Author Attribution with Deep ANN and TF-IDF

Hanyu Zhou, Zixuan Guo, Shengqi Ma

1 Introduction

The problem is authorship attribution, which identifies authors for a given paper. Normally there is a pool of candidate authors, and true authors are predicted through machine learning methods. Authorship attribution has many applications. It obtains the time in which the paper was written, is used to develop defense methods to combat authorship attribution for people who shared their documents anonymously and reveals true authors of papers for plagiarism detection.

2 Preprocessing of Dataset

The training dataset is split into two datasets, 80% is the training dataset, 20% is the validation dataset, and prolific authors' and coauthors' values are extracted. In the next step, it keeps 25% of such papers in training set by selecting 7,460 papers with prolific authors, copying them ten times, then adding all the papers without prolific authors into the training set and shuffling them at the end.

In preprocessing, it also uses the Term frequency-inverse document frequency (TF-IDF), a weighting factor showing the importance of one word to a document in a collection, to make the feature selection. To be more specific, it applies TF-IDF to the feature of 'abstract' and 'title' to obtain the importance of each feature element, and then keeps relevant feature elements in the training set and deletes unimportant ones, which increases the efficiency of the model. By the way, the applied TF-IDF was originally developed since the package of TF-IDF in Python does not fit the training data well.

After preprocessing the data, the training dataset has five features: 'year', 'venue', 'coauthors', 'abstract' and 'title', and a label called 'authors'.

3 Methodology

3.1 Deep Artificial Neural Network (Deep ANN)

Deep ANNs are fully connected neural networks with an input layer, multiple hidden layers and an output layer. This model establishes deep ANN architecture with forward direction for authorship attribution and applies embedding to all features, reducing the dimension of the feature input. For the features called ‘abstract’ and ‘title’, to combine these two features with the other three features, this model needs to do the embedding aggregation on these two features. Here, it takes the mean method to capture the most information in one instance instead of minimum or maximum.

After the embedding process, the training data goes to the input layer and hidden layers and uses ReLU as the activation function. The model prefers BCEWithLogits to cross-entropy regarding loss function and uses the linear function to calculate the output layer instead of the sigmoid function, since the combination of BCEWithLogits and linear function is superior to the combination of cross entropy and sigmoid function in terms of accuracy. Moreover, the model applies dropout to prevent overfitting.

3.2 Recurrent neural networks (RNN)

RNN is good at processing sequential data, which makes it a competent candidate for authorship attribution. An RNN has internal memory that memorizes critical information about input and has a deeper understanding of the input and context, making accurate predictions possible. This report establishes RNN architecture with forward direction for authorship attribution.

3.3 Long Short-Term Memory (LSTM)

LSTM is a modified version of RNNs, which solves the problem of gradient vanishing and exploding by utilizing the input gate and forget gate. This new technique helps LSTM to decide whether this kind of information could be left. In this paper, we use it to establish one of the models.

4 Experiment & Result

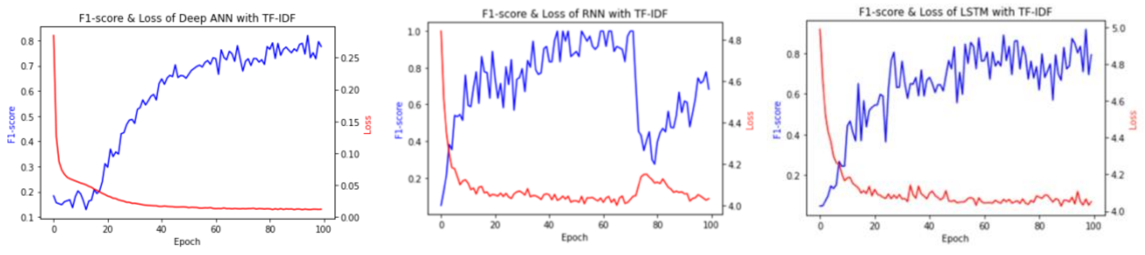

Figure 1. F1-score and loss for Deep ANN, RNN, LSTM with TF-IDF

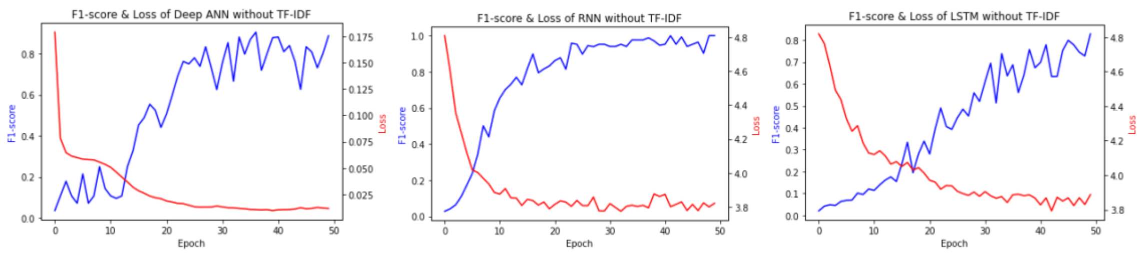

Figure 2. F1-score and loss for Deep ANN, RNN, LSTM without TF-IDF

This part will compare the performance of each methodology and discuss the reason behind it. After comparing the three graphs in the first row, it is evident that deep ANN achieves a better and more stable F1 than RNNs and LSTM do. The main reason behind it is that deep ANN, LSTM and RNN are more suitable for text classification, which means that when the input only contains text features, it would perform better. In our case, our features do not only contain text information, but they also have other features such as year, venue and coauthors. It is quite different for model solving multi-feature from text features. Moreover, RNN and LSTM have increasingly huge time complexity and space complexity as the datasets grow. As a result, they consume excessive computation power and take longer time to train and may cease operating during training. Therefore, they are inferior in this case compared to deep ANN, and deep ANN is the final approach.

After comparing the graphs in the first row with those in the second row, it is clear that after adding TF-IDF, the decreasing rate of loss and increasing rate of f1-score is faster than without adding TF-IDF. This is because TF-IDF removes all common information and extracts critical information, thus, achieving the purpose of reducing the dimension, which results in the amount of computation of each epoch dropping dramatically. In addition, selecting a good number of features is also essential. The final approach uses average length of abstract and that of the title to be the number of features for themselves.

5. Discussion & Future work

This part will discuss the problem encountered during the project, the corresponding solution, and future improvements. There are four main problems: data imbalance, the insertion of information about author order, feature selection, and embedding aggregation.

As for the data imbalance, since the training set has too many instances without prolific authors, if we keep it, it will distort the model to predict the author that is non-prolific. In order to solve it, we first try to randomly shuffle the instance without prolific authors and select 2000 of them into the training set without non-prolific authors. However, this method could not keep all the information of the training dataset, which means that our final approach should not only keep the balance of data but also try to have more information about the training set as much as possible. In the end, in the preprocessing part, this paper would copy ten times of instances with prolific authors and then add all the data without prolific authors.

Referring to the insertion of information about author order, it is hard to input the order information into the training dataset since after doing one hot encoding, that data would be the things like [0,1,0,0,1,...,0], the information of order would be ignored. Here is our solution: adding another 100 columns, which is similar to the result after one hot encoding, but the column belonging to the author in the first place would multiply by 6, and that in the second place would be multiplied by 2. For the author after second place, it would not be changed. These 100 columns would be used to train the model, but in the evaluation part, it should use the original 100 columns. The final model does not include this method due to limited time.

The third problem is feature selection. In the training set, the length of the abstract is too long and some of it reaches 800 words. To reduce dimension and retain more information about the training set, this report uses TF-IDF to select important words for each paper in the training set. This method not only considers the frequency of the word in each paper but also takes the frequency of the word in all of the paper into account, which means that it will only keep the important word for each paper and delete the high-frequency word that occurs in a lot of papers. As we have mentioned, this method reduces training time in model fitting. Aside from TF-IDF, our model adds an embedding layer for each feature which is a useful method to reduce dimension by weight. Nevertheless, it also brings a problem: after embedding, the 'abstract' feature and 'title' feature would have different dimensions from the rest of the three features. To solve it, we use mean to do embedding aggregation to capture more words' information.

In the future, it also has some improvements. For the model part, the architecture of our model already used the best parameters, it may be useful to try another model, such as Resnet, GoogleNet and VGGNet. Except for this, there are several ways to improve embedding aggregation. One of the ways is to use an attention model which could calculate the weight of different features. When it could be added into embedding, it could emphasize the importance of a feature by its weight. The other way is to use the weight calculated by TF-IDF. The value it shows could represent how important each word is. For the feature selection part, we could try to use more algorithms, such as PageRank, TextRank or LDA.

6 Conclusion

In this project, in order to solve data imbalance and keep all the information of the training set, we combined ten times copy of training data with prolific authors along with all the data without prolific authors to be new training data. Then, we use TF-IDF to select all the important words. The model we used is based on deep ANN. Aside from this, there is an embedding layer inserted before the first layer of the original deep ANN. After using the model to fit the training dataset, f1-score of the validation dataset could reach 0.88.