COMP90051 Statistical Machine Learning, Semester 2 2019

Project 1: Who Tweeted That?

Fan Yang, Yinghjao Zhu, Xinhao Zhang

1. Introduction

Authorship attribution is to find the writer for a given unknown text. It can be applied in many fields like identifying ancient texts and plagiarism detection. The project is about to find the author for a tweet by some machine learning methods. We explore three approaches to accomplish this goal: Naïve Bayes classifier, Word2Vec with Convolutional Neural Network (CNN) and Bert with Softmax classification. Except the Naïve Bayes classifier, the other two methods both have decent performance. The final architecture which we use in the project is the Bert with MLP and its accuracy is 18.5%.

2. Data & environment

Data used in this project is from Kaggle, including "train_tweets.txt" including 328k labelled tweets and "test_tweets_unlabeled.txt" including 35k unlabelled tweets. The labelled data set is split into two parts whose size ratio is nine to one. The larger one is used as training set in this project while the smaller one is development set. Additionally, the unlabelled data is test set which is used to predict labels.

This project is mainly written on ipython notebook and run on colab provided by Google because on colab we can use GPU for free and even eight core TPU to promote training efficiency. Additionally, the memory used in this project is 25 gigabytes.

While doing feature engineering, we first inspect the training data set. It can be found that a user tends to compose tweets with smiling face, e.g. ":)". Therefore, we try to look for various features from training data set, including number of URLs, "RT"s, emotions, mentions, hashtags, cash mentioned, percentages, and capitals in tweets. Furthermore, we have observed that the punctuations, and more importantly, the meanings and sentiments of tweets potentially indicate the authors of the tweets.

As for the first part of features, they might be used to classify authors clearly. For example, some twitter users love using all capital letters while some others often retransmit tweets. Moreover, these signs are difficult to be converted into vectors in the following methods. Therefore, they are extracted with regular expression and store in arrays to be concatenated later. . As we decided to use neural networks, which will be discussed later, these features are added into the feature vector one after another to check if they affect the results during the training process. Then, we find that adding these features can more or less improve the result. However, the feature – number of capitals – can be improved because it ignores the length of tweets. In this case, it is changed to the fraction of capitals in tweets to identify those who edit tweets with almost all capital letters. Finally, these features are extracted and deleted from the tweets so the rest of them are tokenized to extract the second part of features.

BERT, Bidirectional Encoder Representations from Transformers has achieved state-of-the-art performance in wide range areas of Natural Language Processing[4]. The self-attention mechanism of BERT is really useful in extracting the relationships between words and the structure of a sentence. Hence, the pre-training encoder part of BERT always used for word embedding. Furthermore, the encoder model of BERT works well in understanding the meaning of a word according to its context while embedding models like Word2Vector always return same word embedding even though in different context. For author identification in this project, there are many features of tweets may represent characteristics of authors. For example, the structure of a sentence (some users may tend to use more clauses while others not), the emotions of tweets (someone always negative while others positive). In order to optimize the extraction of these potentially useful features in this project, BERT is used for sentence Embedding.

4. Model

The Naïve Bayes classifier is a probabilistic model. It assumes that all features are independent and simplifies the probability calculation based on the Bayes' Theorem. $p(X|C) = \prod_{i=1}^{n} p(X_i|C)$, where X is the feature vector of the instance and C is the class. In the project, we first attempted to use a compressed inverted index to build a Bayes model. We extract features of lemmatized words from the training set. Those features are combined into an array with the length of 118 and used to train the Naïve Bayes classifier. The result for the classifier is bad. The accuracy for identifying tweets in the test set is around 0.66%. This might be caused by the unproper feature selection. The number of features is small, while the number of predicted classes is massive. These might also make the Naïve Bayes classifier have poor performance. It cannot be used as a baseline in this project with such low accuracy.

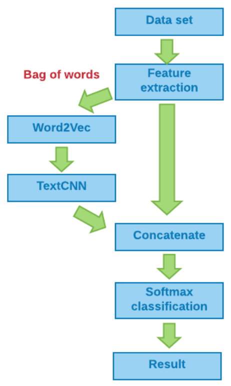

Figure 1 Word2Vec and TextCNN

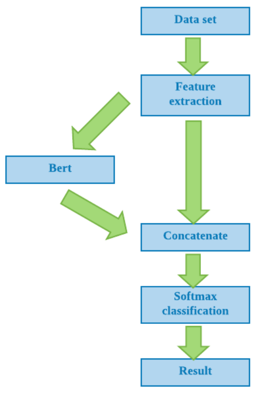

Figure 4 BERT with SoftMax classification

One neural network model we have tried is combining the method of Word2Vec and CNN (Figure 1). After eliminating the unwanted formats, such as URLs and punctuations, and extracting the features preliminarily, we generate vectors by Word2Vec. Word2Vec[1], raised by Mikolov, convert words into vector considering the context. Then, text CNN is used to map a tweet into vector. In Kim's paper, text CNN[2] is used for sentence classification. It consists of embedded layer, convolution layers with filter windows of 3, 4, 5 and max pooling layers, which extract features from each row and then concatenated. Finally, the vectors are concatenated with the corresponding features from tweets and then fitted into the softmax classification layer. We trained the model with batch size of 16 and 10 epochs. As a result, the accuracy is around 10%.

The final model that we apply in the project is the Bert with SoftMax classification (Figure 2). The first step is the same as Word2Vec with TextCNN, we extract those features from our training set. Instead of using an embedding layer in the neural network to transform words into vectors, the Bert with pre-trained model can generate a vector for a given whole sentence. Therefore, we use it to embed tweets in the training set and the output is a 768-dimension vector. Then, we concatenate it with those important features which we get from the former step. The inputs for the classifier are 789-dimension feature vectors. The activation function is Softmax and the loss function is cross-entropy. We set the batch size as 64 and train it in 10 epochs where the validation accuracy seems to be stable. It has the best performance among three models and the accuracy for the test set is 18.5% which is larger than the baseline (16%).

5.Analysis

As discussed above, a model consisting of BERT and a classification layer is selected because of its higher performance showed below. In this experiment, Word2Vec & TextCNN and BERT used Adam optimizer whose result is shown in Table 1. BERT has a higher accuracy because BERT can extract more features than Word2Vec, understanding not only the meaning of a word, but the structure of sentences. Therefore, from the sight of features, BERT works better than Word2Vec.

As displayed in figure below, in terms of model BERT, the train loss dropped largely as number of epochs increased, but validation loss just decreased at the first two epochs and gradually grew after that. This means overfitting occurred. For overfitting, regularization and dropout layers are utilized. However, we observed that if these methods were applied, the model cound not fit the data as well as that did not include these methods. Specifically, the accuracy of BERT model with regularization was around 13 percent whereas the model without regularization could reach 18.5 percent. Another case of model using SGD optimizer in which decay of learning rate was set as 10^{-7} and momentum was 0.9, both train loss and validation loss declined but the validation accuracy rose much slower than Adam. It mainly because This after a large number of training epoch, the learning rate is reduced to extremely low, which is not enough to adjust the parameters. Although it seems that overfitting had less impact on model with SGD optimizer, the accuracy was around 18 percent after the model was trained 30 epochs. Similarly, if regularization or dropout layers were added, the accuracy would not reach 18 percent.

	Bayes	Word2Vec & TextCNN	BERT
Accuracy(%)	0.68	10	18.5

Table 1 Result of Bayes, Word2Vec & TextCNN, BERT

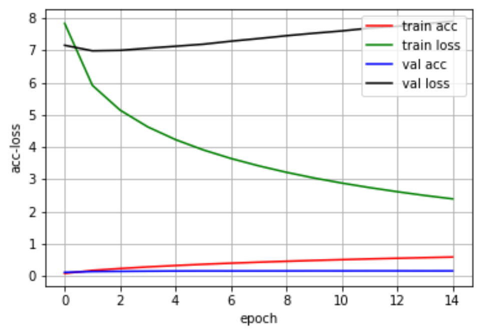

Figure 3 BERT trained with Adam

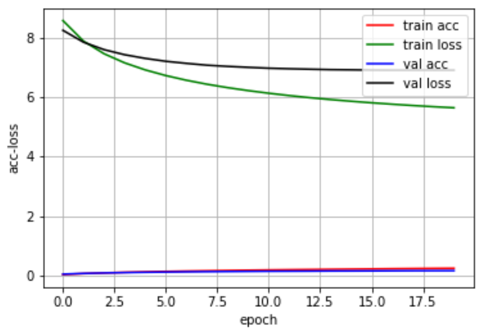

Figure 4 BERT trained with SGD

Comparing SGD and Adam optimizer, we found that the model with Adam optimizer would be trained more efficiently because it combines the ability of AdaGrad that tackles sparse gradients, and the ability of RMSProp which deals with non-stationary objectives[3]. Meanwhile, model with Adam may overfit after a few epochs, so we used EarlyStopping to stop training if the validation loss increased consecutively. Therefore, we selected model with Adam optimizer and it was trained eight epochs.

As described, if regularization or dropout layer were added, the performance of the model may be reduced even if λ in regularization was set to be a small number, e.g. 10^{-6}. Therefore, trade-off between regularization and model fitting needs more research in the future. In this project, we set λ in regularization as 10^{-8}.

In the future research, we plan to look for more useful features, such as words not in dictionary fitting the data well. Furthermore, the structure of the model and the method dealing with overfitting needs improving. For instance, we can train our own BERT model instead of the pretrain model, which also requires a large amount of memory.

Reference

[1] Le, Q., & Mikolov, T. (2014, January). Distributed representations of sentences and documents. In International conference on machine learning (pp. 1188-1196).

[2] Kim, Y. (2014). Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.

[3] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[4] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.