Authorship Attribution Research Based on Deep Learning Models

Group45

Simin Zuo (1199404) Ruiqi Pang (1312816) Haozhou Qin (1125140)

1 Problem Description

Generally, authorship attribution aims to predict the authors of a given document. In this project, around 20 thousand authors are coded as numbers from 0 to 21,245 and classified as prolific authors and coauthors. On the one hand, a hundred prolific authors are the project's target. That is to say, the model we trained is the type of multi-label classifier which need to clarify whether each of the prolific authors is an author of a given paper. On the other hand, coauthors are viewed as a feature. Besides, the feature set contains the year, venue, title, and abstract, all coded as numbers or a list of numbers somehow.
We only deploy one traditional machine learning model and one deep learning model in the research. We focus on pre-processing the data by feature engineering to circumvent overfitting and improve the F1 score.

1.1 Data Description

We use pandas to read two JSON file and study our data. Coauthors is a vital feature of this research, therefore it is necessary to explore the distribution of frequencies of each coauthor. As the Figure 1 shown, the number of coauthors with a frequency of 2 is the most and the occurring frequency of coauthors that are over 10 is relevant less. Consequently, we change the prediction results of instances with more than ten coauthors to -1.

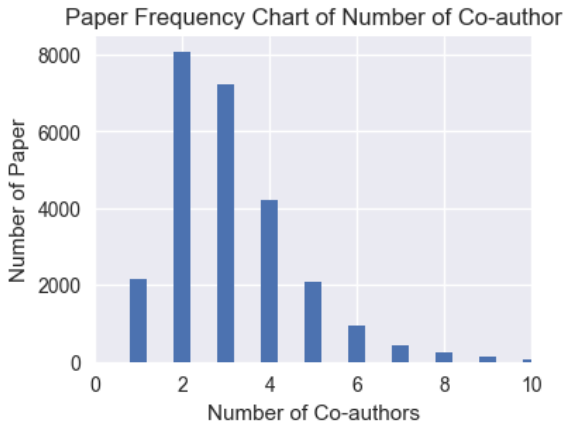

Figure 1: Number of Co-authors

2 Final Model and Results

2.1 Features Engineering

2.1.1 Authors and Venue

This section talks about the feature engineering detail of the final solution. First of all, we split prolific authors (predict labels) and co-authors out of authors (the given columns). Then, the co-authors and labels are encoded by one hot to keep a fixed-length feature vector. Although the given authors are represented by their ID numbers, these are discrete and categorical data types. In other words, numerically, categorical variables that need to be represented are converted into a form that machine learning algorithms can easily

exploit. Besides, in deep learning, there are large amounts of weighted average computation. If we use the categorical values directly, the model's error will be tremendous. Furthermore, we also utilize one hot encode venue.

2.1.2 Abstract and Title

The regular NLP pre-processes consist of segmentation. Cleaning, normalization and feature extraction. The given data set has been vectorized and mapped into an index list. Since numbers replace the words, we can't implement cleanings, such as removing stop words and normalization. In this project, we have to decide on the technique to extract features. Finally, we choose the Word2Vec model of the *gensim* package as the feature extraction method abstract and title. Generally, Word2Vec uses a series of document words to train the model, mapping the sentences or words to a fixed-length continuous vector in distributed representation form (Le and Mikolov, 2014).
Unlike Bag of Words and N-gram models, Word2Vec considers the relationships between words and the distribution density of the whole text set. When we need to rely on the meaning of words to do prediction, the extracted features that could indicate such relationships between words would be helpful. Because different prolific authors may publish articles in various professional fields, when the word dense vector of the test data set abstract appears near the dense vector of some training instances, the prediction of the test instance is more affected by these training instances. Moreover, in research on word representation, the performance of Word2Vec for larger data sets in a word similarity task has been compared with other previous such techniques and Word2Vec indeed refresh the highest accuracy with lower computation complexity (Mikolov et al., 2013). we also try to use Doc2vec during the experiments, but we find that by using the Doc2vec it will create a high relationship among the list. but in the abstract there exits some noise which does not provide any useful information. under this situation, it is better to avoid the connection with the noise word. So using word2vec can have a better performance compared with Doc2vec.

2.2 Final Model Structure

2.2.1 Deep Learning Model Structure

The deep learning model can perform better in most cases. According to our input feature size and previous experience, we establish a network structure with the number of neurons in the input layer of 25,789 and four hidden layers. At the same time, with the help of the suggestions in this paper(Taylor et al., 2021) and after some experimental tests, we finally construct that the number of neurons in the four hidden layers are 2,048, 1,024, 512, 256, respectively. Under such a quantity distribution, the network we build can better extract the useful information of the input feature and resist the interference caused by some invalid information to a certain extent.Moreover, we also apply the dropout to each layer to prevent the overfitting and increase training speed.

2.2.2 Parameter Tuning

ID	Epoch	Early stop	learning rate	batch size	train $f1$	kaggle result
1	128/300	10	0.1	32	0.966716	0.54022
2	128/300	10	0.01	32	0.966716	0.54022
3	128/300	10	0.001	32	0.966716	0.54022
4	202/300	10	0.001	64	0.966770	0.55730
5	125/300	10	0.001	128	0.969240	0.55464
6	128/300	10	0.001	256	0.966581	0.54955
7	**100**	**10**	**0.001**	**64**	**0.999508**	**0.56783**

Table 1: Results of Different Parameters

Throughout the process of our experiment, we follow the principle of controlling a single variable. We can see from above Table1 several parameters have been tuned(Taylor et al., 2021). We modulate them in different ways depending on the properties that they produce in the network. For example, in the adjustment of the epoch, we find too many epochs will lead to the generation of a fitting, corresponding to the smaller epochs would produce owe fitting, so we adapt the parameters of the early stop to help us detect when, After ten rounds of iteration, we stop the iteration in advance when the loss value did not change. In this way, we effectively prevent the occurrence of overfitting. In addition, we also find in the experiment that appropriately increasing the batch size of a specific size can better extract the information in the feature to bring a better training effect. Last but not least, the learning rate selection is also essential. According to the nature of the learning rate, we conduct

a series of experiments and finally find that Our results are the best when the learning rate is equal to 0.001, which also means that the model can find the best results and converge faster under this learning rate.

3 Other Alternatives

3.1 Naive Bayes model

When we finish feature engineering, we need a multiclass classification model to help us complete the prediction task. We first thought of the Naive Bayes model and quickly completed the training and prediction because we can use the model in the *sklearn* library. However, the effect of the model is not ideal. We think there are two main reasons. The first is that the important assumption of the Naive Bayes model is feature independent, but our author and abstract do not meet this assumption. Secondly, the Naive Bayes model cannot complete multi-label prediction. The model can only predict one prolific author for the paper. Although we use $OneVsRestClassifier$ to solve the second problem, the f1 score of the Naive Bayes model does not exceed 0.3.

3.2 Dimensionality Reduction

There are 21,245 authors in total, so when we use a one-hot encoder for authors, the length of our feature exceeds 20,000, and it is very sparse. In addition, we also worry that this highly long feature will affect the effectiveness of other features. We try to reduce the dimension of features through an autoencoder. There are also two drawbacks of an autoencoder. First, we will lose some information after using this technology. When we keep other parameters unchanged using the autoencoder, which aims to reduce data dimension, the f1 score will drop by 0.1-0.2. The second reason is that since there are too many nodes in the autoencoder, our computer is hard to run this model. Although we save only one hidden layer, the model is prolonged.

3.3 Model Based on Different Venue

In order to further improve the performance of our result, We would like to train the model separately for journals with an exceptionally high number of occurrences. For example, In the training data set, the venue0 occurs 3,782 times. It also appeared 183 times in the test set. Venue 7 is also crucial in these two data sets. So, We try to train two more models and put the result into the corresponding position. We first used the neural network model of our entire data set, which was utterly overfitting. We tried to reduce the network's number of layers and nodes to solve this problem. However, the final result did not improve. We think the main reason is the lack of data. For a neural network with 20,000 nodes in the input layer, 3,782 pieces of data are not enough to complete the training.

4 Conclusion and Future Scope

In this project, we have tried to solve authorship attribution through a deep learning network and the Naive Bayes model, and it turns out that the deep learning model possesses irreplaceable superiority. Besides, the deep learning model can handle imbalanced data and high-dimensional features better than traditional machine learning methods. Moreover, we realize the importance of feature engineering in the machine learning pipeline. Only by ensuring that the data input to the model is of high enough quality and has the appropriate number of dimensions would the model be able to predict accurately. Otherwise, it will be "garbage in, garbage out". Even though all the used features are processed, the dimension of the data we input to the deep learning model is still huge, which might lead to many computation resources spent on model training.
For future work, we might try to improve our model by using year feature. We don't use year feature in our project since we don't find a suitable way to handle it. Besides that, we think we can try to classify venue and authors by using an advanced cluster algorithm.

References

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*. https://doi.org/10.48550/arxiv.1301.3781

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. *31st International Conference on Machine Learning, ICML 2014*, *4*, 2931–2939. https://doi.org/10.48550/arxiv.1405.4053

Taylor, R., Ojha, V., Martino, I., & Nicosia, G. (2021). Sensitivity analysis for deep learning: Ranking hyper-parameter influence. *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, 512–516.