Climate Change Misinformation Detecting System

Author: Chuanru Wan
ID: 1074738

1 Introduction

This project is about detecting whether a text is a climate change related misinformation, and label it as positive if it is. Three datasets are provided for training, validation and submit. We have considered three ways to address this project:

- Using one-class classification algorithms by provided train dataset which only contains positive data.
- Using Multi-class classification algorithms by combined train dataset and negative data which is crawling from web.
- Using two models to classify data step by step.

We will describe the details of these methods, respectively.

2 One-Class Classification

Description: One-Class SVM, IsolationForest and LocalOutlierFactor in sklearn are used here. And we have compared their performances.

Texts Pre-Processing: We have clear texts by filtering unexcepted format and removing stop words. And then converting each text into one-hot vector by BOW after counting token frequency.

Train Model: To choosing hype-parameters, we have set serval different parameters for One-Class SVM, IsolationForest and LocalOutlierFactor. After getting the best result of each algorithm, we have compared them to figure out which algorithm performance best in this task.

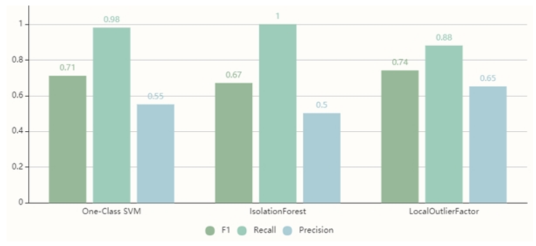


Figure 1: Performance Comparation (dev dataset)

Test Result: After Comparing F1 score (see Figure 1), we used the model of LocalOutlierFactor which got 0.74 under dev dataset to predict test dataset and submitted result onto the Codalab. The Codalab result is shown ins Table 1

F1:	**Recall:**	**Precision:**
0.47	0.94	0. 31

Table 1: Result on test dataset (ongoing)

Result Analysis and Error Analysis: The Recall score is acceptable, but the Precision score is low, and this cause the score of F1 low. We can figure out that almost positive texts are tagged correctly, but this algorithm tagged too many negative texts as positive. I suppose that the model may overfit to the positive texts because there are only positive texts in train dataset. So, I plan to crawl some negative data and design a two-class classification model.

3 Two-Class Classification

3.1 Deep Learning Model Version 1.0

Description: Two deep learning algorithms, LogisticRegression and MultinomialNB in sklearn are used here.

Data Crawling: We have crawled 1400 pieces of text from CNN with keywords, climate change, as negative data.

Texts Pre-Processing: We have clear texts by filtering unexcepted format and removing stop words. Using both CountVectorizer and TF-IDF with parameters, 'ngram_range=(2,3)' and 'max_feature =5000', to generate vectors.

Train Model: Train both LogisticRegression, MultinomialNB models under CountVectorizer vectors and TF-IDF vectors respectively. Cross validation is also applied here to find hype-parameters. I have set a set of value for parameter alpha, [0.001,0.01,0.1,1,5,10], in MultinomialNB. Also, a set of value for parameter C, [0.001,0.01,0.1,1,5,10,50,100,1000], in LogisticRegression. And the best performance of each algorism is shown in Figure 2.

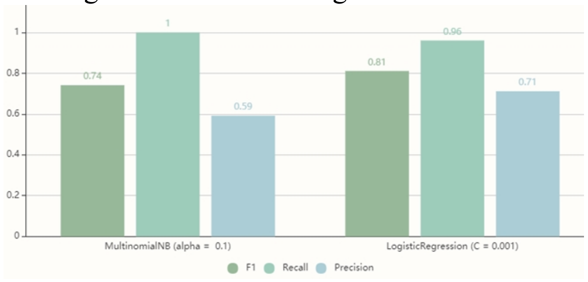


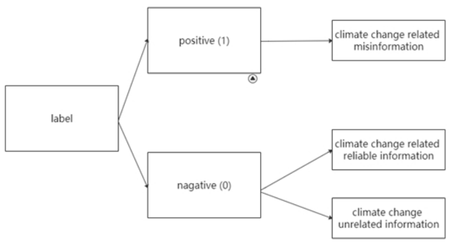
Figure 2: Performance Comparation (dev dataset)

Test Result: After Comparing F1 score (see Figure 2), we used the model of LogisticRegression which got 0.81 under dev dataset to predict test dataset and submitted result onto the Codalab. The Codalab result is shown in Table 2.

F1:	Recall:	Precision:
0.46	0.98	0. 30

Table 2: Result on test dataset (ongoing)

Result Analysis and Error Analysis: The Two-Class Classification model performance better than One-Class Classification model on dev dataset, but their performances are approximately same on test dataset. The problem of the Two-Class Classification model is still mis-labelling negative texts as positive. We tried to solve this problem by analyzing the train dataset, dev dataset and test dataset. We noticed that there are a lot of texts which are not related to climate change in dev dataset and test dataset while all texts in my train dataset are related to climate change. As the requirement of task, the standard of classification should be like figure 3 below.



label
positive (1)
climate change related misinformation
nagative (0)
climate change related reliable information
climate change unrelated information

Figure 3: Classification Standard

As shown in figure 3, texts which are unrelated to climate change should be regarded as negative even it is misinformation. After checking the predict list of dev and test dataset, we found a lot of unrelated texts are mis-labelled as positive. We think this may be the reason of low performance on test dataset. We suppose that if we make the structure of train dataset more similar with the dev and test dataset, the performance of our model should performance better. And there is another problem is we have not done lemmatization in text processing step. Some feature like "word" and "words" should be considered as a same feature "word".

3.2 Deep Learning Model Version 2.0

Description: After the error analysis of Version 1.0, we recollected train dataset which contain both climate change related texts and unrelated texts. And clean data by lemmatization.

Train Model: After training model, we got results of both algorithms shown in Figure 4. As we can see the performance on the dev dataset is extremely similar with last model on Version 1.0, but we think it will performance better in test dataset than Version 1.0 because the structure of test dataset is different with dev dataset, test dataset contains a lot of climate change unrelated texts which can be detected easier by the new model.

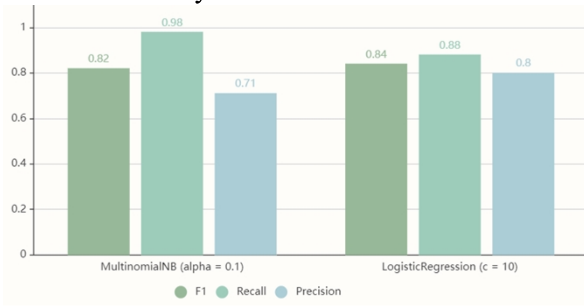


Figure 4: Performance Comparation (dev dataset)

s

Test Result: After submitting our result, the result is shown in Table 3.

F1:	Recall:	Precision:
0.62	0.98	0.46

Table 3: Result on test dataset (ongoing)

Result Analysis and Error Analysis: The F1 score slightly raised from 0.81 to 0.84 on dev dataset which, but it raised dramatically from 0.46 to 0.62 on test dataset. The performance improvements are huge different between dev dataset and test dataset. This situation may be caused by the different structure of dev dataset and test dataset. We think compare to dev dataset which contain 50 positive texts and 50 negative texts, the test dataset may be unbalance which means most of texts in test dataset are negative and a small part of them are positive. Because all methods mentioned above is using BOW to extract features which means the semantic of texts is not analyzed. We try to use some pre-trained word embedding model to extract features and train neural network in the following experiments.

3.3 Feedforward Neural Network + Bert

Description: We used Bert as a Service which can help us translate texts to vectors in serval sample steps and followed the instructor of Workshop 4 to build a sample feedforward model which contain two layers.

Texts Processing: Cleaning the unexcepted format like HTML of texts.

Word Embedding: we started a Bert service by a pretrained model 'english_L-12_H-768_A-12' which is provided by Google and encoded texts to vectors relay on it.

Train Model: After training the feedforward neural network under 20 epochs, we got the results shown below.

Accuracy:	F1:	Recall:	Precision:
0.8	0.83	0.98	0.72

Table 4: Result on dev dataset

Test Result: The result on dev dataset seems good, but when we submitted the labelled test file onto Codalab, we got a result below which is worse.

F1:	Recall:	Precision:
0.51	0.98	0.34

Table 5: Result on test dataset (ongoing)

Result Analysis and Error Analysis: We are confused about this result. It shows that this model performance worse in test dataset. We tried to use another neural network and pretrained word embedding to do analysis.

3.4 TextCNN + Word2Vec

Description: The structure of TextCNN model can be seen in Figure 5.

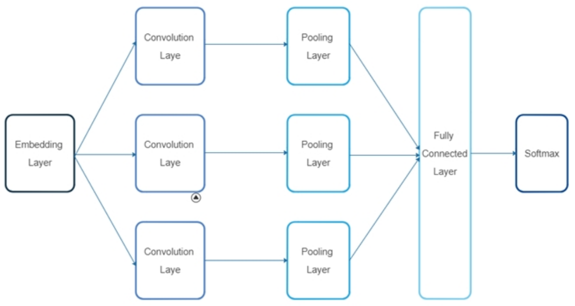


Figure 5: TextCNN

Texts Processing: Firstly, cleaning the unexcepted format like HTML of texts

Word Embedding: I download a pretrained model 'GoogleNews-vectors-negative300.bin' which is provided by google. And loaded it as the embedding layer.

Train Model: After training the TextCNN model with 20 epochs, we got the results shown below

Accuracy:	F1:	Recall:	Precision:
0.75	0.77	0.82	0.72

Table 6: Result on dev dataset

Test Result: The result on dev dataset seems good, but when we submitted the labelled test file onto Codalab, we got a result below which is worse.

F1:	Recall:	Precision:
0.59	0.96	0.42

Table 7: Result on test dataset (ongoing)

Result Analysis and Error Analysis: The results of both two neural network models performance worse than the deep learning model which I mentioned above. Because I am lacking experience with neural networks, I failed to figure out what caused this low performance. I supposed that may be my neural network design is worse. Another reason may be that we did not do fine tuning for the pretrained models.

4 Using Two Models to Do Classification

Description: We decided to back to use deep learning model, but by building two different models to do classification. The structure and process can be seen in Figure 6.

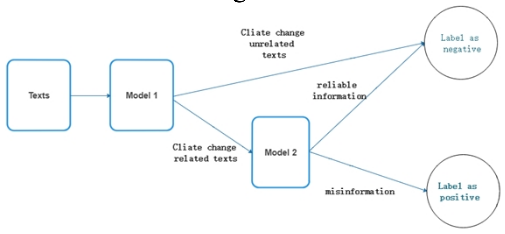


Figure 6: Two model processes

Firstly, we train model 1 by climate change related data with positive tag and climate change unrelated data with negative tag. And then we train model 2 by climate change related misinformation with positive tag and reliable information with negative tag. The model 1 was used to identify whether a text is related to climate change. If a text is identified as unrelated text, it will be labelled as negative. While once a text was classified as climate change related, it will be input into model 2 which classify it whether it is a misinformation. Both models are using LogisticRegression.

Result on dev dataset:

F1:	Recall:	Precision:
0.88	0.92	0.84

Table 8: Result on dev datasets

Result on test dataset:

F1:	Recall:	Precision:
0.66	0.98	0.49s

Table 9: Result on test dataset (ongoing)

Result Analysis and Error Analysis: In this two-model classification system, the model 2 is same with the model in Deep Learning Model Version 2.0 which mentioned above. However, the model 1 using only noun words as feature because noun words are more useful to analyze topics rather than other types of words.

5 Conclusion

Final Choice: After comparing all methods above, we found that the two-model classification system perform best not only on the dev dataset but also the test dataset. In the model 1 of two-model classification system, its accuracy is high as 0.99. As the requirement of task, all texts which are unrelated to climate change should be label as negative whether they are misinformation or not. This design can really help reduce the influence of topic unrelated texts.

Issues in implementation:

1. We found that the results on dev dataset and test dataset are huge different and the results of dev dataset are always good. So, it is hard for us to get enough information from the result of dev dataset. This is also the reason for why we did hill climbing on the Codalab.
2. Lacking knowledge on fine tuning pretrained embedding model. We have tried to fine tune a pretrain model of Bert following the instruction of Workshop 6, but our model label all texts in dev dataset as negative. We do not solve this problem and give up this method.

Further Implementation Plan: I think there are two direction can be applied to improve the performance of system:

1. Try to Crawl train texts from different sources. And to make the composition of train dataset more similar with dev dataset and test dataset. All my current external train texts are crawled from CNN, this may help my system work fine with classifying texts which are also from CNN. However, it will work worse with those texts which have different writing style from texts on CNN.
2. Try to do fine tune on pretrained model of Bert. And apply this word embedding onto a CNN model.

6 Result on Final Evaluation Board

F1:	Recall:	Precision:
0.63	0.96	0.47

Table 10: Result on test dataset (Final Evaluation)