Fake-EmoReact 2021-TeamZulu: Deep Neural-based Models for Detecting Fake News on Twitter

William W. Chen[1], **Wei-Wei Du**[2], **and Yi-Ting Chang**[1]
[1]National Yang Ming Chiao Tung University, Hsinchu, Taiwan
[2]National Tsing Hua University, Hsinchu, Taiwan
{willyc.cs06, joshchang0111.ee06}@nycu.edu.tw
wwdu@gapp.nthu.edu.tw

1 Introduction

With the increasing popularity and the rapid development of social media, news and plenty of information can travel swiftly on the internet nowadays. However, people are not always aware of the truthfulness about the information they read, while misinformation can usually result in enormous loss on the society in different aspects. For example, a study shows that roughly 6,000 people were hospitalized in the first three months of 2020 due to the coronavirus misleading news. Moreover, researchers even claim that at least 800 people died during this time period because of disinformation related to the topic of coronavirus.[1]

Twitter has become a dominant internet community which allows people to share whatever they see no matter it's a fact or not. Due to this reason, fake news can possibly be spread widely in a short period. To mitigate the damaging effect that fake news brought to the society, automatically detecting fake news on social websites has drew a lot of attention in recent years.

We participate in the Fake-EmoReact 2021 challenge, whose goal is to determine whether a tweet contains fake news or not. The challenge aims to predict the label of the source tweet by taking its conversational thread into consideration, including all textual and animated GIF replies. We implement some popular deep neural-based models as our fake news classifiers, including convolutional neural networks, recurrent neural networks and BERT-based models. Also, we apply a recently published method proposed by Clark et al. (2020) to deal with this problem.

To analyze how each component of our training process influence the results, we construct several experiments during the training and practice phase to see the F1-score on the development set. We compare the results between training our models with and without fine-tuning the pre-trained word embedding vectors. Besides, we also compare the results between training with or without weighted-loss on two classes because of the data imbalance issue in this dataset.

Also, due to the special property of Twitter dataset described on the challenge website, the truth label depends on whether the source tweet contains hashtag fake news ("#FakeNews") or not. In other words, all replies under the same source tweet must have the same label. Consequently, we come up with a new concept named *Reply Vote*, which means after the predictions from each model, we further do a majority vote on each conversational thread, the source tweet's label is decided by all its replies. Namely, after this approach, all the data points with the same source tweet will have the same label.

The main contributions of our paper are summarized as follows:

- We compare the strengths and weaknesses on this challenge between traditional models and pre-trained models.
- We discuss different training strategies and ensemble techniques.
- We propose a method called *Reply Vote* that is suitable for analyzing tweet conversations.

2 Related Work

In this section, we briefly review the word embedding and sequence models that we've implemented in this challenge, and we also investigate some studies of fake news detection.

[1]https://www.who.int/news-room/feature-stories/detail/fighting-misinformation-in-the-time-of-covid-19-one-click-at-a-time

Word Embedding Pre-trained word representations are very important and serve as a key factor in neural language understanding, previous approaches on learning vector space word representations have many accomplishment in fine-grained semantic and syntactic regularities by doing vector arithmetic.

In the work of Pennington et al. (2014), they succeed in analyzing and make explicit the model properties needed for the regularities to be captured in word vectors, introducing a new global log-bilinear regression model that combines the advantages of global matrix factorization and local context window methods, efficiently leveraging the statistical information by only training on the nonzero elements in a word-word co-occurrence matrix.

Ideally, learning high quality vector space word representations should capture both complex characteristics of word used, such as syntax and semantics, and also how these uses vary across linguistic contexts. In Peters et al. (2018), their work introduces a new type of deep contextualized word representation that successfully addresses the challenges mentioned above, significantly improves the state of the art in that time for a range of language understanding problems.

Sequence Model Dealing with sequential data is crucial in the field of natural language processing. Lots of works tackle this problem by utilizing various kinds of deep learning frameworks. Kalchbrenner et al. (2014) propose a DCNN (Dynamic Convolutional Neural Networks) to capture word relations of varying size and achieve excellent performance in sentence classification tasks. Also, RNN-based models are widely used in this area due to the property of retaining memories in previous states by its sequential architecture.

However, these RNN-based models still have limited ability to capture the relations between long-distance words in inputs with very long sequences. Besides, RNN-based model takes large amount of time to train since it requires the inputs to be processed in sequential order, on the other hand, attention mechanism allows the model to be trained in parallel, and obtain outputs after looking at the whole input sequences.

Based on the development of attention mechanism, Vaswani et al. (2017) propose a transformers network which solely leverages attention architecture and achieve impressive results on several language modeling tasks. Since the training time is largely reduced by attention-based model like transformer, it enables training on larger datasets and leads to the development of state of the art pre-trained systems such as BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) , which gain more completely information by their bi-directional architecture.

Based on the framework of BERT, (Clark et al., 2020) introduce Electra, a text encoder which is trained to distinguish input tokens from high-quality negative samples produced by a small generator network. Therefore, it can significantly decrease the computational resource needed.

Fake News Detection Several research teams have been endeavored to detect fake news or unverified rumors on the internet based on textual content. For instance, Ma et al. (2016) use recurrent neural networks such as LSTM (Long-Short-Term-Memory) and GRU (Gated-Recurrent-Unit) to learn the hidden representation of the contextual information from different posts on microblogs. Aside from sequential-based models, Volkova et al. (2017) utilize a CNN (Convolutional Neural Network) to classify the veracity of news from Twitter and still achieve competitive results comparing to LSTM. In addition, Kaliyar et al. (2021) combine deep Convolutional Neural Network (CNN) and BERT, proposing a BERT-based model to handle the ambiguity in fake news detection.

3 Dataset

The dataset contains thousands of tweets from Twitter, including the source tweets and their response tweets. The source tweets are text only, but the response tweets include their text content and mp4, which stands for the file name of the GIF. The label is provided at the end of every sample, indicating whether the source tweet is fake news or not. Samples with "#FakeNews" are labeled as fake, and those without "#FakeNews" are labeled as real, which is combined from EmotionGIF (Shmueli et al., 2021) last year.

Label	# source tweet	# reply	# GIF
Real	31,799	31,799	31,799
Fake	3,219	136,722	7,076

Table 1: The number of label in training set

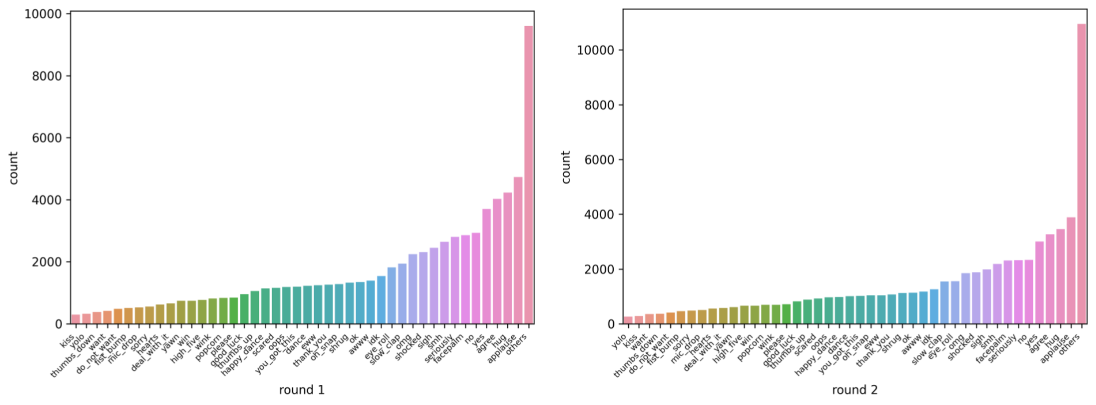


Figure 1: Distribution of categories

Dataset	**Round 1**			**Round 2**		
	# source tweet	# reply	Avg. # reply	# source tweet	# reply	Avg. # reply
Training	35,018	168,521	4.8	35,018	168,521	4.8
Development	8,891	40,487	4.5	1,202	45,036	37.5
Evaluation	2,400	88,664	36.9	2,798	110,492	39.5

Table 2: The number of source tweet and reply

Exploratory Data Analysis The label distribution of the training set is list in table 1. For the training set, the two classes are highly imbalanced. We can see that for every real tweet, there is only one reply and it must contain a GIF. In contrary, for every fake tweet there are 42 replies in average and only about two GIFs. It is obvious that there exists significant difference between the data distribution and structure for the two classes. Also, figure 1 shows the categories distribution of round 1[2] and round 2[3], we can easily observe that both rounds suffer from uneven distribution, the category "others" are much more than the other categories.

Another survey we do is to calculate the number of source tweets and reply respectively for both rounds, the results are shown in table 2. As we can see, in round 1, the average number of replies per tweet in development set is similar to training set, which is much less than the average number in evaluation set. We consequently infer that the evaluation set has different distribution from training and development set.

4 Methodology

The system we design for the Fake-EmoReact challenge can be divided into four parts: data preprocessing, word embedding, classification models, and models ensemble. The detailed work of each part is shown in the following subsections. Additionally, we discuss word embedding and classification models with the two categories below.

- CNN and RNN-based models: CNN, (Bi-) RNN, (Bi-) GRU, (Bi-) LSTM
- BERT-based models: BERT-cased, BERT-uncased, RoBERTa, Electra

4.1 Data Preprocessing

First, we concatenate the source tweet and the response tweet into one sentence then utilize the NLTK TweetTokenizer and perform some steps on the text content:

- Remove non-ascii characters.
- Convert emoji into meaningful text.
- Transform some weird punctuation into normal ones.
- Transform apostrophes to original words.
- Replace URL by a special token.
- Convert all the letters into lowercase. However, the result is not better than the original, so we skip this step in the end.

[2]Round 1 from Student Track
[3]Round 2 from Main Track

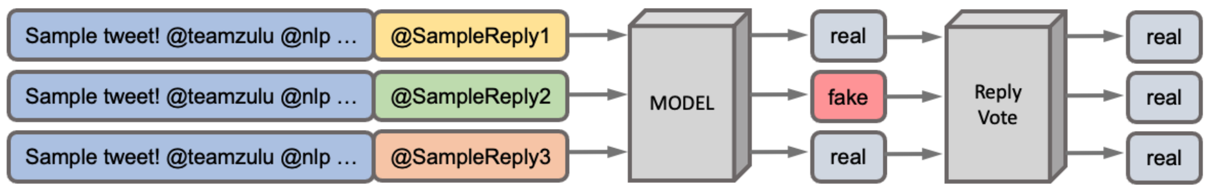

Sample tweet! @teamzulu @nlp ...
@SampleReply1
Sample tweet! @teamzulu @nlp ...
@SampleReply2
Sample tweet! @teamzulu @nlp ...
@SampleReply3
MODEL
real
fake
real
Reply Vote
real
real
real

Figure 2: Reply vote process

4.2 Word Embedding

In this task, we implement several kinds of models. For CNN and RNN-based models, the embedding layers are initialized with weights using pre-trained GloVe, a global log-bilinear regression model with 840B tokens and 300 dimension word vectors.

4.3 Models

The goal of Fake-EmoReact challenge is to distinguish whether the tweet represents fake news or not, so the task is modeled as a binary classification problem. To tackle this task, we implement a total of seven different classifiers, including CNN, (Bi-) RNN, (Bi-) GRU, (Bi-) LSTM, BERT, RoBERTa, Electra, to compare their results and construct the ensemble model.

For CNN and RNN-based models, we use cross entropy as the loss function along with weighted loss, which gives the wrong prediction that predicts real news as fake news four times the loss, since the dataset is imbalance with fake-to-real ratio as 4-to-1. Next, we take the word embedding as inputs to feed to the actual CNN and RNN-based models, which have a fully connected layer in the end that outputs the predictions of fake news or real news.

For Bert-based models, the model framework includes two phases, enhancing the pre-trained language model and fine-tuning the binary classification model. To be in line with the input format, for BERT-cased and BERT-uncased, we add the special token [CLS] at the beginning of the sentence and add the special token [SEP] both between the text and reply and at the end. For RoBERTa-base, we replace the token by <s> and </s>.

4.4 Ensemble

Since every model have its strengths and weaknesses, the ensemble framework aggregates the prediction of each model and results in a final prediction. Voting techniques are most commonly used when combining predictions for classification. We use three types of voting strategies, majority vote, soft vote and reply vote.

Majority Vote: Also known as hard vote, which selects the class label that has more than half the votes.

Soft Vote: In some cases, soft vote can improve hard vote because it takes into account more information. Based on the probabilities of all the predictions made by different classifiers, the target label with the greatest sum of weighted probabilities wins the vote.

Reply Vote: In this task, regardless of the number of reply, every data point with same source tweet has the same label. Therefore, we propose a voting strategy, *Reply Vote*, which means after the predictions from each model, we further do a majority vote on each conversational thread. The source tweet's label is decided by all its replies, thus transforming all tweets with the same source tweet into the same label by majority vote. Figure 2 shows the detailed reply vote process with inputs having the same source tweet and different reply.

5 Experimental Results

In this section, we present the detailed experiment settings used for constructing our models, and show the experiment results and competition results of the Fake-EmoReact challenge.

5.1 Implementation Details

During the training phase, we split the original training data into new training set and validation set, with 4-to-1 ratio, then train the model on the new training set and observe the testing results on the validation set.

CNN and RNN-based The experiment details are shown in table 3. Additionally, the max sequence length of input sentences are set to 256 for both CNN and RNN-based models, and the embedding layers on both CNN and RNN-based models have been fine-tuned during training phase as well.

Hyper-parameters	CNN	RNN-based
Batch size	16	16
Learning rate	1e-5	1e-5
Optimizer	Adam	Adam
Dropout	0.3	-
Input channel	1	-
Output channel	3	-
Kernel heights	[2,3,4]	-
Stride	1	-
Padding	0	-
Hidden dimension	-	256
Number of layers	-	1

Table 3: Hyper-parameters of CNN and RNN-based models

BERT-based For BERT, RoBERTa and Electra, because of the GPU memory limitation, we set the maximum sequence length as 256. Besides, most of the hyper-parameters were set as default arguments. The details are list in table 4.

Hyper-parameters	BERT and RoBERTa	Electra
Batch size	32	16
Learning rate	2e-6	2e-6
Optimizer	Adam	Adam
Epoch	4	3

Table 4: Hyper-parameters of BERT, RoBERTa and Electra models

5.2 Practice Phase Results

In this task, we use *Macro-F1-score* as the evaluation metrics.

$$Macro\text{-}F1 = \frac{\sum_{n=1}^{N} F1\text{-}score}{N}, \quad (1)$$
$$where\, N\, is\, the\, number\, of\, classes.$$

Results of both rounds are list in table 5. In round 1, We can find that BERT-based models achieve about 98% in F1-score, CNN and RNN-based models are around 92%. However, in round 2, all of the BERT-based models are only around 50%. We can deduce that our models are overfitting. In other words, our BERT-based models can not generalize to the unseen data. On the contrary, our CNN and RNN-based models still remain 75% in F1-score. Therefore, we choose three different types of classifiers, CNN, Bi-GRU and Bi-LSTM, then combine the results by majority vote, and the resulting F1-score rise to 83%. Moreover, it reaches 87% after reply vote. This observation leads to the conclusion that both majority vote and reply vote can significantly boost the performance when the individual performance is not good enough.

5.3 Evaluation Phase Results

The results are list in table 7. Our Bi-LSTM model achieves good results in both rounds, and *Reply Vote* improves the F1-score in round 1.

Round	Method	F1-Score	Rank
1	Bi-LSTM+ Reply Vote	80.66%	5th
2	Bi-LSTM	79.81%	4th

Table 7: F1-Score on evaluation set

5.4 Ablation Study

To analyze the performance using different training strategies, we do the ablation study and compare the results of them. Since most BERT-based models contain huge amount of parameters and thus require larger GPU memory and training time, the following experiments only involve CNN and RNN-based models.

Fine-Tuned Word Embedding As mentioned in previous section, we initialize the weights of the embedding layer in our CNN and RNN models with the pre-trained GloVe vectors. Then, we compare the results between fixing and fine-tuning the parameters of the embedding layer during backpropagation. The results are shown in table 8. As we can observe, almost all models have gained significant improvement after training with fine-tuned embedding.

Models	Fine-Tuned Embedding	
	No	Yes
CNN	90.37%	**95.57%**
RNN	85.98%	-
Bi-RNN	87.52%	90.06%
LSTM	87.51%	94.10%
Bi-LSTM	90.71%	93.67%
GRU	87.51%	90.62%
Bi-GRU	91.55%	95.01%

Table 8: F1-Score of CNN and RNN-based models on development set in round 1

Weighted-Loss Since the quantity of the real samples are four times smaller than the fake samples in this dataset, training in a normal way could possibly make the models classify the minority class incorrectly. To tackle this issue, we assign different weights to two classes on the loss function during the training process. In detail, when the model misclassifies a real sample into a fake one, we give four times the loss, indicating a higher penalty. We assume that this strategy can help the model learn more features of the minority class in a better manner.

The experiment results are list in table 9. We can see that training with weighted-loss worsen the performance in round 1. However, this strategy does make a considerable improvement in round 2. We infer that the different phenomenon is caused by the unique data distribution bias. Note that all the models can perform much better in round 1 than in round 2, which possibly means that the development set in round 1 has extremely similar distribution as the training set, unlike the development set in round 2. This inference is identical with the statistical result in table 2. Therefore, we consider training with weighted-loss can be more generalized across different datasets.

6 Conclusion

In this paper, we present the details and experiment results of constructing several deep neural networks to accomplish the Fake-EmoReact 2021 challenge. We implement different models, training strategies, and ensemble techniques to carry out the experiments, then compare the performance between models and choose the most suitable ones to submit to the challenge. Additionally, we propose a novel concept named *Reply Vote* to analyze tweet conversations. Our Bi-directional LSTM with *Reply Vote* eventually outperforms other models in the evaluation phase of round 1 and achieves 80.66% on F1-score, which wins the fifth place in the challenge.

7 Appendix

7.1 Explainable AI

To get a glimpse of what our models have learned, we implement the saliency map to visualize the amount that each word has contributed to the final prediction, reference from Li et al. (2016). The saliency map is constructed with saliency scores S calculated as follow, which is the magnitude (absolute value) of the first derivative of V, predicted output value of trained model, w.r.t the word embedding. Where e is the word embedding in the input sentences, and $class$ is the predicted output label.

$$S(E = e) = \mid \frac{\partial V_{class}(e)}{\partial e} \mid \tag{2}$$

The saliency score indicates how sensitive the model's final prediction is to each word embedding, which could give us a hint on how much each word embedding contributes to the model's final decision.

We implement the saliency map for CNN model and Bi-LSTM model. In comparison with BERT-based models, we also implement the saliency map for BERT-uncased model and RoBERTa model, figure 3 and figure 4 present two example results of the saliency map, one fake news and one real news.

8 Task Allocation

William W. Chen Implement and perform experiments on BERT (uncased), (Bi-) RNN, (Bi-) LSTM, ensemble techniques (majority vote, soft vote), saliency map.

Wei-Wei Du Preprocess data. Implement and perform experiments on BERT (uncased, cased), RoBERTa, Electra, ensemble techniques (majority vote, reply vote) and saliency map.

Yi-Ting Chang Exploratory data analysis. Preprocess data. Implement and perform experiments on BERT (uncased), CNN, (Bi-) GRU, ensemble techniques (majority vote, reply vote).

9 Question and Answer

Q1. How many layers of LSTM do we use?

A1. We use one-layer LSTM.

Q2. Would it be better to tackle data imbalance by sampling equal quantity on both classes?

A2. We sample 31,799 fake data points so that the quantity of two classes are the same. Then we train our BERT model with this training set and obtain the results of round 2 development set shown in table 10. It is obvious that the performance is worse than the original. Since the number of training data has been reduced, we consider the model cannot learn features of fake samples as good as usual.

Data	Precision	Recall	F1-Score
original	73.47%	51.71%	59.62%
balanced	69.97%	32.9%	50.17%

Table 10: Bert-base-uncased in Round 2 dev.

Q3. How do we feed natural language sentences into CNN? Would the position of each word be considered?
A3. First we obtain the embedding matrix from the input sentence so that its shape becomes (sequence length, embedding dimension). Then we perform convolution operation on this matrix with kernel of varying height but constant width which is same as the embedding dimension. Different kernel heights are list in table 3. Our CNN model only learns the collocation by different kernels, so the position of each word is not considered.

Q4. What might be the reason that simple models like CNN which learns more local information can outperform some complex models like BERT which learns more global information?

A4. Our guess is that the tweet dataset might not follow the proper syntax format, and the important information in tweets might only contain in a short sections, hence CNN is able to extract good features from looking at a local window compare to BERT which looks more globally and considers the whole sentences thoroughly, this probably increase the chances of overfitting.

Q5. Is it possible to know what kind of model to choose to perform the task from observing the data? For example, can we decide whether to choose BERT or CNN, LSTM before the experiment?
A5. We would say this is a case by case situation, there is probably no such perfect way to decide the models to use before carrying out the experiments. However, there are some empirical researches showing that CNN or LSTM can still perform fairly well on small dataset, and BERT can perform better when dealing with huge dataset, so the size of the dataset might give you an initial hunch on the models to choose from.

References

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Rohit Kumar Kaliyar, Anurag Goswami, and Pratik Narang. 2021. Fakebert: Fake news detection in social media with a bert-based deep learning approach. *Multimedia Tools and Applications*, 80(8):11765–11788.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 681––691.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, page 2227–2237.

Boaz Shmueli, Lun-Wei Ku, and Soumya Ray. 2021. Socialnlp emotiongif 2020 challenge overview: Predicting reaction gif categories on social media. *arXiv preprint arXiv:2102.12073*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Svitlana Volkova, Kyle Shaffer, Jin Yea Jang, and Nathan Hodas. 2017. Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 647–653.

Round	BERT -cased	BERT -uncased	RoBERTa	Electra	Majority Vote	Soft Vote	Majority + Reply Vote
1	97.77%	99.03%	**99.5%**	97.21%	99.32%	99.29%	99.32%
2	54.95%	59.62%	56.75%	50.1%	-	-	-

Table 5: F1-Score of BERT-based models on development set

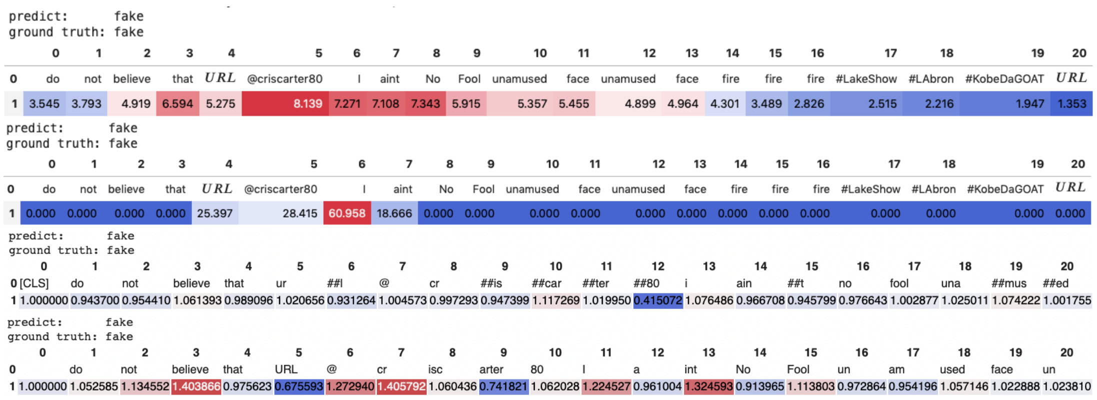
Round	CNN	RNN	Bi-RNN	LSTM	Bi-LSTM
1	95.57%	85.98%	90.06%	94.1%	93.67%
2	82%	-	73.62%	69.96%	79.54%

Round	GRU	Bi-GRU	Majority Vote	Soft Vote	Majority + Reply Vote
1	90.62%	95.01%	95.76%	**95.79%**	93.96%
2	68.82%	80.99%	83.63%	-	**87.02%**

Table 6: F1-Score of CNN and RNN-based models on development set

Round	Weighted-Loss	CNN	Bi-LSTM	Bi-GRU
1	No	**95.57%**	93.67%	95.01%
	Yes	92.57%	88.28%	92.87%
2	No	80.2%	75.6%	77.07%
	Yes	**82%**	79.54%	80.99%

Table 9: F1-Score on development set between training with and without weighted-loss



```
predict:      fake
ground truth: fake
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	do	not	believe	that	*URL*	@criscarter80	I	aint	No	Fool	unamused	face	unamused	face	fire	fire	fire	#LakeShow	#LAbron	#KobeDaGOAT	*URL*
1	3.545	3.793	4.919	6.594	5.275	8.139	7.271	7.108	7.343	5.915	5.357	5.455	4.899	4.964	4.301	3.489	2.826	2.515	2.216	1.947	1.353

```
predict:      fake
ground truth: fake
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	do	not	believe	that	*URL*	@criscarter80	I	aint	No	Fool	unamused	face	unamused	face	fire	fire	fire	#LakeShow	#LAbron	#KobeDaGOAT	*URL*
1	0.000	0.000	0.000	0.000	25.397	28.415	60.958	18.666	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

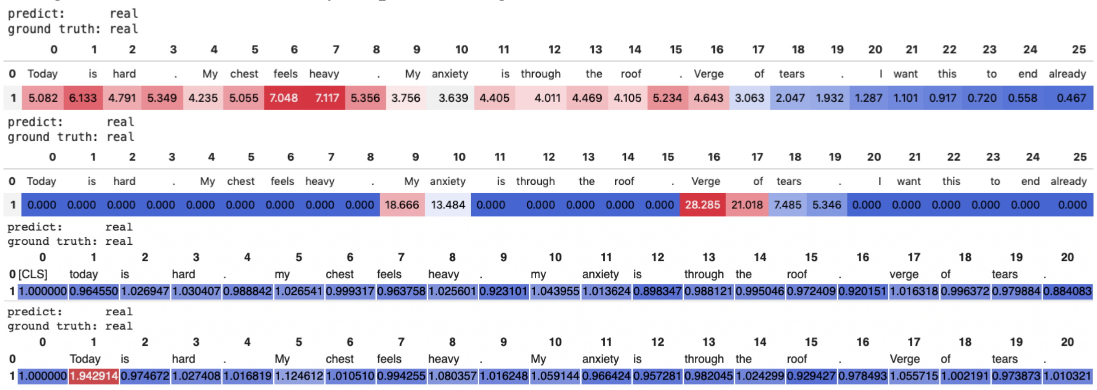
```
predict:      fake
ground truth: fake
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	[CLS]	do	not	believe	that	ur	##l	@	cr	##is	##car	##ter	##80	i	ain	##t	no	fool	una	##mus	##ed
1	1.000000	0.943700	0.954410	1.061393	0.989096	1.020656	0.931264	1.004573	0.997293	0.947399	1.117269	1.019950	0.415072	1.076486	0.966708	0.945799	0.976643	1.002877	1.025011	1.074222	1.001755

```
predict:      fake
ground truth: fake
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0		do	not	believe	that	URL	@	cr	isc	arter	80	I	a	int	No	Fool	un	am	used	face	un
1	1.000000	1.052585	1.134552	1.403866	0.975623	0.675593	1.272940	1.405792	1.060436	0.741821	1.062028	1.224527	0.961004	1.324593	0.913965	1.113803	0.972864	0.954196	1.057146	1.022888	1.023810

Figure 3: Fake news saliency map results, figures are Bi-LSTM, CNN, BERT, RoBERTa in order.



```
predict:      real
ground truth: real
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	Today	is	hard	.	My	chest	feels	heavy	.	My	anxiety	is	through	the	roof	.	Verge	of	tears	.	I	want	this	to	end	already
1	5.082	6.133	4.791	5.349	4.235	5.055	7.048	7.117	5.356	3.756	3.639	4.405	4.011	4.469	4.105	5.234	4.643	3.063	2.047	1.932	1.287	1.101	0.917	0.720	0.558	0.467

```
predict:      real
ground truth: real
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	Today	is	hard	.	My	chest	feels	heavy	.	My	anxiety	is	through	the	roof	.	Verge	of	tears	.	I	want	this	to	end	already
1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	18.666	13.484	0.000	0.000	0.000	0.000	0.000	28.285	21.018	7.485	5.346	0.000	0.000	0.000	0.000	0.000	0.000

```
predict:      real
ground truth: real
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	[CLS]	today	is	hard	.	my	chest	feels	heavy	.	my	anxiety	is	through	the	roof	.	verge	of	tears	.
1	1.000000	0.964550	1.026947	1.030407	0.988842	1.026541	0.999317	0.963758	1.025601	0.923101	1.043955	1.013624	0.898347	0.988121	0.995046	0.972409	0.920151	1.016318	0.996372	0.979884	0.884083

```
predict:      real
ground truth: real
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0		Today	is	hard	.	My	chest	feels	heavy	.	My	anxiety	is	through	the	roof	.	Verge	of	tears	.
1	1.000000	1.942914	0.974672	1.027408	1.016819	1.124612	1.010510	0.994255	1.080357	1.016248	1.059144	0.966424	0.957281	0.982045	1.024299	0.929427	0.978493	1.055715	1.002191	0.973873	1.010321

Figure 4: Real news saliency map results, figures are Bi-LSTM, CNN, BERT, RoBERTa in order.