1 Introduction

Fake-EmoReact 2021 is a competition related to natural language programming, in which participants have to distinguish real news and fake one. Participants get the data collected from twitter, including context, reply, gif picture, and categories of a corresponding gif picture. In this challenge, a total of 209,008 sequential utterances and their fake news labels were provided to the participants who registered for this shared task.

There are three files that participants will get, the training data *train.json*, Unlabeled development data *dev.json*, and Unlabeled evaluation data *eval.json* After developing the model using the training data, participants can test their model using unlabeled development data, whose answers are unknown, but they can get the precision score, recall score, and F1 score. The final score will be the accuracy of evaluation data, whose answers are unknown either, but participants will not get the score until the evaluation phase finish.

2 Related Work

According to SocialNLP EmotionGIF 2020 Challenge Overview, there are many kinds of approaches that can do the task.

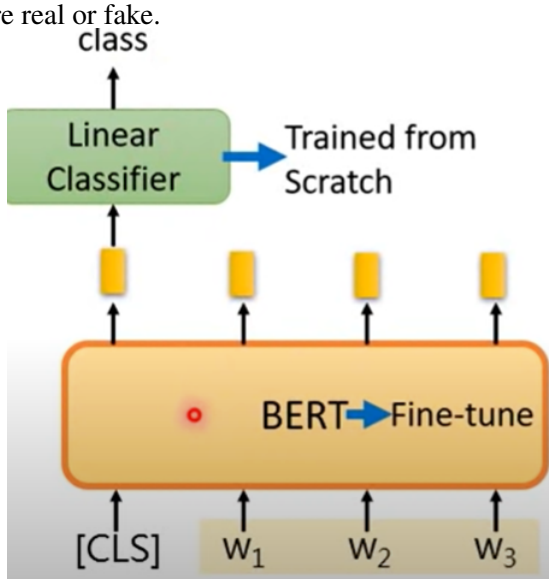
Rank	Approach
1	Ensemble of transformers (large), label embedding, mixconnect, PAL-N
2	RoBERTa
3	Preprocessing, ensemble of transformers (base)
4	Statistical features, similarity features, BERT, LightGBM
5	Ensemble of RNNs (CNN, BiLSTM, GRU)

And according to a blog written by LeeMeng, in which the maintainer share his experience that he participated in a contest held by Kaggle in 2019, called WSDM - Fake News Classification and the topic was also differentiating real news from fake one.

Based on both of the experiences, I decided to use BERT to handle it. BERT(Bidirectional Encoder Representations from Transformers) is an encoder in the transformer. It can output an embedding after we input a sentence, then we can use linear classifier to predict our target.

The way of training BERT is that we mask out some of the words in a sentence and then predict them. The other way of training BERT is to predict whether two sentences should be related context or not. With both of the methods, we can get a better trained model.

For our target, we can just fine-tune the BERT and train our linear classifier to predict the tweets are real or fake.



3 Data

The training data is a json file, which contains the following fields.

Field	Description
idx	running index of the samples
text	the text of the original tweet; may include mentions, hashtags, emojis
categories	the GIF category of the GIF response
context_id	the index of the response tweets to the idx
reply	the text content of the response tweet

Note that there may be some data that contains empty category or empty reply, so we have to consider both of them in the mean time.

The submitted data should be in csv file, containing idx, context_idx, label to uniquely set the label to evaluation data.

4 Method

4.1 Preprocess

The text and reply field contains many kinds of characters, like hashtag #, mention @, emoji, and new line character \n, \r.

For text and reply, I had done the following preprocess before I trained my model. Users in twitter may use hashtag to emphasize some words in a sentence, so I keep the words after hashtag and only delete the hashtag. In my opinion, the mention is used to tag user in twitter, so I delete all of the words that contain mention. I deleted all of the emoji because I thought it is hard to determine what an emoji really means. Because \n and \r are meaningless in a sentence, which have no semantics, I delete all of them.

As for categories, since some words are the combination word, for example, happy_dance, I replaced it with two words "happy dance".

Some reply includes url that is somekind related to it, but I think url itself is meaningless to the sentence, so I decide to delete it.

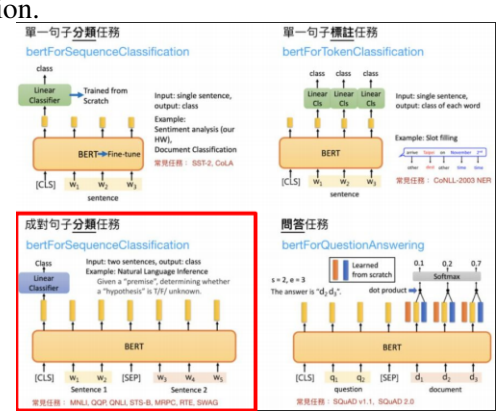
The following table is an example of all I did to the input data.

Before Preprocess	After Preprocess
@user	empty string
#word	word
emoji	empty string
happy_dance	happy dance
\n, \r	empty string
http://xxx	empty string

Last, I combine replies and categories into a string, by which we can construct a valid input to the model.

4.2 Making Valid Input

As we can see, the following picture shows what BERT can do, and our target is sequence classification.



I use PyTorch as my machine learning template, and according to the it requires input that need to have some steps of transformation to become a valid one.

As a result, we cannot just input the pure string to BERT. Instead, we have to tokenize each word first. After tokenization, we will get a tensor that contains indexes of the words in the PyTorch word sequence.

Then, because our input contains two sentences, we need a segment tensor that contains which word belongs to first sentence, and the other belongs to second sentence.

To set the ground truth to train the model, we also need a label tensor. Because the output is either real or fake, we can simply set 0 for real and 1 for fake.

The following table shows an example what I did to transform the original string data into the valid input for PyTorch BERT model.

Variable	Value
String	Context: private underage kid; Reply: nothing news
Label	fake
tokens	1037, 2797, 2104, 13366, 14074
segments	0, 0, 0, 1, 1
label	1

4.3 Setting Parameter

According to there are many kinds of pretrained model we can use. I have tried bert-base-uncased, bert-large-uncased, bert-base-cased, bert-large-cased and get the score. Finally I found that I can get the highest score by choosing bert-base-uncased, which had the score 0.987 in the practice phase.

About epoch, I use 6 because I found that after 5 epoch, the accuracy is no longer get higher and the loss may be higer which represents over-fitting.

There are 168520 pieces of training data in the training set, but it is very time consuming to train on all of the data, so I randomly choose 10% of the data for training.

Here is the table of the parameters of the training process.

Parameter	Value
Pretrained Model	bert-base-uncased
Epoch	6
Percent of Training Data	10%

4.4 Mini-batch

Because there is to much training data, we can hardly input all of the data once. I create a mini-batch data loader to cut the data set into many batches and train the model one by one.

5 Experiment

As I mentioned above, I did the preprocess which contains replacing url, mention tag, hashtag, and so on. If I did not do this preprocess, the score would be lower, which is 95%, 3% less than the one with preprocess.

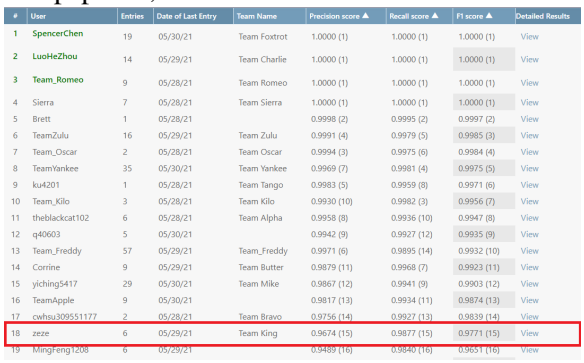
Preprocess	Score
True	98%
False	95%

It takes me thirteen minutes to train a epoch with 10% training data, namely 16852 pieces of data. It is so time-consuming that I cannot deal with all of the data simultaneously.

The following is the table that represents the loss and accuracy after n epoch. We can find that the accuracy is 0.999 after third epoch and fifth epoch had the lowest loss.

Epoch	Loss	Accuracy
1	2189	0.982
2	765	0.997
3	424	0.999
4	206	0.999
5	178	0.999
6	232	0.999

I get 18 place in total of 28 participants in develop phase, where the final F1 score is 0.977



#	User	Entries	Date of Last Entry	Team Name	Precision score ▲	Recall score ▲	F1 score ▲	Detailed Results
1	SpencerChen	19	05/30/21	Team Foxtrot	1.0000 (1)	1.0000 (1)	1.0000 (1)	View
2	LuoHeZhou	14	05/29/21	Team Charlie	1.0000 (1)	1.0000 (1)	1.0000 (1)	View
3	Team_Romeo	9	05/28/21	Team Romeo	1.0000 (1)	1.0000 (1)	1.0000 (1)	View
4	Sierra	7	05/28/21	Team Sierra	1.0000 (1)	1.0000 (1)	1.0000 (1)	View
5	Brett	1	05/28/21		0.9998 (2)	0.9995 (2)	0.9997 (2)	View
6	TeamZulu	16	05/29/21	Team Zulu	0.9991 (4)	0.9979 (5)	0.9985 (3)	View
7	Team_Oscar	2	05/28/21	Team Oscar	0.9994 (3)	0.9975 (6)	0.9984 (4)	View
8	TeamYankee	35	05/30/21	Team Yankee	0.9969 (7)	0.9981 (4)	0.9975 (5)	View
9	ku4201	1	05/28/21	Team Tango	0.9983 (5)	0.9959 (8)	0.9971 (6)	View
10	Team_Kilo	3	05/28/21	Team Kilo	0.9930 (10)	0.9982 (3)	0.9956 (7)	View
11	theblackcat102	6	05/28/21	Team Alpha	0.9958 (8)	0.9936 (10)	0.9947 (8)	View
12	q40603	5	05/30/21		0.9942 (9)	0.9927 (12)	0.9935 (9)	View
13	Team_Freddy	57	05/29/21	Team_Freddy	0.9971 (6)	0.9895 (14)	0.9932 (10)	View
14	Corrine	9	05/29/21	Team Butter	0.9879 (11)	0.9968 (7)	0.9923 (11)	View
15	yiching5417	29	05/30/21	Team Mike	0.9867 (12)	0.9941 (9)	0.9903 (12)	View
16	TeamApple	9	05/30/21		0.9817 (13)	0.9934 (11)	0.9874 (13)	View
17	cwhsu309551177	2	05/28/21	Team Bravo	0.9756 (14)	0.9927 (13)	0.9839 (14)	View
18	zeze	6	05/29/21	Team King	0.9674 (15)	0.9877 (15)	0.9771 (15)	View
19	MingFeng1208	6	05/29/21		0.9489 (16)	0.9840 (16)	0.9651 (16)	View

However, in the evaluation phase, I only get 27 place from 31 participants, where the F1 score is 0.415.

#	User	Entries	Date of Last Entry	Team Name	Precision score ▲	Recall score ▲	F1 score ▲	Detailed Results
1	ccc_gogo	8	06/02/21		0.8532 (2)	0.8456 (1)	0.8435 (1)	View
2	ChenMian	8	06/01/21	Team Edward	0.8399 (3)	0.8334 (2)	0.8314 (2)	View
3	Papa	11	06/01/21	Team Papa	0.8300 (4)	0.8239 (3)	0.8219 (3)	View
4	Yao	1	06/02/21		0.8560 (1)	0.8129 (4)	0.8095 (4)	View
5	TeamZulu	15	06/02/21	Team Zulu	0.8075 (6)	0.8066 (5)	0.8060 (5)	View
6	TeamJuliet	17	06/01/21		0.8002 (7)	0.7997 (6)	0.7998 (6)	View
7	SpencerChen	5	05/31/21	Team Foxtrot	0.8215 (5)	0.7951 (7)	0.7886 (7)	View
8	Brett	13	06/01/21		0.7905 (9)	0.7870 (8)	0.7855 (8)	View
9	yuchingtw	7	06/02/21	Team Victor	0.7442 (16)	0.7211 (9)	0.7162 (9)	View
10	LuoHeZhou	5	06/01/21	Team Charlie	0.7480 (14)	0.7142 (10)	0.7016 (10)	View
11	TeamIndia	9	06/01/21	Team India	0.7726 (10)	0.6981 (11)	0.6725 (11)	View
12	Team_Oscar	4	05/30/21	Team Oscar	0.7941 (8)	0.6851 (12)	0.6490 (12)	View
13	yiching5417	6	06/01/21	Team Mike	0.6565 (25)	0.6489 (16)	0.6432 (13)	View
14	linzinofan	8	06/02/21	team November	0.7394 (18)	0.6664 (13)	0.6353 (14)	View
15	ku4201	1	05/30/21	Team Tango	0.7483 (13)	0.6647 (14)	0.6302 (15)	View
16	TeamYankee	10	06/01/21	Team Yankee	0.7129 (20)	0.6567 (15)	0.6293 (16)	View
17	TeamGolf	6	06/01/21	Team Golf	0.6215 (26)	0.6202 (19)	0.6197 (17)	View
18	Team-Uniform	6	05/31/21		0.6873 (22)	0.6309 (17)	0.5975 (18)	View
19	cwhsu309551177	1	05/31/21	Team Bravo	0.7563 (11)	0.6251 (18)	0.5657 (19)	View
20	TeamApple	3	05/31/21		0.6814 (23)	0.5932 (20)	0.5331 (20)	View
21	Team_Ink	2	06/02/21		0.5178 (30)	0.5177 (27)	0.5177 (21)	View
22	Team_Delta	8	06/01/21	Team Delta	0.5337 (28)	0.5284 (26)	0.5071 (22)	View
23	jchsieh	6	06/02/21		0.5014 (31)	0.5014 (30)	0.5014 (23)	View
24	Sierra	22	05/31/21	Team Sierra	0.7277 (19)	0.5779 (21)	0.4905 (24)	View
25	maxmax	3	05/31/21	Team X-Ray	0.6640 (24)	0.5621 (23)	0.4776 (25)	View
26	Team_Romeo	3	06/02/21	Team Romeo	0.7427 (17)	0.5690 (22)	0.4707 (26)	View
27	zeze	2	05/30/21	Team King	0.6903 (21)	0.5370 (24)	0.4159 (27)	View
28	Team_Freddy	12	05/31/21	Team_Freddy	0.7490 (12)	0.5353 (25)	0.4039 (28)	View
29	Corrine	1	05/31/21	Team Butter	0.5308 (29)	0.5021 (29)	0.3468 (29)	View
30	theblackcat102	7	06/01/21	Team Alpha	0.5681 (27)	0.5023 (28)	0.3397 (30)	View
31	Team_Kilo	2	05/30/21	Team Kilo	0.7455 (15)	0.5000 (31)	0.3293 (31)	View

I think that why the accuracy was such low is because of the following reasons.

- Testing data in evaluation phase is more complicated than that of develop phase.
- The model may be over-fitting, as we can find that the loss of sixth epoch was higher than fourth and fifth epoch.

6 Conclusion

There are still many points that I can improve.

For example, the way of preprocess, so far I delete all of the urls and mention tags. However, this way somekind lose the semantics, so I consider it better to replace all of the urls into a URL tag, and replace all of the mention tags into USER tag.

Emoji is also an important information to collect semantics, but it is not easy to change all of the emojis into a tag. I think we can still try to turn every emoji into different tags and let BERT learn itself.

As for categories, maybe it has no problem to the underline, which means we can train it directly without any preprocess.

The time-consuming event can also be dealt with. I did not use early stop to shorten the time for training, and I did not use any pruning technique, either. Both of the ways are reasonable for BERT to shorten the time for training, so we can improve it with such techniques.

In addition, it is worth testing to use different model to do the same work. For instance, we can conbine LightBGM and BERT and see the result of voting. Using different model like RoBERTa is maybe a good way, too.

The following table is the summary of what improvement I can do and the new methods I can try.

Item	Improvement
URL	using url tag
Mention	using user tag
Emoji	turn into different tags
Underline	reserve it
Time Comsume	early stop and pruning
Model	Conbine other model or try different model

7 Task Allocation

There is only me in the Team King. I did all of the work independently.

8 Question and Answer

8.1 What is the meaningless string and how to deal with it?

The definition of meaningless string is that the strings have no semantics to the sentence, so we cannot get any information from them. I deleted all of them, but now I think it may be better to turn them into tags.

8.2 Do you have handle emojis?

No, I deleted all of them, but now I think it is better to turn them into tags and let the model learn by itself.

8.3 Why do you delete the underline which is in the combination words?

I did not know there is such combination word before, so maybe I can check whether the words are in the tokens first.

8.4 Can it improve the evaluation result by increasing the training data?

Yes, so far I use 10% of data for training. I think it can preform better if I use more data for training.

8.5 Have you considered early stopping to deal with the time consume?

I did not know early stopping before, but now I think it may be a good way to do it.