also increases the judging complexity.
- The amount of data is not sufficient to train, especially after the training dataset is separated with rate 7:3.

2 Approach

2.1 Baseline Model

This model use GLOVE to embed the words.

The model structure is showed in Fig.1.

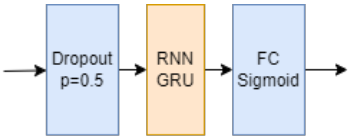

Figure 1: Structure of Baseline Model

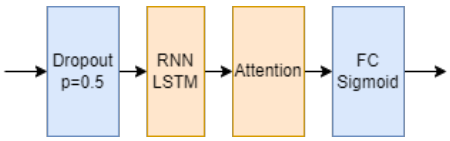

Figure 2: Structure of Improvement Model1

Dropout layer: This is a regularization technique that randomly drops out some of the input elements with a specified probability (dropout_rate), in order to prevent overfitting.

GRU layer: This is the main recurrent layer of the model, which processes the input sequences and produces a sequence of hidden states. The GRU has hidden_units number of hidden units, and one layer. GRUs are a type of recurrent neural network (RNN) architecture designed to address the vanishing gradient problem that can occur in traditional RNNs during training.GRUs have two gates, the reset gate and the update gate. The reset gate controls the extent to which the previous hidden state influences the current hidden state, whereas the update gate determines how much of the new hidden state is a linear combination of the previous hidden state and the candidate hidden state.

Linear layer: The model applies a linear transformation to the context vector obtained from the attention layer. The output of this layer is passed through the sigmoid activation function to obtain a probability score between 0 and 1, indicating the likelihood of the input belonging to the positive class.

2.2 Improvement1

This model use GLOVE to embed the words.

The model structure is showed in Fig.2.

Dropout layer: This is a regularization technique that randomly drops out some of the input elements with a specified probability (dropout_rate), in order to prevent overfitting.

LSTM layer: The model process the input sequences and produce a sequence of hidden states. The LSTM has hidden_units number of hidden units and one layer. Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) architecture specifically designed to address the vanishing and exploding gradient problems that can occur during the training of traditional RNNs. The input to the LSTM layer is a tensor of word embeddings, and the output is a tuple of hidden states and cell states. An LSTM unit consists of three primary gates: the input gate, the forget gate, and the output gate. Additionally, there is a cell state that maintains long-term information. The input gate controls the flow of new information into the cell state, while the forget gate determines which information from the previous cell state is retained or discarded. Finally, the output gate modulates the information that is passed on to the next hidden state. These gates work together to ensure that LSTMs effectively learn and retain relevant information from sequences.

Attention layer: The model uses an attention mechanism to compute attention weights for each time step of the LSTM output sequence. The attention weights are used to compute a context vector that captures the most important parts of the input sequence. The Attention class defines this attention mechanism to compute attention scores for each time step.

Linear layer: The model applies a linear transformation to the context vector obtained from the attention layer. The output of this layer is passed through the sigmoid activation function to obtain a probability score between 0 and 1, indicating the likelihood of the input belonging to the positive class.

2.3 Improvement2

For the third model, we used the pretrained model "bert-base-uncased" in huggingface. We also use the corresponding embedding method to embed the words.

The structure of BERT is showed in Fig.3. This model was pretrained on a large corpus of English data in a self-supervised fashion with the amount training parameter of 110M. Then we perform simple fine-tuning method on our dataset.

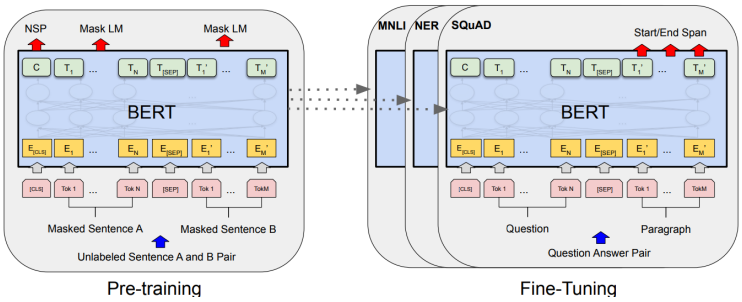

Figure 3: Structure of BERT [Devlin et al., 2019]

3 Experiments

3.1 Data

3.1.1 Description

Training Dataset

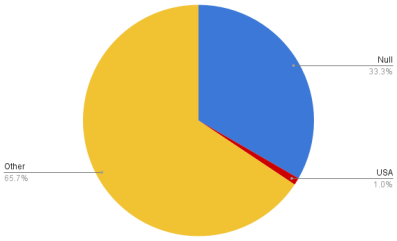

Figure 4: Distribution of Attribute Location-Training

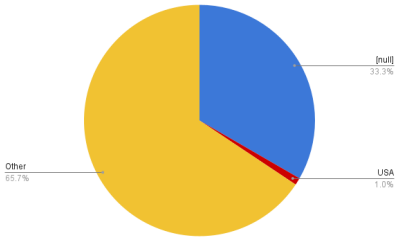

Figure 5: Distribution of Attribute Location-Testing

- Training set contains a unique identifier for each tweet, 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) and the target denoting whether a tweet is about a real disaster (1) or not (0).
- The distribution of label is around 0:1 = 0.57:0.43 in the training dataset. The amounts of false emergency and true emergency are approximately similar. This training dataset can still be considered as a balanced dataset.
- The distribution of location is showed in Fig.4. There exists significant part of missing data and most of the amount of the locations are not significant.

Test Dataset

- The distribution of location is showed in Fig.5. There exists significant part of missing data and most of the amount of the locations are not significant. It is worth to note that the distribution of location in the test data is similar to the distribution of location in the training data.

3.1.2 Preprocessing

As mentioned in the Section 3.1.1, there exists significantly amount of missing values in the data. Therefore, during training and prediction, we only use the information from text for better results.

Two approaches are adopted with respect to different improvement. Approach1 is used in model improvement1. Approach2 is used in model improvement2.

Improvement1

- Remove URLs: URLs in tweets are often irrelevant to the content and meaning of the text. By removing them, we eliminate potential noise that could negatively affect the model's understanding of the data.
- Remove special characters, numbers, and punctuation: These elements can introduce unnecessary complexity and noise into the text. By removing them, we simplify the text and make it easier for the model to focus on the important words and their relationships.
- Convert to lowercase: Text data can contain a mix of uppercase and lowercase letters. By converting all characters to lowercase, we standardize the text and make it easier for the model to recognize similar words and patterns, regardless of their case.
- Tokenize and remove stopwords: Tokenization breaks the text into individual words (tokens) for easier processing. Removing stopwords helps the model focus on more meaningful words, as stopwords are common words that usually do not carry much meaning.
- Lemmatize the tokens: Lemmatization reduces words to their base forms (e.g., "running" becomes "run"). This helps the model treat different forms of the same word as a single entity, which can improve the model's understanding of the text and lead to better training results.

Improvement2

- Convert the tweet to lowercase: Converting the text to lowercase helps to maintain consistency and reduce the vocabulary size, as words in different cases are treated as the same token (e.g., "Hello" and "hello" are considered the same).
- Expand contractions: For each contraction found in the contractions dictionary, the function replaces the contracted form with the expanded form in the tweet (e.g., "won't" will be transformed into "will not"). This step helps to standardize the text and makes it easier for NLP models to learn patterns and relationships between words.
- Remove special characters and digits: The function uses a regular expression pattern that matches any non-alphanumeric character (i.e., anything that is not a letter or a digit), and replaces it with an empty string. This step removes special characters, punctuation marks, and digits, leaving behind only alphabetic text. This simplifies the text and reduces noise that may not be relevant to the task.

3.2 Experimental details

3.2.1 Baseline

Details

- Optimizer: Adam - learning rate 1e-4
- Batch: 32
- Dropout rate: 0.5
- Epoch: 100

Motivation

No speicifc motivation here. We follow the instruction from tutorial.

As for the setup of number of epoch, from validation accuracy figure in Fig.6, there is no need to add more epochs because baseline modelś accuracy keeps fluctuate after 30 epochs.

3.2.2 Improvement1

Details

- Hidden unit of attention layer: 64
- Others are the same as the one in Baseline model.

Motivation

- Usage of Attention Layer: The attention layer is adopted to help the model focus on relevant parts of the input sequence when generating output, which could improve the model's performance.
- Usage of Hidden Layer Unit Parameter: The hidden layer's parameter 64 is set as a choice that balances model complexity, computational cost, and performance.
- We tried other dropout ratio. The ratio 0.5 turns out to work efficiently.

3.2.3 Improvement2

Details

- Optimizer: Adam - learning rate 1e-5
- Batch: 16
- Epoch: 10

Motivation

- Usage of Pre-trained Model: Pre-trained models have been trained on massive amounts of data, enabling them to learn general representations of language. Also, they have been trained on a diverse range of tasks, which makes them capable of performing well on a variety of NLP tasks.
- Usage of BERT: BERT adopts a bidirectional structure, meaning that it can consider both the preceding and following words when processing a given word in a sentence. This allows it to capture long-range dependencies in language, which is crucial for many NLP tasks.
- Usage of Learning Rate: For fine-tuning task, learning rate is set to be small to tune the model in a careful way.
- Usage of Batch Size: For fine-tuning task, batch size is set to be small to tune the model in a careful way.
- Usage of Epoch: For fine-tuning task, epoch is set to be small to tune the model in a careful way.

3.3 Results and Analysis

For start, the measurement of training process and testing process should be justified.

We use BCE loss to evaluate training loss,

$$L = \frac{1}{N}\Sigma[-(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i))] \tag{1}$$

where N is the amount of data in training set.

We use number of correct predictions evaluate validation accuracy,

$$Acc = \frac{1}{N}\Sigma[\mathbf{1}(\hat{y}_i = y_i)] \tag{2}$$

where N is the amount of data in validation set.

3.3.1 Baseline

Result

The plot of training loss and validation accuracy during training process are showed in the red line in Fig.6.

Analysis

For training process, the training loss starts at a low loss and decreases slowly as the epoch increases.

The training process shows that the model learns well at first and keeps learning the information in the training data.

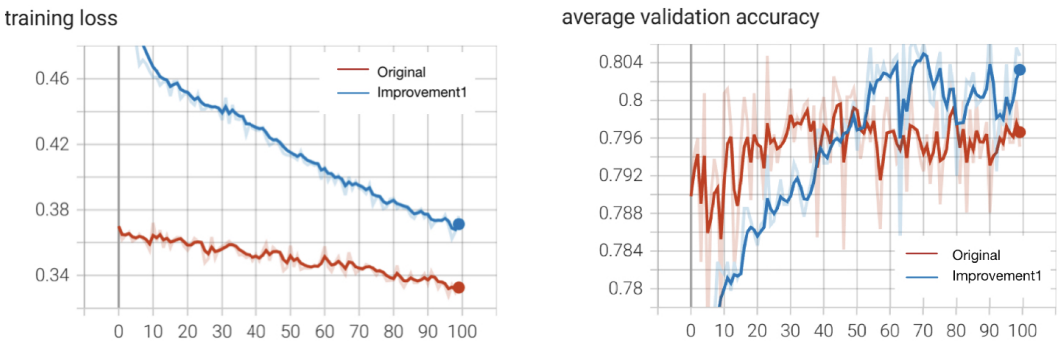

Figure 6: Training Loss(L) and Validation Accuracy(R) of Baseline and Improvement1 Model

For validation process, the validation accuracy increases fast in the first ten epoch, then the validation accuracy fluctuates at around 0.796.

The validation process indicates that in the first few epoch, the validation accuracy starts at a relatively high position, which can be explained by the low starting training loss. In the later epochs, the model increases its validation accuracy a bit but fails to perform better results on the validation set. This indicates that the prediction error may caused by the structure deficiency of model. Compared with the decreasing training loss, the model may overfit the data.

3.3.2 Improvement1

Result

The plot of training loss and validation accuracy during training process are showed in the blue line in Fig.6.

Analysis

For this part of analysis, we mainly focus on the improvement of this model compared with baseline model.

For training process, the training loss starts at a relatively high position than the baseline model. It decreases faster than the baseline model during the process. However, the training loss is always greater than the baseline model.

The training process shows that the improvement1 model takes more time to learn. This mainly due to the use of attention layer and LSTM. Both of them requires more parameters than GRU. The reason that improvement1 model's training loss is higher may be the over-fitness in the baseline model.

For validation process, the validation accuracy starts at a relatively low position and increases faster than baseline model. In the later epochs, the validation accuracy fluctuates at around 0.802, which is higher than the baseline model.

The validation process indicates that in the first few epoch, the faster increasing validation accuracy can be explained by the faster decreasing training loss, which also shows that the information learned in the early epochs is efficient in validation. In all, the model reaches higher validation accuracy then baseline model, which indicates that it learns more useful information about data. However, its prediction error also shows that it suffices from structure deficiency of model.

It is also worth to note that improvement1 model has higher validation accuracy with higher training loss compared with baseline model. This further proves that there exists over-fitting issue in the later epoch of baseline model.

Explanation about the improvement:

- Adding an attention layer can help LSTM focus on different parts of the input sequence, improving their ability to capture important information across longer sequences.
- Detailed data preprocessing is adopted for machine to learn better.

3.3.3 Improvement2

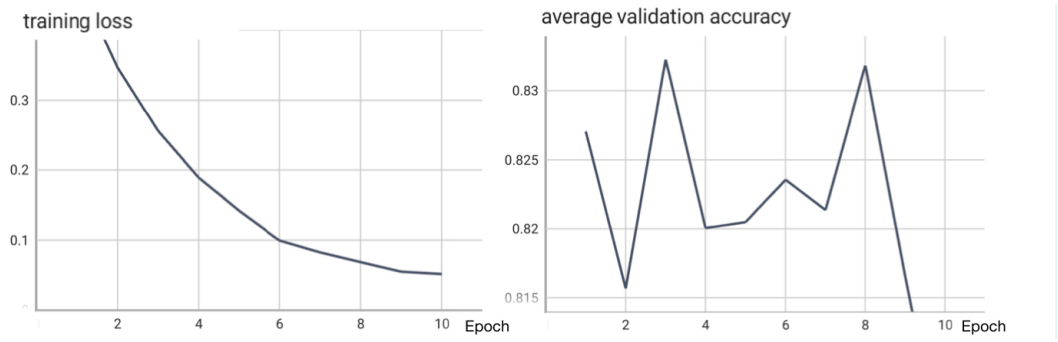

Figure 7: Training Loss(L) and Validation Accuracy(R) of Improvement2

The plots of training loss and validation accuracy of improvement2 model is showed in Fig.7.

Here we do not put the training loss and validation accuracy of improvement2 model together with the first two model because of the large differences absolutely.

For training process, the training loss keeps decreasing in a quadratic way.

The training process indicates that the model keeps learning. The quadratic rate shows that the model learns fast at first and learns details information in the later epochs.

For validation process, the validation accuracy is high at first and fluctuates at around 82.5%.

The fluctuates may be caused by the fitting and over fitting on training data. It is also possible that the fluctuates is caused by the randomness. The decreasing tendency after 8th epoch shows that the model may overfit the data. Further investigation is required for more conclusion.

Both in the aspect of training loss and validation accuracy, the improvement2 model outperforms the first two model. It is reasonable **because** improvement2 model uses the pre-trained BERT , which has been trained on a large dataset with powerful structure, and more detailed data preprocessing.

However, it seems that pre-trained model fails to learn much about this specific dataset. This may due to the domain gap and fine tuning techniques, which might be a direction to explore and improve in the future. (However, it still reaches the top 5% on the kaggle, so we stop here)

3.3.4 Kaggle Submission

Here is the leaderboard link: https://www.kaggle.com/competitions/nlp-getting-started/leaderboard .

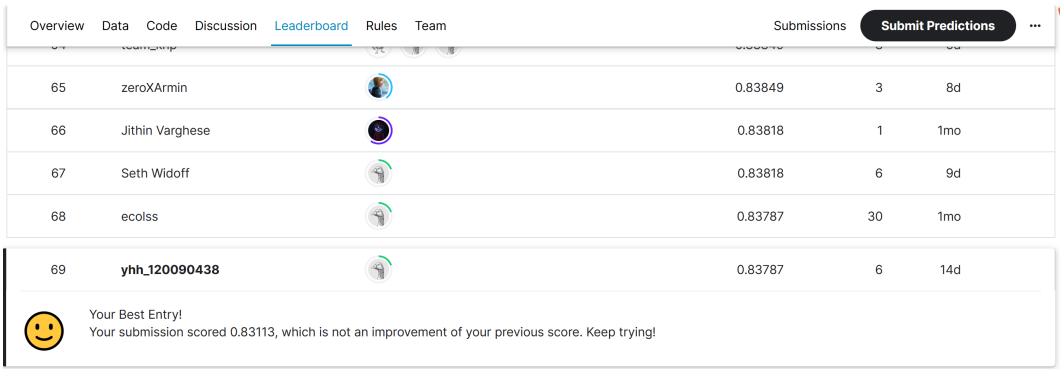

Figure 8: Result on Kaggle

Bibliography

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.