COMP90051 Statistical Machine Learning Project 1 Report: Authorship Attribution of Tweet Text

Peiyong Wang*[1], Mingzhe Du†[1], and Junrong Su‡[1]

[1]Kaggle Group: AddingWater

September 12, 2019

1 Introduction

Authorship attribution, or sometimes referred as stylometry, supported by traditional human-expert knowledge based approaches, statistical or computational methods and even modern machine learning algorithms, has a long story since the 19th century(Phani, Lahiri, & Biswas, 2017)

However, unlike traditional tasks of authorship attribution, the identification of tweet authors is far from trivial. Tweet data, unlike text for traditional tasks, often contains symbols like emoticons composed by punctuation and other Unicode symbols like emojis, together with urban slang, URLs and hashtags. Such a variety of texts increase the difficulty of our task by a lot.

In our task, we roughly have 320k tweet passages with 10k different authors. Detailed statistics of data set see Table 1. Also, the provided data is unbalanced as well. Some authors tweeted over 200 tweets while some tweeted only one. More than 2k authors tweeted less than 30 tweets in the data set.

Table 1: Statistics of the Data

Num. of Labeled Tweets	328932
Num. of Authors	9297
Num. of Tokens	157638
Num. of Chars	1099
Max Length after Preprocess[1]	118
Min Length after Preprocess	0
Avg. Length after Preprocess	15.20
Num. Tweets with Length > 50	32

2 Feature Engineering

2.1 Data Preprocess

The provided data comes originally in `*txt` format, which is not convenient for the task. The preprocess of the given data can be roughly divided into the following steps:

1. Read and acquire the file handles;
2. Extract the label and tweet text;
3. Divide the training and validation data set.

After the preprocess, the data will exist in the form of python list data structure in our program.

2.2 Feature Engineering

In raw tweets, there will be mention sign "@", hashtag symbol "#" as well as old version of re-tweet sign "RT", along with emojis an URLs. In normal practice involved with tweets, the mention, emojis, hashtags and URLs will be removed. We will follow this common practice in our approach.

2.2.1 Sparse Features

Sparse features are usually relevant to the frequency of tokens as well as the n-gram of tokens in the document. For this task, we adopted the `TfidfVectorizer` from `sklearn` to extract sparse features. However, before feature extraction, we will remove stop words and lemmatize the words to its original form to reduce the number of different tokens with python package `nltk`. After removing the stop words and lemmatization, we set the n-gram range between 1 and 2 and the max number of features to 12000.

2.2.2 Dense Features

Besides getting the dense embedding of sparse features, a popular way for converting the words and sentences to a form that computer understands is convert the words to vectors according to their position in sentences and the words appear around them (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013). Word vectors are considered to be better than sparse features like TF-IDF because they provide more information about the words itself, especially the meaning of the words. In our approach, we adopted word vectors for neural network based models. The embedding process was mostly automatically completed by the internal modules in Tencent's open sourced `NeuralClassifier`. The architecture of `NeuralClassifier` is presented in Figure 1(Tencent, 2019).

3 Learners

The traditional way for text classification is usually naive Bayes algorithm with count of words in each passage as features. However, imbalance of data may be very suitable for

*peiyongw@student.unimelb.edu.au, 955986
†mingzhed@student.unimelb.edu.au, 892896
‡junrongs@student.unimelb.edu.au, 963294
[1]Tweet length is calculated after tokenization

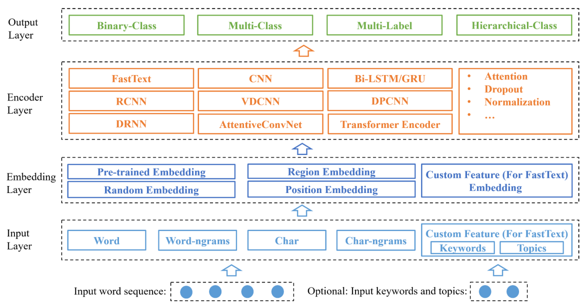

Figure 1: Architecture of Tencent's NeuralClassifier

the assumptions for naive Bayes. Complement naive Bayes was proposed to address both systemic issues as well as problems that arise because text is not actually generated according to a multinomial model (Rennie, Shih, Teevan, & Karger, n.d.). In the preliminary selection of learning algorithms, we included both learners in our baseline.

Another famous baseline for text classification would be FastText. FastText adopts a linear model with rank constraint with features from word vectors and averaged n-gram features. It can efficiently deal with large number of classes with hierarchical softmax loss (Joulin, Grave, Bojanowski, & Mikolov, 2017).

Also included in our preliminary selection of learning algorithms are TextRCNN, which is a deep neural network based algorithm combining a recurrent structure to capture contextual features as well as window-sliding convolutional operations to filter out the noise (Lai, Xu, Liu, & Zhao, 2015); and support vector machines, which have been known to be more likely to avoid overfitting due to the tuning of the regularization parameter.

4 Model Selection

The result of our preliminary results for different learns is presented in Table 2. Although TextRCNN had the best performance on training set, the validation accuracy was not a match for its outstanding accuracy on training data, possibly due to the large number of parameters in the algorithm and large number of classes in our data, which leads to a numerous number of class might be too sparse to provide enough infomation for a deep learning based model.

Also, since the word embedding were generated inside the modules of fastText and TextRCNN, it would be difficult for us to explore those features closely. By making a comparison between the five algorithms listed in Table 2, and taking time as well as computing resource at hand into consideration, we decided to choose support vector machines for our authorship attribution task.

For support vector machines, we have tried different kernels as well as different feature extraction and selection approaches. We achieved our best accuracy of 0.2955 on public leaderboard with linear kernel and 12000 TF-IDF features with n-gram range between 1 and 2. See Table 3.

[2]FastText with wrapper skift: https://github.com/ shaypal5/skift
[3]Implementation from Tencent's NeuralClassifier: https://github.com/Tencent/NeuralNLP-NeuralClassifier/blob/master/model/classification/textrcnn.py

Table 2: Models and Learners

Learner	Setting	Result
Multinomial Naive Bayes	tf-idf with 2-gram Max. 12000 features alpha = 1.0	Val Acc. 0.08

Complement Naive Bayes	tf-idf with 2-gram
	Max. 12000 features
	alpha=1.0
	norm=True
		Val Acc.
		0.12

FastText [2]	learning rate=0.2 epochs = 1000 word n-gram=3 other settings by default	Val Acc 0.20
TextRCNN[3]	batch_size= 128 num_epochs= 20 l2_lambda= 0.1 loss_type= "SoftmaxFocal CrossEntropy" other settings by default	Train Acc 0.95 Val Acc 0.18

SVM	kernel='linear'
	C=1.0
	Dual=False
		Val Acc
		0.26

5 Analysis and Future Improvement

Although our best LinearSVC achieved more than 0.29 on leaderboard, the training time for SVM is still very long, which may be due to the large number of classes. Also, for a classification task, an accuracy score of 0.29 is not quite satisfying. Normally in a classification task like this the data would be highly unlikely linear separable, but in our results a linear kernel outperformed an RBF kernel, which indicates that the approach of feature extraction and selection holds a very crucial position in statistical machine learning tasks. However, Linear SVC did not achieve satisfiting results as well, although it brought the highest accuracy amongst our models, which may suggest the real distribution of the data is far more complex than we expected.

Based on the experiments above, we are well aware of that feature engineering and model selection are the two most significant parts throughout the project. Therefore, improvements will be focused on these following aspects.

- *Feature Selection.* A good feature space makes a solid foundation for subsequent steps. In our project, we adopted tf-idf with n-gram to represent tweet features, and conceptually that works well. However, the fixed n-gram range is not able to retrieve deep-level information from a sentence, like long-distance textual relationships. For this deficiency, word embedding algorithms can project each word on the specific feature space, so that Bi-LSTM(Bidirectional Long short-term memory) can be trained on the vector space, and output a better tweet feature representing.

- *Triplet Loss.* In supervised learning, we usually use a fixed number of classes and train that network by softmax cross-entropy loss. However, we need more than

Table 3: Models and Parameter Selection for SVM

Model	Setting	Result
Baseline SVM	kernel='RBF' Unlimited feature numbers	Val Acc 0.26
Linear SVC	tf-idf with 2-gram Max. 12000 features Trauncated SVD with 1000 components after TF-IDF	Val Acc 0.16
SVM	kernel='rbf' C=1.0 count vector features	Val Acc 0.05
Linear SVC (best one)	tf-idf with 2-gram Max. 12000 features C=1.0 Dual=False	Val Acc 0.2955

9,000 classes to identify whether two tweets are from the same author in which the dimension of final softmax layer even larger than the input tweets embedding that may result in the accuracy not being guaranteed. Hence, triplet loss (Dong & Shen, 2018) is a way to learn better embedding of each tweet. In the embedded space, tweets from the same person should be close together and form well-separated clusters.

Technologically, Triplet loss includes three elements: Anchor, Negative and Positive. During training process, the distance between the Positive element and Anchor should be minimized, while the distance from Negative element to Anchor should be maximized, in which the Anchor is a sample randomly selected from the training data set, the Positive element is a sample of the same class of Anchor, and Negative is from different class of Anchor.

Since triplet loss can generally learn better features for measuring similarity than softmax classification, it might be able to improve the entire accuracy of identifying authorship from more fine-grained level.

- *Label Embedding.* In our task, the target of our model is very sparse. A well designed loss function could mitigate it to a degree, but it may not be the suitable solution to this problem. Although we are classifying the author according to their IDs provided by the original data, the real factor that can separate one author from another is the information contained in the text of his/her tweet. So the label of the authors should also be able to be embedded in a continuous space rather by a group of one hot vectors. In such a case the distance between some authors can be more closer than others and these authors can be identified as a group or cluster in the author space, hence enables us to classify the authors in a hierarchical manner, which sometimes could provide more ways to explain how the models works.

References

Dong, X., & Shen, J. (2018, September). Triplet loss in siamese network for object tracking. In *The European Conference on Computer Vision (ECCV).*

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017, April). Bag of tricks for efficient text classification. In *Proceedings of the 15th conference of the european chapter of the association for computational linguistics: Volume 2, short papers* (pp. 427–431). Association for Computational Linguistics.

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015). Recurrent convolutional neural networks for text classification. In *Proceedings of the twenty-ninth aaai conference on artificial intelligence* (pp. 2267–2273). AAAI Press.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th international conference on neural information processing systems - volume 2* (pp. 3111–3119). USA: Curran Associates Inc.

Phani, S., Lahiri, S., & Biswas, A. (2017, August). A supervised learning approach for authorship attribution of bengali literary texts. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, *16*(4), 28:1–28:15. doi: 10.1145/3099473

Rennie, J. D. M., Shih, L., Teevan, J., & Karger, D. R. (n.d.). Tackling the poor assumptions of naive bayes text classifiers. In *proceedings of the twentieth international conference on international conference on machine learning.*

Tencent. (2019). *NeuralClassifier: An Open-source Neural Hierarchical Multi-label Text Classification Toolkit.* https://github.com/Tencent/NeuralNLP-NeuralClassifiere. GitHub.