Report of COMP90042 Project 2020: Climate Change Misinformation Detection

Hualong Deng
Student Number: 1103512
`hualongd@student.unimelb.edu.au`

1 Introduction

Nowadays, we receive lots of information every time, and how to select reliable information from them is significant for us to get true knowledge. In this project, the task is creating a system to distinguish the misinformation about climate change from some text article, and this report presents a system based on the XGBoost algorithm, which could efficiently find the misinformation text.

2 Dataset

There are three data files provided from the project, *train.json* file, *dev.json* file and *test-unlabelled.json* file. *train.json* is used in training process, but there are only 1168 positive documents (misinformation article) in it. *dev.json* is used in evaluating the system in local, and it has 50 positive documents and 50 negative documents. *test-unlabelled.json* has 1410 unlabelled documents and it is for final submission for grade.

The first caveat of this project is that the training data only has positive documents, which is hard to train a reliable model on common methods. To solve this situation, we choose to extend the training set to include documents that do not have climate change misinformation from a reliable and reputable source. Finally, we choose an independent journalism `Guardian` as the source. In this extending process, we should not only collect the documents related to climate change but also the documents unrelated to climate change, because the system should detect the climate change misinformation from all possible documents, no matter they are related to climate change or not. I scrape 1000 articles about climate change and 1000 non-climate-related articles from `Guardian` Australian editions. Some scrapped article has not content because their special html formats can not be scrapped. After filtering, there are 879 climate-change-related articles (*climate.json*) and 862 non-climate-related articles (*no-climate.json*).

3 Models

After data extending, the next caveat of this project is selecting a great model. There are many outstanding classification models, and we finally choose the following models as potential models.

3.1 BERT

Bidirectional Encoder Representations from Transformers (BERT) (Thomas, 2019) is a pre-trained language representations model developed by Google in 2018. It obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks, such as question answering task and natural language inference task. It is trained on a large text corpus (like Wikipedia), and then use that model for downstream NLP tasks. BERT outperforms previous methods because it is the first unsupervised, deeply bidirectional system for pre-training NLP.

In this report, we use Transformer library developed from Huggingface. Transformer library provide a great BERT implementation on Pytorch.[1]

3.2 TextCNN

The convolutional neural network for text, is a useful deep learning algorithm for sentence classification tasks presented by Yoon Kim from New York University in 2014 (Yoon, 2014). It is a convolutional neural networks trained by pre-trained word vectors and achieves excellent results on multiple benchmarks.

In this report, we uses the basic TextCNN model from Denny Britz, which is a great and useful implementation of TextCNN. [2]

[1] `https://github.com/huggingface/transformers.`

[2] `https://github.com/dennybritz/cnn-text-classification-tf.`

3.3 XGBoost

XGBoost (Extreme Gradient Boosting) is a decision-tree-based ensemble Machine Learning algorithm developed by Tianqi Chen from the University of Washington in 2014 (Chen, 2016). The model choice of XGBoost is decision tree ensembles, which consists of a set of classification and regression trees. Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

In this report, we use tf-idf (term frequency–inverse document frequency) of documents as the input of XGBoost model, because it could reflect how important a word is to a document in a collection or corpus. For each document, we will calculate its tf-idf and use this measure to train the model.

4 Method

4.1 Main methods

The target of this project is to detect climate change misinformation and there are two methods to realize it:

The first method is just using one classifier to divide documents into climate change misinformation documents and other documents. In this method, the positive data are documents from *train.json*, and the negative combines *climate.json* and *no-climate.json* two files.

The second method is using two classifiers. One classifier judges if the document is climate-related or not and another one judges if the document is misinformation or not. For first classifier, the positive data are documents from *train.json*, and the negative data are documents from *no-climate.json*. For second classifier, the positive data are documents from *train.json*, and the negative data are documents from *climate.json*. The final result will be the second classifier's result filtered by the first classifier's result. For example, for some unlabelled documents, two classifiers' results are:

- The first classifier's result: [1, 1, 0, 0, 1]

Value 1 means the document is climate-related, and 0 means not.

- The second classifier's result: [1, 0, 1, 0, 1]

Value 1 means the document is climate change misinformation, and 0 means not.

Firstly, we copy the second classifier's result as the final result. For each document whose result is 0 in the first classifier's result, we set its result value in the final result to 0. This process means that if a document is classified to misinformation class, we still need to guarantee it is climate-related. Then, the final result should be [1, 0, 0, 0, 1], because the third document gets 0 value in the first classifier.

In this report, we call the first method as `one-step` and the second method as `two-step`. We will use `one-step` method to select the best model from three potential models and use the best model to compare these two methods' performance.

4.2 Preprocess of input

The document in data is string type. To make the input document more meaningful, we should preprocess the document. In this report, there are following preprocesses transforming the string document into the token document:

- Delete meaningless punctuation
- Transform to lowercase
- Tokenize
- Lemmatize
- Remove stopwords

4.3 Model input length

The length of input affects lots on text model, because it affects how much information the model gets for one input instance. The input instance has more information does not mean that the model will get better performance, so the length of the input is also a parameter for selecting the best model. In this report, the length of input represents how many words in the input document and there are three selected lengths: 128, 256, 512. For example, if the length is 128, then we will select the first 128 words from a document and set it as the representation of this document. If a document does not have 128 words, models will pad it to the required length by adding meaningless elements.

4.4 Evaluation metrics

To evaluate the performance of models, I use models' precision, recall and F1-score as evaluation metrics in this report. All three metrics are based on three models' performance in *dev.json* data.

We choose F1-score as the main metric because it considers the precision and recall meanwhile, and we suppose the model with the best F1-score is the best.

5 Result

5.1 Model selection result

Table 1 shows the performance of BERT model when classifying the documents from *dev.json* data with different input length. The method is `one-step`.

Input length	Precision	Recall	F1 Score
128	**0.652**	0.900	0.756
256	0.575	**0.920**	0.708
512	0.611	0.820	**0.721**

Table 1: BERT model's performance in one-step.

Table 2 shows the performance of TextCNN model when classifying the documents from *dev.json* data with different input length. The method is `one-step`.

Input length	Precision	Recall	F1 Score
128	0.857	0.600	0.706
256	0.755	**0.800**	0.777
512	**0.897**	0.700	**0.787**

Table 2: TextCNN model's performance in one-step.

Table 3 shows the performance of XGBoost model when classifying the documents from *dev.json* data with different input length. The method is `one-step`.

Input length	Precision	Recall	F1 Score
128	0.889	0.640	0.744
256	0.921	0.700	0.795
512	**0.929**	**0.780**	**0.848**

Table 3: XGBoost model's performance in one-step.

By comparing three tables, we could find the XGBoost model with 512 input length gets the best performance in `one-step` method. Then we select XGBoost model to compare `one-step` method and `two-step` method.

5.2 Method selection result

Table 4 shows the performance of XGBoost model when classifying the documents from *dev.json* data with different input length. The method is `two-step`.

Input length	Precision	Recall	F1 Score
128	0.821	0.640	0.719
256	0.864	0.760	0.809
512	**0.909**	**0.800**	**0.851**

Table 4: XGBoost model's performance in two-step.

By comparing Table 3 and Table 4, two methods' best performance are all in 512 input length. We could find `two-step` method has better recall and F1 score, and `one-step` method has better precision. Finally, we choose `two-step` because it has a better F1-score.

6 Error analysis

Obviously, from Table 4, the recall is always higher than precision, no matter what input length is. This situation means that in the document predicted by the model to be positive, the probability of actually being positive is higher. In the actual positive results, the model predicts a small proportion of the results. It shows that the model tends to judge some positive documents as negative.

We should divided error analysis into two parts because of there are two classifiers. One considers the first classifier (if a document is climate-related)'s performance, and another one considers the second classifier (if a document is misinformation)'s.

After comparing the first classifier's result and the true labels of *dev.json*, we find that its performance is very well. There is only one document which is belong to misinformation is judged as a no-climate-related document, which means that this classifier could recognize the climate change misinformation document as climate-relative well. And this result points out that the main error is in the second classifier.

In XGBoost model, we use tf-idf as its input, so a good way to judge this feature is comparing the word frequency between training data. We use the word cloud image to show the word frequency in different data files. Figure 1 represents *climate.json* and Figure 2 represents *train.json*. From

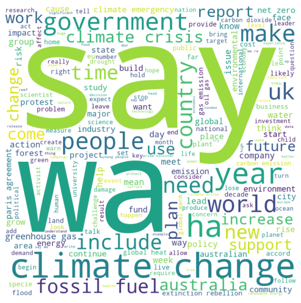

Figure 1: Word cloud of *climate.json*.

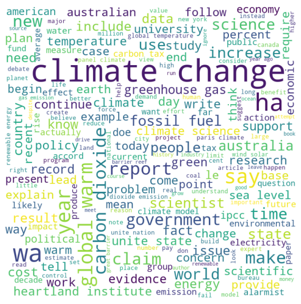

Figure 2: Word cloud of *train.json*.

these two figures, we could find that Figure 1 and Figure 2 are very different, which means the documents from them will have very different tf-idf features, but the second classifier's performance is no well. There may be two reasons for this situation. First one is that tf-idf is not a very good feature for classifying if a climate-related document is a misinformation or not. We need to add other efficient features to represent the document. The second reason is the XGBoost model is not appropriate for the second classification task. We could apply some other models in the second classification to find whether the error is due to XGBoost model.

7 Conclusion

In this report, we present a climate change misinformation detection system based on XGBoost algorithm, which has relatively great performance in this task. This system has two parts. One part judges if the document is climate-related, and another one judges if the document is climate change misinformation. This system finally get 64.8 score in Codalab competition.

References

Thomas W, Lysandre D, Victor S, Julien C, Clement D, Anthony M, Pierric C, Tim R, R'emi L, Morgan F, Jamie B. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv, abs/1910.03771*.

Yoon K. 2014. Convolutional Neural Networks for Sentence Classification. *ArXiv, abs/1408.5882v2*.

Chen, T. Guestrin, C. 2016. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016. ACM. pp. 785–794.*.