SDML Assignment #1 2019 AI Cup

陳子恆
r08922030
r08922030@ntu.edu.tw
鄭淵仁
r08933067
r08933067@ntu.edu.tw
林承德
r089220a4
r089220a4@ntu.edu.tw

20, October 2019

1 Introduction

The first assignment in Special Directions on Machine Learning class is going to join the AI cup competition. With four weeks' struggle, we totally tried 99 different models, not only the state of the art technology such as Bert, XLNet, but also many classic methods' improvements. Among all the models we tried, the best one is the ensemble of XLNet and Bert. Its performance is up to 0.702956 which is strongly improved from our teaching assistant's GRU model. In the following section, we will briefly define the problem that we deal with. Also, will describe the method we tried and its' performance in section 3 and 4 respectively then finally end with our conclusion in section 5.

2 Problem Description

The competition aims at designing a system that could automatically classify a thesis into four classes. Note that a thesis may have multiple classifications.

The competition provides Title, Abstract, Authors, Categories, Created Date, and Label of a thesis, and we have to classify it into the following classes: Theoretical Paper, Engineering Paper, Empirical Paper, and Others.

Additionally, this assignment provides the extra data, Citation Networks, which states the reference connections of each thesis.

3 Methods

In this section, we describe the methods we have tried to classify the thesis. The methods could be divided into the following three parts, prepossessing, models, and some tricks.

3.1 Preprocessing

For the data from the competition, we developed two types of preprocessing: One is treating the abstract as a long sentence. Another one is treating the abstract as a list of sentences, that is, we first split the abstract by '$$$', and then tokenize it.

Note that when tokenizing sentences, we use the tokenizer from the corresponding algorithm. For example, when using Bert, we use the tokenizer from Bert.

After tokenizing sentences, we use the embeddings from Google's pretrained Word2Vec and GLOVE to convert them into vectors.

For the extra data from this assignment (citation graph), we use a novel graph embedding method "PRUNE" which is proposed by NTU, to convert them into graph embeddings.

Besides abstracts and title, there are 3 fields (Authors, Categories, Created Data) which can be used as categorical features. We convert each features from these three fields to id and remove the features which only appear less than 5 times. Finally, there are 585 unique features from these 3 fields and we use it to train a simple logistic regression model as our

3.2 Models

We have tried several models in this competition. Our models can be divided into two group, which are tradition machine learning model and Neural Network model. Traditional machine learning model includes Logistic Regression and Tf-idf while neural network model includes GRU, LSTM, Bert and XLNet. All the objective function in the neural network model is the binary cross entropy (log-loss) function The results of each model will be shown in next section while the details of our model are shown as follows:

3.2.1 Logistic Regression

We only use the categorical features from Authors, Categories, and Created Date to train a simple Logistic Regression model as our baseline. Each paper is represent by a one hot vector x, if i-th feature appears in the paper, then $x[i] = 1$. Otherwise, $x[i] = 0$.

3.2.2 Tf-Idf + Support Vector Machine

Support Vector Machine (SVM) always has good performance in classification task. We use abstracts to calculate tf-idf for each paper. For example, if there are 100 unique vocab in all the data, include training data and testing data, then each paper can be represented as a vector x. The length of x is 100 and $x[i]$ is the tf-idf value of i-th vocab.

3.2.3 Simple GRU and LSTM

We split the abstract as a list of sentences, use Word2Vec and GLOVE to convert the sentences into embeddings. Then feed them into a RNN model which is GRU or LSTM in this case. Next, a max-pooling is used over all embedding dimensions, followed by a max-pooling over all sentences in an abstract.

Since GRU and LSTM could not learn long sentences very well due to their limitaton of structure, we split the abstract into a list of sentences instead of consider the abstract as a long sentence.

3.2.4 Bert and XLNet

We represent the abstract as a long sentence. Then, feed them into the state-of-the-art model, such as Bert and XLNet. Next, a max-pooling layer is used over all embedding dimensions and use a perceptron to classify the abstract into four categories. To prevent overfitting, we use the dropout before feeding the embedding into perceptron.

Besides, we also try to split the abstract into a list of sentences and convert the list of sentences into Bert or XLNet embeddings. After that, we add a max-pooling layer over all embedding dimensions first, and the second max-pooling layer over all sentences in the abstract.

3.2.5 Concatenation of Graph Embedding and XLNet embedding

This model is similar to the model above. The only difference between these two model is the graph embedding and XLNet embedding are considered in this model instead of using XLNet embedding only. We concatenate the XLNet embedding and Graph Embedding as input to the perceptron. To prevent overfitting, we also use the dropout before feeding the embedding into perceptron.

Suppose σ is the sigmoid function, x is the concatenation of XLNet embedding and graph embedding, x_i is the XLNet embedding for i-th token, g_j is the graph embedding for j-th paper. Then the model's predictied probability p can be expressed as

$$p = \sigma(Wx + b)$$
$$x = [x_1, x_2, ..., x_n, g_j]^T$$

3.3 Tricks

We use the following tricks to boost up our performance.

3.4 Separate training

There are four categories in the data. We can use a model to predict four labels and the loss function can be the mean of log-loss in each category. But instead of using one model to train four categories, we use four models to predict each category. So, the loss function of our model is the log-loss of one category instead of the mean of log-loss in each categories.

3.4.1 Threshold

We did not use the ordinary 0.5 as the threshold to transfer the number outputed by the perceptron into 0-1 classes. We used validation data to find the best threshold. The reason why we didn't use 0.5 is that the Binary Cross Entropy does not match F1 Score, because Binary Cross Entropy counts True Negative as a good sample, but F1 Score doesn't. By this trick, we could obtain a better F1 Score.

3.4.2 Ensemble

We ensemble several models. That is, we sums up the output from the perceptron in multiple models instead of one. And then use a threshold to transfer them into 0-1 classes. By this trick, the performance could be better and stabler.

Category (C)	Number of Positive Label (P)	Positive Rate (P / 7000)
THEORETICAL	3218	0.4597
ENGINEERING	3391	0.4844
EMPIRICAL	2140	0.3057
OTHERS	259	0.037

Table 1: Analysis of the positive label in training data.

3.4.3 Loss Function

We analyse the the number of positive label in training data and the results are shown in table 1. From this table, we find out that the labels are imbalanced in all categories, especially in the "EMPIRICAL" and "OTHERS" categories.

Therefore, we need to make some changes to the original log-loss function to help our model can learn well from these imbalanced data. A weighted log-loss function should consider the class weights and use it to penalize errors on minority class (which is negative label in our case) harder.

Mathematically, the original log-loss function is

$$-ylog(p) + (1-y)log(1-p)$$

To consider the class weights in the loss function, we change the loss function into

$$-ylog(p) + w * (1-y)log(1-p)$$

y is the ground truth label which is obtained from training data, p is the model's predicted probability and w is the class weights.

3.4.4 Calibration

In the Section3.3.3, we explained about our loss function by introducing the class weights. This loss function can help our models to learn well but it also introduce another problem, which is underestimating the probability of each class.

For example, if we set the threshold to 0.5, and we underestimate the probability of paper A in category "ENGINEEING" to 0.4. After calibration, the predicted probability is adjusted to 0.6, this changes will affect the final results of f1-score.

Many researches have shown that calibration is very important in the classification model but most of the classification model are poorly calibrated. One of the papers about calibration will be shown in the reference.

Suppose σ is the sigmoid function, b is the bias, p and p' are the predicted

original prediction	always predict 0
0.7299016772	0.7303240740

Table 2: The Impact of treating the abstract as a long sentence or a list of sentences

probability before and after calibration respectively, then the formula of calibration is:

$$p' = \sigma(\sigma^{-1}(p) + b)$$

Use another model to minimize the original log-loss function

$$-ylog(p') + (1-y)log(1-p')$$

b is the only trainable variable in this calibration model. Finally, we will obtain p' and use p' to calculate f1-score.

3.4.5 All-0 in 4-th Dimension

If we print out our training data's label, we can easily observe that on the 4-th dimension, there is only 3%'s 1 and 97%'s 0 which is really imbalance. Thus, a bold idea came into our mind. We did an experiment about "what will happen if we always predict 0 in 4-th dimension." And the experiment result is show in table 2.

4 Experiments

4.1 Performance of Each Model

Table 2 shows the performance of each model. The symbol '-' in the third column indicates that we did not upload the model to the T-brain due to the limited upload times.

From this table, it shows that logistic regression (LR) obtains the worst performance on validation data. It is reasonable that the performance of Support Vector Machine (SVM) is better than LR due to the higher complexity of SVM.

It is also very interesting to find out that all the neural network model are better than tradition machine learning model. The f1 score of Bert-large, XLNet-based, Ensemble(Bert-large, XLNet) are at least 0.1 higher than the f1-score of SVM. It can be seemed as a significant improvement in this task.

The performance of LSTM is similar to the GRU, both of them almost have

Model	F1 score (Val)	F1 score (Testing)
LR	0.6244	-
Tf-Idf+SVM	0.6316	-
RNN (LSTM)	0.673	-
RNN (GRU)	0.6768	0.6802
Bert-based	0.6885	0.6492
Bert-large	0.7335	-
XLNet-based	0.7452	-
Ensemble (Bert-large + XLNet)	**0.7536**	**0.703**

Table 3: Performance of each model

Model (Bert-based)	F1 score (Val)	F1 score (Testing)
View as a Long Sentence	0.6748057713651499	-
View as a List of Sentences	0.6870562534134352	-

Table 4: The Impact of treating the abstract as a long sentence or a list of sentences

the same validation f1-score. Then it is also reasonable to find out that Bert-large model is better than Bert-based model on validation data. This is because Bert-large model have more parameters and higher complexity than Bert-based.

XLNet is a model which is recently proposed by Google and they claimed XLNet outperform Bert in 20 NLP tasks. In this classification task, XLNet also shows a better performance than Bert-based and Bert-large.

The best model is the ensemble of Bert-large and XLNet. The validation f1 score is 0.7536 wile the testing f1 score is 0.703

4.2 Performance of treating an abstract as a Long Sentence or a List of Sentences

In table 3, we trained the two model both with BERT, but these two models preprocess the abstract into a long sentence and a list of sentences respectively.

From table 4, we observed that the performance of a list of sentences performs better than as a long sentence. However, after the presentations of the top-10 group in the class, we think that the randomness due to the different validation seed may influence the results. Hence the results might be controversial.

Model	F1 score (Val)	F1 score (Testing)
XLNet-based	0.6901	0.6535
XLNet-based (with graph embedding)	0.709	0.6483

Table 5: The Impact of Graph Embedding

4.3 Performance of Graph Embedding

We have tried adding graph embedding into our models in the beginning of the competition. However, in the first few weeks, we fail to well-tuned the XLNet and Bert model. Therefore, the results shown in table 4 have some difference with the results of Table 2.

Before the experiment, we believe that graph embedding can enhance the performance of XLNet. But from table 4, we realise that by using XLNet embedding and graph embedding, the model has a higher f1-score on validation data but a lower f1-score on testing data. In our opinions, there are two reasons that can cause this unexpected results. First, the validation data cannot represent the distribution of testing data. Therefore, the f1-score on validation are higher but the f1-score on testing is lower. Another reason is the graph embedding only contains a bit of information about paper. This is due to the paper only has one or two citations and cause the information of this paper on the graph is not sufficient.

5 Conclusions

In this assignment, we have tried many models to get the better performance. We have met many challenges in this assignment, especially in choosing the validation data and hyperparameters.

We realise that the distribution of validation data is very important. A good validation data can has the same performance as testing data. In this assignment, we have tried several validation data but we still fail to find a good validation data. In table 2, the validation f1-score of the best model is 0.7536 but the testing f1-score is 0.703. There is a big gap between the performance of testing data and our validation data.

Tuning the model is also a difficult task in the machine leaning. Due to the limited time and limited resources, we cannot test all combinations of hyperparameters, therefore how to choose a good hyperparameters isa a big challenges for us.

From this assignment, we find out that the performance of high complexity model such as Bert and XLNet are better than low complexity model such as LR, SVM. This is the first time we writing our own code to see the results of

these models instead of looking the results from original paper. Although we do not get top-10 in the class, we still enjoy this assignment and learn many skills from it.

References

On Calibration of Modern Neural Networks
`https://arxiv.org/pdf/1706.04599.pdf`