Authorship Attribution Classification

Xiuge Chen, Shizhe Cai, An Luo

1 Introduction

As social media is getting popular, authorship attribution of short text, which is to identify authors of short anonymous texts, is getting more and more important. Authorship Attribution was achieved by using linguistic analysis and is now facilitated by natural language processing and machine learning, and it mainly works by learning textual patterns such as frequent words, sentiments, and sentence structure in the origin post text. The application of such a concept may help to identify the authorship of some unknown pieces of literature and detect plagiarism of paperwork.

2 Datasets and Features

2.1 Dataset

The dataset used is a collection of tweet texts. It has approximate 328k tweets labelled with over 10k authors.

2.2 Preprocessing

Different preprocessing technologies were applied here, and their outcomes were varied significantly.

There are three approaches available when handling retweets: remove all retweet content, remain the authors' comments, and keep all retweets intact, replace the retweet symbol "RT" with "@RT" only. The last one is the most effective approach because that author comments may reveal the writing style of an author, and retweeted content may show authors' interest or hobby.

Tweets also contain some special terms such as mentions, hashtags, links and URLs, and other special characters. We keep all mentions and hashtags to capture the relevant textual features, then replace links and URLs with "@url" to remove noise words and characters. All special characters are kept because they are emotional representative or part of authors' writing styles, thus can be utilized to attribute authors.

Other techniques such as converting all characters to lowercase, removing stop words, word lemmatization and merging all tweets from the same user into one document were tested but not applied as they are considered to result in information loss.

After finishing the preprocessing, texts are tokenized into separate words which will be further engineered.

2.3 Feature Engineering

The data obtained from the previous preprocessing step needs to be further engineered to obtain effective features. There are various methods to process the data.

Firstly, constructing word n-grams could reveal the similarity of texts regarding word punctuations. It has been shown from Sari [1] and Schwartz [2] that word n-gram has a good performance on the short text.

Constructing Part of Speech (POS) tags and its n-gram is another way (Jafariakinabad [3]) of adding important features. Since syntactical contents are associated with the personal lingual habit, it can be used to identify authorship. In the experiment, we utilized the out-sourced package called "nltk" to process POS tags.

Sentiment intensity analysis is also performed to indicate the intensity of sentence(positive, negative, neutral). We use a lexicon-based sentiment analysis tool called VADER[4] to score the sentiment of each text.

Some additional feature engineering techniques like character N-gram (Schwartz [2], Shrestha [5]), lexicon features and PCA dimension reduction also tried out in the experiments. However, they require a complex computation, and the result is mediocre. Therefore, the first three methods are mainly used in the experiment.

The final step of the feature extraction is to select useful features by applying TF-IDF transformation to represent word frequency features in the sparse matrix.

3. Learners and Model Selection

In this task, we assumed that the high-dimensional features space we constructed is linear separable when using one-versus-all strategy to handle multiclasses. We adopted one-versus-all because it is faster to train and test comparing to one-versus-one. Linear separability is a commonly used characteristic for authorship attribution (Schwartz [2], Silva[6], Bhargava[7]). The features above, combining with tf-idf frequency estimation, makes recognizable users differentiated from others in some dimensions. Therefore, we decided to choose learners that were suitable for finding linear hyperplane between classes.

In the linear learner selection, we firstly considered Logistic Regression, Linear SVM and SGD Classifier. Logistic Regression was slower to coverage and doesn't optimize the margin between classes. Thus, we performed our experiment and analysis mainly on LinearSVM and SGDClassifier. After a series of experiments on the parameters, we found that the best penalty parameter C of LinearSVM varies when using a different subset of data, as the variance of the dataset is unavoidably high. Therefore we chose 0.68, which is more robust with aslo good performance. Additionally, LinearSVM performs better with squared hinge loss, l2

regularization, while SGDClassifier performs better when used hinge loss, l2 regularization.

Penalty parameter C of LinearSVM controls the tradeoff between smooth decision boundary and classifying the training points correctly; that is the balance between training error and generalization. Usually, large C will force the hyperplane to classify more training point correctly thus might overfit the data. Since different posts from the same users might not be quite similar (high variance of post contents), a smaller C(0.68) makes SVM consider the ignore of unusual and not representative posts. Thus it could generalize better on test task. Also we noticed that the greater of C, the more uncertainty about the performance of SVM (varies a lot when using different dataset), it is because some robot-like users are very consistent, while real users are quite versatile, where larger C is only suitable for the formal one.

L2 regularization outperforms than L1 regularization in both learners for the following reasons. Firstly, L2 penalizes the estimates so that it leads to a balanced minimization of weights; less influential features will suffer more penalization. Also, L2 is more sensitive to outliers since it squares the losses, in our task, outliers might also contain indicative features, which L1 might not be able to catch due to its “robustness”. Moreover, the built-in feature selection and sparse output (solution on axis) of L1 might also filter out some useful features. Lastly, L2 is computationally faster and more stable (unique solution to avoid the ill-posed problem) than L1, so we chose L2 as regularization methods.

It is interesting that hinge loss and squared hinge loss performs differently on LinearSVM and SGDClassifier. When testing with 200 users, using hinge loss and squared hinge loss, LinearSVM achieved 44.99% and 46.35%, and SGDClassifier achieved 26.61% and 45.94% respectively. Squared hinge loss minimizes at $(yw'X)2 = 0$ while hinge loss minimizes at $yw'X = 1$, this might be the main causes of parameter differences between them. Also, LinearSVM uses batch gradient descent to update parameters, but SGDClassifier uses stochastic gradient descent, the randomness (shuffle data and update parameters only using a subset of instances) that introduced in SGD makes each gradient step more noisy (fluctuate), which might cause the difference between LinearSVM and SGDClassifier. Finally, we chose LinearSVM with squared hinge loss and SGDClassifier with hinge loss.

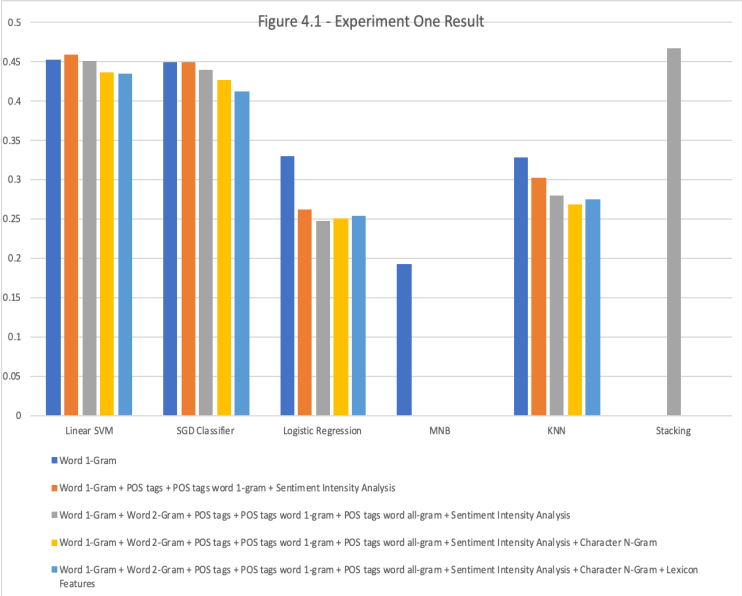
In order to improve the accuracy, we also built a simple stacking structure from LinearSVM and SGDClassifier. The stacking base learner contained three LinearSVM dealing with different features, and a meta LinearSVM learner learning their results. Due to time limitation, we adopted the same parameter settings.

We also tried a few other models that do not require linear separable propriety, such as Multinomial Naive Bayes and K-Nearest Neighbors Classifier, in order to validate our assumptions and approaches.

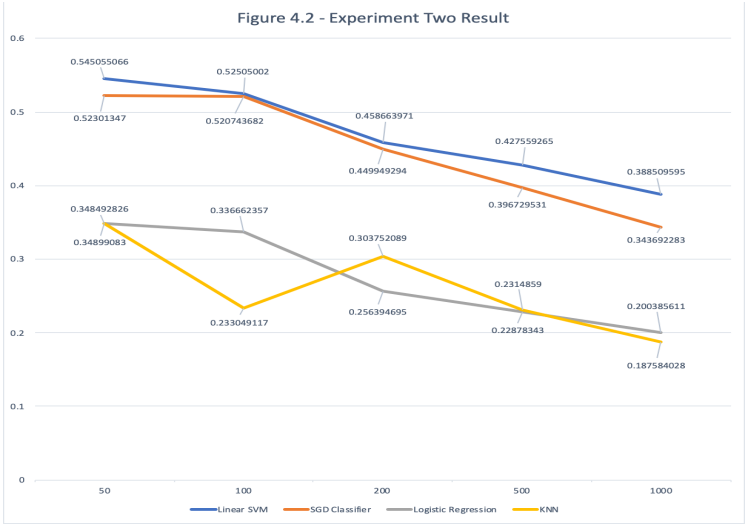
4. Experimental Result and Analysis

We have conducted two experiments to compare the combination of learners and features, and choose accuracy after 10-fold cross-validation to evaluate the effectiveness, as it is simple to interpret and could reflect how well is our classification.

The first experiment is to test different combinations of models and features, with training data contains 200 users’ tweets. The results are listed in figure 4.1 below.



Another experiment was conducted with the feature combination b and the different number of users, as shown in figure 4.2.



According to the experiments, It is clear that, for all learners, the accuracy degrades as the number of users increases. It is predictable because as the number of users increases, more and more similar users and noises data coming in might impact on the classification task negatively.

Option 'a' and 'b' outperform across six preselected combinations in terms of accuracy. As mentioned above, word features reveal authors' word using hobby and potentially their personal information such as interests, POS tags are a strong indicator of writing styles, and sentiment could help correlate related sentences since some users might complain or excitingly interact with others on twitter. However, it is interesting here that adding more features such as word 2gram and lexicon features doesn't help classify more accurately. Lexicon features could vary a lot for different tweets, and n-gram features might be too unique to each tweet, so adding them will introduce more noises and outliers that affect the decision boundary, the model won't be able to generalize well.

Also, given the same training instances and feature combinations, linear models like LinearSVM and SGDClassifier outperform from others, and Logistic regression doesn't work quite well. The reason why SVM-like learners are more suitable in this task is that, tf-idf features constraint the samples in a fairly dense space, and they are not purely linearly separable, SVM-like learners with soft margin don't penalize correctly identified instances and try to maximize the margin between classes, so it could identify classes as long as some indicative features appear. Logistic Regression only minimizes the error but does not try to find the best hyperplane(multiple possible solutions), so its solution might be too close to some points (support vectors in SVM), so that it could not generalize well on high dimensional noisy instances. As other learners such as MNB, the contribution of informative features might be influenced by noise data since they take all features into account.

5. Conclusion

In conclusion, unlike other text classification tasks, authorship attribution does not require lots of preprocessing such as remove retweet and special characters, because these terms might also reveal the important characteristic of authors such as interests. Out of multiple features with tf-idf extraction, word 1gram, POS tags, together with sentiment analysis, achieve better classification accuracy. Adding features that are too unique to each post such as word 2gram makes learner overfit on train data, adding features that are too variant for same users such as lexicon features, will introduce noise to the model.

After feature extraction, the samples become nearly linearly separable. Thus linear models that try to find max-margin between classes with L2 regularization term, are validated suitable to be used in authorship classification, such as LinearSVM and SGDClassifier. Also, due to the high variance of user posts, learners should be robust to noise term to generalize well, which could be achieved with the help of soft margin and smaller penalty term. Comparing to other models like Logistic Regression and MNB, they achieve about 20% higher on accuracy. LinearSVM and SGDClassifier have similar but not the same performances since they use different optimization methods and loss functions. The use of ensemble learning on LinearSVMs could further improve the accuracy by 2%.

6. References

[1] Sari, Y., Vlachos, A., & Stevenson, M. (2017, April). Continuous n-gram representations for authorship attribution. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers* (pp. 267-273).

[2] Schwartz, R., Tsur, O., Rappoport, A., & Koppel, M. (2013, October). Authorship attribution of micro-messages. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (pp. 1880-1891).

[3] Jafariakinabad, F., Tarnpradab, S., & Hua, K. A. (2019). Syntactic Recurrent Neural Network for Authorship Attribution. arXiv preprint arXiv:1902.09723.

[4] Hutto, C. J., & Gilbert, E. (2014, May). Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*.

[5] Shrestha, P., Sierra, S., Gonzalez, F., Montes, M., Rosso, P., & Solorio, T. (2017, April). Convolutional neural networks for authorship attribution of short texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers* (pp. 669-674).

[6] Silva, R. S., Laboreiro, G., Sarmento, L., Grant, T., Oliveira, E., & Maia, B. (2011, June). 'twazn me!!!;('automatic authorship analysis of micro-blogging messages. In International Conference on Application of Natural Language to Information Systems (pp. 161-168). Springer, Berlin, Heidelberg.

[7] Bhargava, M., Mehndiratta, P., & Asawa, K. (2013, December). Stylometric analysis for authorship attribution on twitter. In International Conference on Big Data Analytics (pp. 37-47). Springer, Cham.