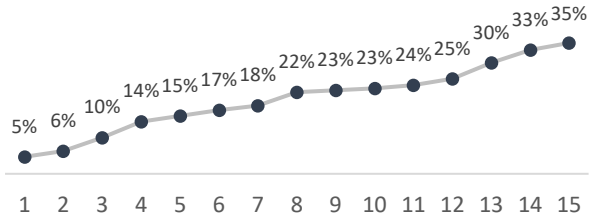
1. Introduction

This report aims to describe the approach we have undertaken to address the authorship attribution on Twitter. We were given two datasets, a training set that contains almost 10,000 authors and approximately 330,000 tweets and another unlabelled test set. The main goal is to evaluate machine learning approaches that can effectively identify the authors of tweets in the given test set.

The test prediction has been submitted in "Who tweeted that?" Kaggle competition and were able to achieve a considerably high accuracy score for such extreme multiclass problem. Our team made 39 submissions on Kaggle and achieved first rank (by the time of report submission) in the public leaderboard with prediction accuracy of more than 35%. The following chart shows the progress of the accuracy score.

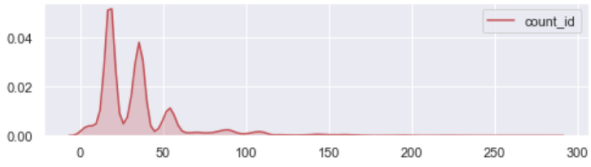


2. Methodologies

The final submission is using one-vs-all SVM with linear kernel model with L2 norm regularization with features comprised of a combination of modified TF-IDF and character level bigram and trigrams. The detailed explanation of our methodology is structured in this report as follows. The first section describes the initial experiments that were performed to gain a general understanding of the overall problem. This helped in choosing the machine learning model, learners, and feature engineering. Next section describes the feature engineering undertaken for this task. After that, we describe two the two SVM based machine learning models along with their hyper-parameters tuning, that was our final model candidates used for this task. We also present our approach to explore using Convolutional Neural Network to solve this problem, which could not yield the expected accuracy. In the end, future works and the conclusion is presented.

3. Initial Experiments

We started to gain an understanding of features selection and the choice of Machine Learning models by performing several experiments on a small sample of given training data. We evaluate the distribution of tweets per author in training set and found that it is right skewed. Some authors have more tweets than others and other 29 authors have only one tweet which does not support enough variation for prediction. We thought of reducing the datasets based on tweet authors.



We derive features that represent the author's writing style such as adding flags to indicate retweets, URLs and user mentions. We also considered length of tweets, number of words, number of unique words, frequencies of punctuations, number of capital letters, number of hashtags, number of stop-words, number of unique part-of-speech (POS) and frequencies of each 27 types of POS tagging. Thus, in total there are 43 features representing the stylometry of the author without including the bag-of-words (BOW).

To choose a smaller sample, we first removed the authors having less than ten tweets. Then we took only those authors having F-score greater than 0.2 from k-NN classifier and trained it on 80% dataset. The resulting sample gives us 1137 authors and 67,078 instances with less skewed distribution.

We trained some models on 80% of the smaller sample and validate it using 20% of held-out from that sample and 20% of the held-out from the actual training set. With k-NN, we achieved an accuracy of 24% and 10% respectively. This represented overfitting, thus, we tried to use regularization method with ridge regression which returned lower accuracy of 14% and 1.2% respectively. Lastly, we used one-vs-one SVM and achieved an accuracy of 28% and 5.7% respectively. When we apply the k-NN and SVM models on the actual test set, SVM returned higher accuracy by 1%. Hence, SVM turns out to be more robust than the others.

Further insights we can gain from this initial trial about feature extraction. The features might only have better representation on the 10% classes chosen in the smaller set. Therefore, having more features to represent other authors may increase the accuracy. However, overfitting could be the main challenge in this project, thus, we have to embed regularization parameter in our model.

4. Feature Engineering

4.1. Bag of Words (BOW) and TF-IDF based vectors

With the stylometric features specified in the initial trial we achieved an accuracy of 11% using 80% of the training set on SVM. We then involve vector representation of word, started from BOW with 10000 features. This improved the accuracy to 13%. The next enhancement is to scale the features based on their frequency in the entire set. The hypothesis was if an author uses specific uncommon vocabulary, it can be used to identify that author. Terms frequency can also be used to identify an author. Hence, instead of BOW approach we applied TF-IDF vector representation of tweets and with 5000 features. This resulted in a prediction accuracy of 15%.

We learned that even though we use 50% fewer features in TFIDF compared to BOW, we got higher accuracy. This is due to the scaling transformation in TF-IDF capturing the distribution of each feature compared to the others. It is also important to highlight that TF-IDF has captured the information of the stopwords automatically, thus we do not need to count for number of stopwords. We also prove our hypothesis that enhancing the number of features in TF-IDF transformation improves the accuracy score. However the improvement in accuracy slows as the features are increased. The accuracy improved from 34.7% to 35.3% when features are increased from 100000 to 230000 (almost 130%).

4.2. Capturing Stylometric Features

We tried to identify more stylometric features that can be used to identify the author. An important point is that we used TF-IDF representation of tweet and the generated features were embedded in the normalized tweet so that the feature is accounted in TF-IDF transformation. Starting from basic normalization of just lemmatisation and lowercasing, we enhanced the accuracy to around 22% by applying transformations to the tweets as specified in the table given below. For every new added stylometric feature, the accuracy score goes higher by 0.5% to 2% on the same classifier.

	Modified Normalization
1	convert tweet text to lemma and lowercase
2	detect retweet, mention, hashtag, http
3	detect punctuation and special characters
4	detect emoticons such as :) and :(
5	detect capital I as in I, i, I'm, i'm and w/ for with
6	detect digit and uppercase word
7	detect POS for every word
8	detect sentiment polarity of every word using lexicon based approach and classify it as negative, positive, or neutral
9	detect more word count

As we apply the modified normalization to a tweet, it will extend the keyword detection as shown in the following example. Thus, during TF-IDF transformation every keyword extends for more features.

Original:	RT @handle: More Thanksgiving leftover ideas http://bit.ly/8hpVLJ
Modified:	_retweet_ _handle_ _colon_ more _RBR_ _positive_ _word_ thanksgiving _VBG_ _netral_ _word_ leftover _NN_ _netral_ _word_ idea _NNS_ _netral_ _word_ http

Ultimately, we achieved an accuracy of 25.6% by using all features generated by TF-IDF using the modified normalization.

4.3. Inclusion of Character level bigrams and trigrams

The normalization specified in sections above helps in improving the accuracy, but it could not capture the author's writing style at character-combinations level like using don't (vs do not), "we've" (vs we have) and "Mag's" etc. Therefore, in our last trial we extracted character level bigram and trigram, generated their TF-IDF vector and combined them with TF-IDF vectors of normalized tweet. We achieved an improvement of 4% in prediction accuracy to reach almost 29.5%.

4.4. A trade-off between Number of Features and Overfitting and their implication on Accuracy

As mentioned before, high number of features can cause overfitting model. Even though we have put the L2 penalty to support the regularisation, it is not enough to handle the overfitting issue. One of the reasons could be multicollinearity, as we suspect to have derived many correlated features. In the nature of text features itself, there is a pattern where particular word/character often appear consecutively with other word/character such as "I" and "am". However, as we try to reduce the dimension using the Singular Value Decomposition the accuracy dropped by 2%. Thus, there is a trade-off to get more generalizable model with higher accuracy.

We also found a contradictory manner, when we increase the number of TFIDF features by 130% we got the gap between accuracy in training and test set reduced by 0.3%. Even though the number is insignificant, it shows that having more features does not necessarily worsen the overfitting.

5. Model Selection

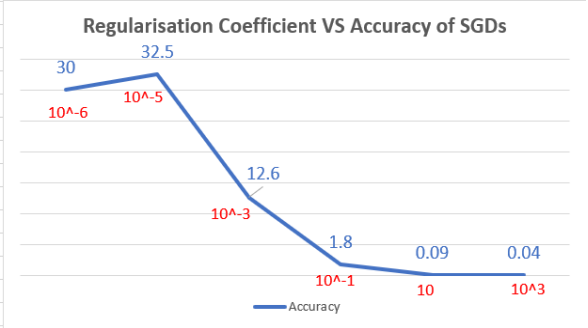
Based on the trials, we decided to use SVMs as training models. We explored two different SVM approaches in detail and ended up with Linear SVM for achieving final accuracy. Moreover, Convolutional Neural Network were also explored and presented in a subsection. We maintain to keep the entire training set divided into 80% training and 20% validation set.

5.1. Linear SVMs with Stochastic Gradient Descent (SGD) based Training (SGD Classifier)

The first configuration of SVM is a one-vs-all SVM with gradient descent on hinge loss with 5 epochs and L2 norm penalty. This is the model which was used during feature engineering. With the final feature selection as specified in Feature Engineering section, we achieved an accuracy of 29.5%. The decreasing misclassification rate along with cross-validation score is shown the following learning curve.

5.1.1. Hyper Parameter Tuning of SGD Classifier

The SGD classifier is tried with various values of regularisation coefficient and its best value in terms of accuracy (32.5 % accuracy on 20% validation set) has been found to be 10^(-5). The plot of various values of accuracy vs the coefficient is as follows:

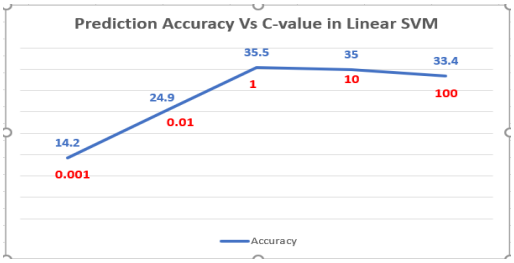


5.2. Linear Support Vector Machines

It is important to consider that SGD Classifier are very sensitive to feature scaling. It is usually recommended to scale the data to have standard normal distribution. Since all the features are weighted TF-IDF values, it is difficult to precisely satisfy such conditions on these many attributes and we hypothesised to use a different Linear SVM that doesn't depend on any particular distribution assumptions. Furthermore, the linear kernel in linear SVM support for faster convergence in a convex model. In this particular datasets using one-vs-all SVM with linear kernel, squared hinge loss as cost function, 5 epochs, and L2 norm penalty, we achieved the highest accuracy of all of our trials.

5.2.1. Hyper-parameters tuning of Linear SVMs

The coefficient C is another regularisation parameter as it controls the trade-off between low training error and low testing error. We tried with various C and choose the one with best prediction accuracy on unseen data which is C=1.0 for an accuracy of almost 35.5%.

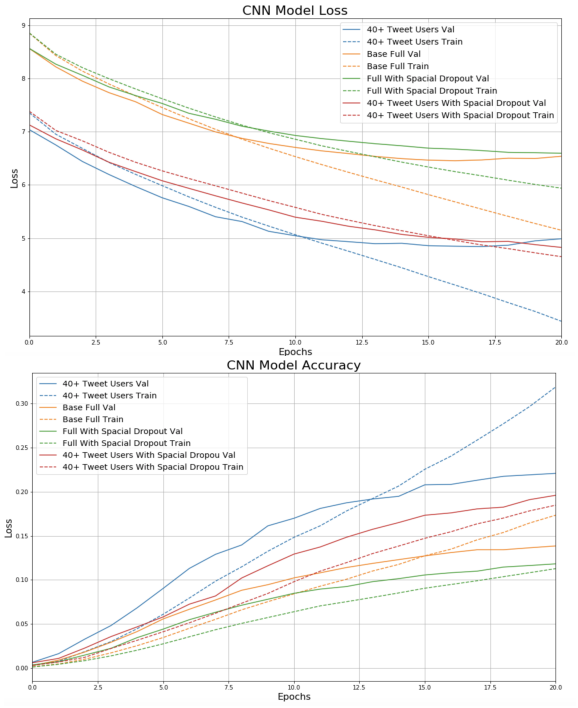


5.3. Deep Learning with CNN

Even though the best accuracy score we got is based on linear SVM, we attempted to build a 1-D Convolutional Neural Network with Max Pooling. Although CNN was generally used in computer vision, they have recently been showing promising results and became wildly popular in text classification. As CNN exploits local correlation of input data using small filters, it has the ability to detect patterns in text which makes it suitable for Authorship attribution.

We have applied various sizes of the kernels (3, 4, 5) on each of the 3 convolution layers respectively, this allows the capability to detect patterns of multiples sizes of adjacent characters (relatively similar to n-gram used in SVM)[1]. The base network design was scaled down from the original 5 to 3 convolution layers due to the smaller input dimension and larger number of expected classification output. The model was trained using SGD optimizer with learning rate of 0.01, for 20 epochs and batch size of 40. After comparing the accuracy and loss between the training and validation set it was evident that the model was overfitting even before the completion of 20 epochs (refer to following diagrams). Therefore, we added a spatial dropout layer with dropout rate of 0.5 after the first convolution. This resulted in more consistent accuracy of around 11% in training and validation dataset and the Kaggle Test sets.

The performance results on this dataset are favourable towards SVM model in contrary to our initial expectations. The extreme imbalanced classes may take part on this underperformance issue. This hypothesis was confirmed when we retrained the model with only users having more than 40 tweets (1888 authors). This model showed the best graph (in red below) in terms of loss and accuracy improvement over time. It is clear that the model has not yet converged after 20 epochs and the accuracy was still increasing, thus we expect the performance to improve with longer epochs. Considering only 4% of the users in the training have more than 40 tweets associated with them our CNN implementation doesn't seem to be the best model for this particular training dataset.



6. Conclusion

In this project we learned how to design a better machine learning algorithm using creative feature extraction based on modified normalization and character level n-gram, model selection and hyperparameter tuning based on the accuracy score and learning rate, and minimising overfitting impact by using regularization penalty term. Even though we have not completely handled the overfitting issue, our model has similar result in held-out dataset and test set thus indicates it is a robust model. Compared to our first trial, we attempted to increase the growth in accuracy score by 600% and we experienced the real efforts in training better model along the way.

7. Future Recommendation

The main issue that we have not completely solved is handling the overfitting. In future research, it is better to do dimensionality reduction using autoencoder. The compressed features from autoencoder can be used in CNN with more convolution layers and number of epochs. Having more features such as n-gram based on POS tag, non-English word detection, and spelling correction may also contribute to increase the accuracy.

[1] The model design was based on the work by Zhang, X., Zhao, J., & LeCun, Y. (2016). Character-level Convolutional Networks for Text Classification.