Natural Language Processing Project1 Report

Hongyu Chen
1062897

Abstract

Text classification is a practical task in NLP field. However, in some datasets, the positive samples and negative samples are unbalanced, which is a fatal factor for model generalization. In this project, I used 'Sample Augmentation' to ease the issue above. Additionally, 'Ensemble Learning' is also taken to help the model make better decision.

Introduction

Climate change is a common issue nowadays for all people in this world. However, although this is an obvious fact supported by sufficient scientific evidence, some debates contain too much subjective factors and are not rigorous enough to discuss this problem. Hence, if a model can filter out these texts with misinformation about climate change, it can help a lot to clear the discuss environment, also will bring many benefits in other similar text classification fields.

Related Work

A lot of work has been done before in text classification field. Kim et.al.[1] proposed TextCNN to capture the feature of text segments. Tang et.al.[2] used GRU to construct their model to extract the semantic features in whole context. Lai et.al combined Bi-LSTM and convolutional kernel to get feature vectors of text. More practically, Liu et.al used different architectures of RNN to solve multi-classification task.

Additionally, there are also some ideas been explored about 'unbalanced dataset', such as 'sample enhancement' and 'sample deduction'. In some cases, Generative Adversarial Network (GAN) are used to generate the scarce samples to make the whole dataset be balanced.

My Model

As for my model, there are two main concepts to handle this classification problem. The first concept is known as 'Sample Augmentation', considering the scarcity of negative samples, we can add some negative samples manually to obtain a balanced dataset. The second concept is 'Ensemble learning', to avoid overfitting problem to some degree, we can train several models (each model trained on a part of the dataset) to vote out the final prediction.

I will introduce my project in three parts: model structure, train methods, evaluation and analysis.

Model Structure

The architecture of my model is similar with classical neural network which used to do classification. It contains an embedding layer to encode the input text as fixed length vector, then passes it to later full connected layers (with ReLU activation funtion), finally ends with an output node with sigmoid function. The architecture is shown in Fig1.

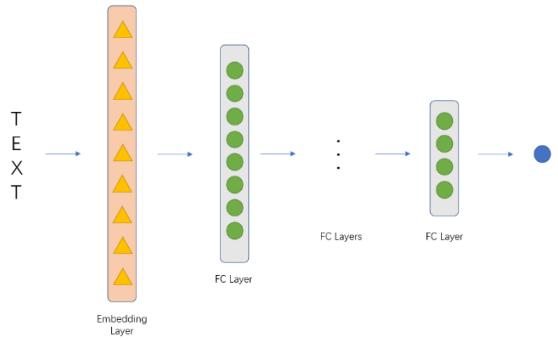

Fig1. Layer construction of the model

Train Method

There are three methods tried to handle this project. The first is a relatively basic method, the second is implementing 'sample augmentation', the third one combines 'sample augmentation' and 'ensemble learning'. The details of these methods will be demonstrated as below.

Method 1

Considering the unbalanced data in "train.json", we can naturally construct a balanced dataset. After observing the syntax structure of negative samples (from "dev.json"), we can manually generate some negative samples with similar structure (normally be human-induced and exaggerated description). Apart from this, I also retrieve some correct climate information from www.ncdc.noaa.gov website. As for the positive samples, I randomly choose some from 'train.json' (but not all, to keep the sample balance of dataset). The additional dataset are stored in "balanced.json".

Now we have a small-scale (around 50 positive samples and 50 negative samples) but balanced dataset to train our model, this method is relatively simple and easy to implement. However, since the train data are manually generated and not sufficient compared with "train.json", there may exist some deviation so it can be regarded just as a baseline.

Method 2

In "train.json", all samples are positive samples, in order to avoid a model which will mark any input as positive, we need to add some negative samples to train. I didn't use extra source from somewhere else, since there are some negative samples already in "balanced.json". For each negative sample in "balanced.json", I generated several similar negative samples from it. The fundamental idea of my augmentation approach is finding synonyms of some core words.

Firstly, for a negative sample (a sentence), we need find the core words that support the semantic meaning of this sentence. The simplest way to do that is using 'regularization rules' to filter out the words contain special symbol, then filter out the 'stopwords'. After that, we can obtain a bunch of words occurred in target sentence, then for each words, we can set a probability (such as 0.5) to alter it with a random choice from its synonym set (can be obtained from NLTK package).

In a word, initial sentence will be transformed on the word level, however, the semantic meaning is almost remained unchanged. After the augmentation, the proportion of negative samples and positive samples is around 1:2, which is a decent value of a normal dataset.

Now, on augmented dataset, we have sufficient samples to train our model without worrying the model will mark all the input as positive. The process is demonstrated in Fig2.

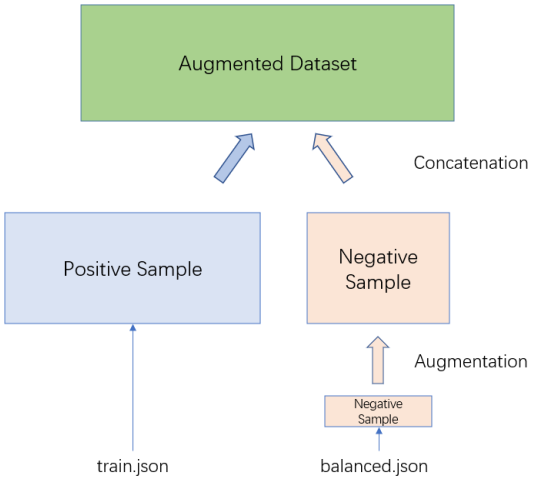

Fig2. Augmentation process

Method 3

Although method 2 eases the unbalance issue to some degree, the model still risks being overfitting. Hence, I use the idea of 'Ensemble learning' to mitigate this problem. Benefitted from the bagging idea, I trained 5 models to vote out the final prediction. The details are illustrated in Fig3.

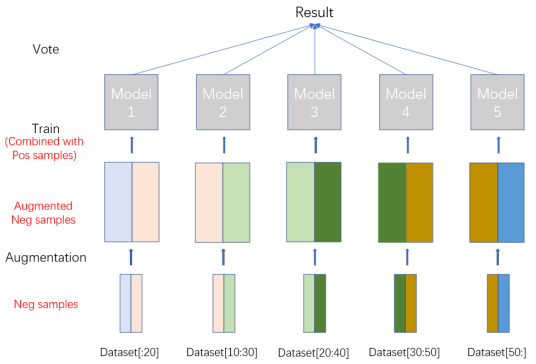

Fig3. For each model, their negative train samples are part of "balanced.json", the extra dataset (the negative train samples of 5 models should have overlapping areas), then the data will be augmented to make sure the model sufficiently trained. Finally, the model will vote for the final prediction.

Details

During the training process, there are also some hyperparameters need to be tuned.

After several experiments, the dropout probability of the Full Connected Layer should not be greater than 0.4, best performance can be obtained during 0~0.2.

Considering the scale of dataset, the training batch should be 20~40, greater will make model omit something and less will slow down the training speed.

Normally, under my experiment settings. The model will convergence after 25 training epochs, so the best training epoch should be set as 30.

For the sigmoid output layer, the default classification threshold is 0.5, but I add 0.33 and 0.66 as a comparison with 0.5. In fact, this measure indeed brings me excellent results.

Evaluation and Analysis

The results of each method are shown in the table1 below.

Method	Precision	Recall	F1
Baseline	0.52	0.56	0.54
M1_0.33	0.75	0.84	0.79
M1_0.5	0.79	0.68	0.73
M1_0.66	0.8	0.56	0.65
M2_0.33	0.53	0.92	0.67
M2_0.5	0.53	0.92	0.67
M2_0.66	0.54	0.92	0.67
M3_0.33	0.60	0.90	0.72
M3_0.5	0.63	0.91	0.74
M3_0.66	0.59	0.88	0.71

Table1. Results of each method experimented on 'dev.json'. For the name of method, first part represents which method I take, the second part is the threshold I choose in the sigmoid layer.

From the results, all three methods have better performance compared with baseline. However, for method 1, since it is trained on

a small-scale dataset, the results will fluctuate obviously if we change the sigmoid threshold. But in total, it indeed finds some relationship between training data and has good performance.

As for method 2, it seems more likely to mark a sample as positive, that's why precision is largely lower than recall. I think the main reason is that negative samples are not generalized enough. During the augmentation process, we just make changes on word level, hence, the generated samples is largely similar with initial sample to some degree. This problem is also known 'mode collapse' in GAN field.

For method 3, it still suffers from the naïve augmentation to some degree, however, benefited from 'ensemble learning', it is more robust and can obtain better decision.

In conclusion, 'sample augmentation' and 'ensemble learning' are the main ideas implemented in my project. The result may not at the perfect level, but it still has significant improvement compared with baseline.

Future Plan

Although I make obvious improvement on baseline. I think the result can be elevated to a higher level.

Firstly, there could be more layers in model structure to extract more and deeper features of text segment. Additionally, RNN is not included in my model, I think this architecture could be tried to handle sequential input such as text. Finally, I plan to use GAN as a tool to do 'sample augmentation', because with its strong generation ability, GAN should have a better performance on this task thus leads to a better result.

Reference

[1]. Kim. Convoutional Neural Networks for Sentence Classification (EMNLP 2014).

[2]. Tang. Document Modeling with Gated Recurrent Neural Network for Sentiment Classification (EMNLP 2015).

[3]. Lai. Recurrent Convolutional Neural Networks for Text Classification (AAAI 2015).

[4]. Liu. Recurrent Neural Network for Text Classification with Multi-Task Learning (IJCAI 2016).