C490H-Coursework OffenEval

Introduction

The competition OffenEval aims at using natural language processing methods to identify offensive language in social media. It includes 3 tasks: offensive information identification, offense types categorization and offense target identification.
Text classification is a core research field in natural language processing and there are many state-of-the-art classification models includes FastText, TextCNN, HAN, DPCNN. We apply convolutional neural network methods, taking use of pretrained GloVe word embeddings, based on Pytorch, to complete binary classification (task 1 and 2) and multi-label classification (task 3) and achieve good performance on the tasks.

Dataset infomation

Table 1. Basic dataset infomation

Task A	No.	Ratio	**Task B**	No.	Ratio	**Task C**	No.	Ratio
Not offensive	8840	66.8%	Targeted	3876	88.1%	Group	1074	27.7%
Offensive	4400	33.2%	Untargeted	524	11.9%	Individual	2407	62.1%
						Other	395	10.2%
Total	13240		Total	4400		Total	3876	

Based on Table 1, task A and task B are binary classification problem and task C is multi-class classification. Some strategies, such as RNN, CNN, SVM, random forest and etc, are commonly used in similar language classification tasks. Although three tasks may have some internal relationships, we treat three tasks independently for simplicity which means we will not train the task B model using not offensive language in task A.

Noticed that the provided dataset is unbalanced for all three tasks especially for task B which has 88.1% positive samples. This unbalanced problem could extremely effects the performance of the model so some upsampling/downsampling strategies or parameter tuning are required.

Preprocessing

1. **Remove stop words.** Since the texts is coming from tweet, lots of casual expression and special symbols exist in the dataset and those noise may affect the performance of the model. Therefore, stop words including punctuation, emoji, '@USER', meaningless words are removed.

2. **Build offensive language dictionary.** The language dictionary is used to support the model. Intuitively, this will be an important feature in task A but due to the cryptic expression and casual language in tweet, the improvement of the dictionary is limited.

3. **Oversampling and undersampling.** Due to the unbalanced dataset, data augmentation is required. Firstly, the downsampling is tested in task A. We sampled 4400 samples in the Not offensive class and train with the same lightgbm model with same parameters and we found the f1_score of the original dataset is higher that means throw out 4400 samples sacrifice too much for balancing the data. The similar situation happens in task B and task C so we decided not to implement under-sampling. Then we tried to upsample the minor class using the SMOTE in imblearn package. The idea is to sample some text in minor class, adding some random noise and modifying by some rules. However, it is not easy to implement to the text and word embedding matrix so we finally decided to modify the threshold in the final classifier to reduce the effect of unbalanced data.

Word Embedding

Traditional word vector learning methods like skip-gram cannot utilize information of the whole corpus as a result of training on separate context and latent semantic analysis is not good at word analogy. GloVe solves these problem based on global information and is essentially a log-bilinear model with a weighted least-squares objective. The model rests on a rather simple idea that ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning which can be encoded as vector differences.
In our model, we take use of pretrained GloVe word vectors glove.840B.300d to build the vocabulary of training set and for the purpose of model input.

LightGBM model

LightGBM is a gradient boost framework based on decision tree. Since it implements some cutting algorithm and using distributed strategy, it has high train speed, high efficiency and low memory usage compared with XGBoost. [1]

Features used in this model include sentence length, terms frequency (HashingVectorizer), tf-idf (TfidfVectorizer) and whether the sentence have a word in offensive word dictionary. Those features are stacked into a compressed sparse row matrix and feed into the lightgbm model.

The macro F1 score in task A using this model with threshold = 0.39 is 0.695 which is lower than the textCNN model so we decide to use the textCNN model for the following tasks.

TextCNN model

1.TextCNN for classification [2]

TextCNN is a useful deep learning algorithm for sentence classification problems,such as sentiment analysis.In practice, we apply word-level neural network for this classification task.

2. Model and traning configuration

Here is the architecture of our two-layer CNN network:

Architecture of Task A & B:

embedding(embedding_dimension = 300)		
conv(kernel_size = 3)	conv(kernel_size = 4)	conv(kernel_size = 5)
dropout(drop_probability = 0.5)		
max-pooling		
fully-connection(output_dim = 100)		
dropout(drop_probability = 0.5)		
fully-connection(output_dim = 1)		
sigmoid		

Architecture of Task C:

embed(embedding_dimension = 300)		
conv(kernel_size = 3)	conv(kernel_size = 4)	conv(kernel_size = 5)
dropout(drop_probability = 0.5)		
max-pooling		
fully-connection(output_dim = 100)		
dropout(drop_probability = 0.5)		

fully-connection(output_dim = 3)
softmax

The network structure of Task A & B includes three convolution layers with different kernel_sizes,out_channel 100,followed by dropout layer.Then max-pooling and concatenate them on dimension 1 and do two layer fully connection. Network structure for Task C is similar but changes the output dimension to 3 as well as output activation to softmax.

By using GloVe, the embed layer translates each word into a 300-dimension vector and this embed layer is static during the whole training process. For all dataset, we use stochastic gradient descent with a mini-batch size of 128 and Adam optimizer.

Specifically, we separate our dataset into training dataset and validation dataset (9:1), and training model on training dataset for 100 epoches and after each epoch we calculate the performance on validation dataset. If the current epoch performance is better than previous one, we save the better model. After 100 epoches, we use the best model to test on our validation dataset and choose the hyperparameter threshold of sigmoid function. We search threshold from 0.1 to 0.5 and choose the one with the best performance (macro F1).

3. Performance

Task A(threshold = 0.44):

	precision	recall	F1
not offensive	0.83	0.88	0.86
offensive	0.72	0.65	0.68

macro F1	0.77

submission results(Team name:ShawnGung):

macro F1	0.77573591

Task B(threshold = 0.32):

	precision	recall	F1
un-target	0.37	0.12	0.18
target	0.87	0.97	0.92

macro F1	0.55

submission results(Team name:ShawnGung):

macro F1	0.60777905

Task C:

	precision	recall	F1
individual	0.77	0.88	0.82
group	0.53	0.48	0.51
other	0.53	0.21	0.3

macro F1	0.54

submission results(Team name:ShawnGung):

macro F1	0.53335832

Discussion

The reason why the TextCNN model performs better than the lightgbm model is that it uses the embedded word vectors as feature which potentially contains much more features and information than the statistic features such as tf-idf. We attempted to encoder each sentence into a 1*300 vector by combining all the word vector using root mean square in the lightgbm model to improve its performance. However, this method introduces lots of noise and discard the sequence information so its performance still worse than TextCNN.

By carefully tuning the threshold value to deal with the unbalanced problem, the final results of TextCNN is better than the provided baseline scores. However, the macro f1 score for Task B and Task C are still much lower than Task A, same as the baseline indicates. This indicates that more techniques for dealing with the unbalanced text dataset are worth investigating.

Future work

1. Implement some proper data augmentation strategy to the text or text vector in order to balance the dataset.
2. Feature engineering. Try to extract more features,for example sentiment behind emoji, from the tweet text and use the embedding model trained from tweet.
3. Model ensemble. In this task, we implemented some models and classifiers independently. Probably, some ensemble strategy could improve the final score.

Reference

[1] LightGBM's document, available at: https://lightgbm.readthedocs.io/en/latest/
[2] Yoon Kim, *'Convolutional Neural Networks for Sentence Classification'*, New York University. Available at: https://arxiv.org/pdf/1408.5882.pdf

Repository

link: https://gitlab.doc.ic.ac.uk/zg18/NLP
ssh: git@gitlab.doc.ic.ac.uk:zg18/NLP.git