Project 1: Who Tweeted That?

Team 69
695281 Steven Maharaj 694419 Joshua Yang 733301 Chenqin Zhang

1 Introduction

In this paper, we present an analysis of the authorship attribution model the team implemented, and discuss about the various approaches that were used or considered for the task. The particular aim of this authorship attribution model is to take in tweets in text forms and predict the corresponding authors individually, based on the tweets corpus that is provided. In this project, there are around 10k authors and it is virtually impossible for human reader to identify authorship. This paper aims to fill in some of the research gaps and provide insights on identifying authorship with a large number of classes.

2 Feature Engineering

2.1 Dataset

The data used in model training is a single text file **`train_tweets.zip`** that contains 328k tweets authored by almost 10k Twitter users. Tweets could contain a lot of irregular patterns, and some of the information could be difficult to capture. However, this diversity of language usage and twitter features also indicates the tweeting styles of particular authors and could provide additional knowledge for classification.

There are four main tweet characteristics that could be used to identify particular authors:

1. Topic: What content the author tends to tweet about? (Finance, politics or personal life etc.)
2. Word usage: What type of words does the author use? (simple language, academic English, or the use of slang, etc.)
3. Punctuation (Ex. Does the author have a higher tendency of using exclamation marks?)
4. Emojis: The tendency of using emojis in tweets.

In our earlier attempts, we generated extra columns and calculate the relevant figures of the key linguistic variables in each tweet: Number of special characters, Number of words, Number of characters, Average word length, Ratio of stop-words, Use of a handle, Use of a link, Is a retweet, Is a modified retweet.

However this approach generated really poor result (<1% accuracy) with different classifiers. It seems to have failed to capture the multifaceted nuances in tweets and language in general. The insufficient features could also hinder accurate prediction as there are a large number of classes in this particular task.

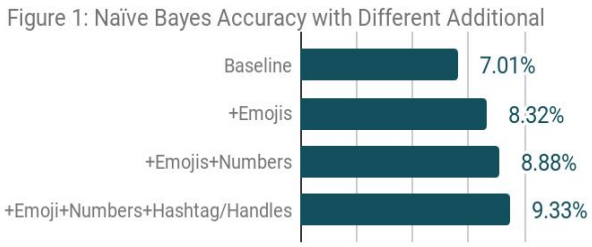
In order to capture all these information in an effective manner, we decided that a bag-of-words (BOW) method would be most suitable. A bag-of-words approach creates a vocabulary of unique words occurring in all the tweets in the training set, and could better capture the information mentioned above with some adequate feature engineering.

2.1 Data Preprocessing

To ensure that more tweeting traits and writing styles could be correctly picked up by the bag-of-words methods, we have to consider how we preprocess the data:

```
8746    RT @handle: Director of Global Brand Marketing, Hotels and Casino's $125k +
        30% bonus - Orlando Fl :D http://bit.ly/4kUmBB #jobs #twitjobs
```

An example from the train set above shows that varieties of linguistic and twitter features that can appear in a common tweet. The bag-of-words method could gather the information on the topic and the word usage of the tweet by collecting these words in lower letter: ‘director’, ‘global’, ‘brand’ & ‘marketing’ etc. In order to for emojis, hashtag signs, special characters, numbers and punctuation to be considered as individual ‘words’ in our bag-of-words methods, extra spaces are padded



around these features in preprocessing. Link URLs and @handle are also replaced in each tweet as #http and #handle respectively to capture the use of links and handles. The improvement of accuracy is shown in Figure1 with the additional features specified, using Naive Bayes as the example.

However, there are some limitations that remain difficult to overcome. For example, some of the tweets are retweets posted by another author and therefore the writing style of the tweet would be different from that of the user. In this case the topic and content would be the main useful information to identify the particular author.

2.2 Number of Features

One of the design decisions involves the number of features used for model training. In Naive Bayes and Linear SVM training, we extract the top n features that appears in the training set and used Gridsearch to find the optimal number of features to reach max accuracy.

Figure 2: Naive Bayes Accuracy vs Number of Features

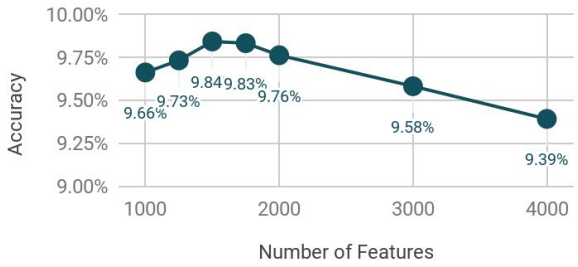
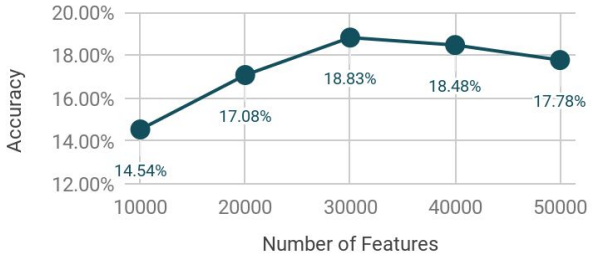


Figure 3: Linear SVM Accuracy vs Number of Features



In two of the models attempted, Naive Bayes and Linear SVM, we observe peak accuracies occurring with different numbers of features used (Figure 2 & 3). In Naive Bayes, accuracy peaks at around 1500 features and decreases with more features added, whereas that figure of Linear SVM lies around 30000. This difference is most likely due to the conditional independence assumption that Naive Bayes holds. A larger number of features would have a higher degree of interactions between features that would not only be difficult for Naive Bayes to capture but also generate more noise for the model. SVM in this case captures more interactions between features and benefits from having a larger number of features.

3 Learners

In this project, various learners were attempted to classify authors. We first experimented with some deep learning models and faced significant challenges in terms of increasing the prediction accuracy, then we tried out classical Machine Learning approaches at the end to obtain better results.

Table 1: Best Prediction Accuracy with Different Classifier

	Neural Net	BERT	FastText	Naive Bayes	Log Regression	Linear SVM
Val Accuracy	0.84%	~0.11%	4.53%	9.84%	11.7%	18.8%

3.1 Deep Learning Approaches: Neural Net, BERT [1] & FastText [2]

We experimented with a Feed Forward neural net with one hidden layer consisting of 1000 nodes (with l2 regulariser), a batch normalisation layer and a dropout layer. Train accuracy was reported to be 3%. Once evaluated on the test set accuracy dropped to below 1%. Although a regulariser, batch normalisation and dropout were included there was still evidence of overfitting. Neural word embeddings are also some common practices for encapsulating distributional semantics. Both BERT and FastText are some state-of-art open sourced neural word-embeddings methods. With limited computational resources, the model only observed ~0.1% accuracy with BERT and around 4.5% using FastText.

It is likely that the model was so complex (many learnable parameters) it fitted random noise in the data. In addition, the number of tweets per author was quite low (averaging ~38 tweets) so if the pool of authors decreased and there were more tweets per author it is likely we would see an overall improvement in accuracy.

3.2 Classical Machine Learning Approaches: Naive Bayes, Logistic Regression & Linear SVM

From Table 1, the classical machine learning approaches have performed significantly better in terms of identifying authors of tweets. This outcome is likely due to the high dimensionality and sparseness of feature vector results and the non-uniformly distributed volume of tweets among authors. Compared to classical machine learning methods, deep learning approaches require large number of data and long training time to achieve decent performances. It is likely that with this dataset, deep learning methods have not avoided overfitting and have converged to biased results.

It was found that the Linear SVM (with stochastic gradient descent) performed the best out of all the learners attempted. The reported accuracy was approximately 19%. As mentioned in section 2.2, the SVM seems to be capable of capturing more information using a larger number of features. In this case, the SVM is the most suitable model to maximise the learning with training set of this scale.

4 Model Selection

Overfitting is often a problem machine learning algorithms suffer. In order to tackle the issue of overfitting in our model, we explored the use of regularisation. Compared to other models attempted (such as the neural net), linear SVM is relatively simple in terms of the learnable parameters in the model. And the selection of the model per-se combats some extent of over-fitting since the model is less likely to fit random noise with less parameters.

On top of model selection, regularisation is a technique that discourages the learning of more complex model, so as to avoid over-fitting. A penalty term which dictates the penalisation of model complexity is added to the loss function. Therefore, the use of regularisation reduces the variance of the model, without substantially increasing model bias. For the linear SVM we implemented, a penalty term involving l2 (ridge) regularisation was added to the loss function. As the penalty term grows, the number of wrongly classified point decreases. However, when lambda reaches infinity, the model approaches a hard-margin SVM which allows no miss-classification, otherwise there would be infinite loss. It should also be noted that we were dealing with a very large data set so the linear SVM was very slow to train. Hence, the problem was reformulated to use the SGDClassifier class, with hinge loss and $\lambda = 1/(mC)$. Where m=2 if the l2 norm is used for the loss function.

5 Conclusion and Future Improvement

1. Non-linear SVM instead of linear SVM: Linear SVM is a parametric model that is much less tunable than a SVM with RBF kernel. Using SVM with a non-linear kernel (Gaussian, rbf, poly etc.) could potentially capture more interactions between features.

2. Extract certain number word features per user: In the bag-of-words implementation in this project, we extract top n feature in the whole training set. However, that can potentially fail to capture the features that occur less frequently overall. For example, non-English words are unlikely to make it to the top n features in our implementation but could be very informative to identify certain users. In future implementation, we can potentially extract the most frequent words per user to ensure that the bag contains sufficient information on each user.

3. Use text parsing to get syntactic structures as features: Instead of focusing on words and symbols, syntactic structures could provide a means to trace tweets similarity by the same authors. For example, some users may prefer to use superfluous attributive clause, but other users prefer to be minimalist.

4. Extensive GridSearch: For the Linear SVM this includes searching over the kernels, regularisation coefficient, type of regulariser, learning rate and possibly other hyperparameters. Therefore, resulting in the most optimal model.

6 References

[1] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Retrieved from https://arxiv.org/pdf/1810.04805.pdf

[2]Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of Tricks for Efficient Text Classification.