**Introduction**

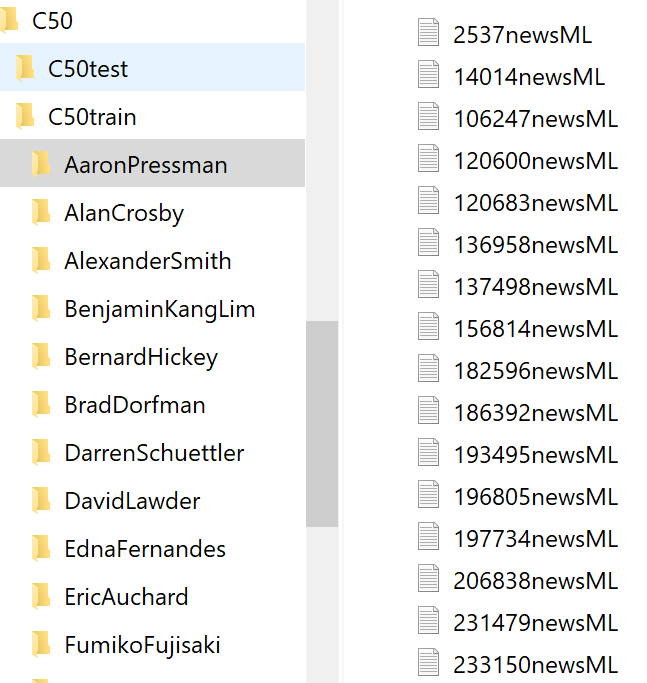
Authorship attribution is an important task in text categorization, which uses author as the category label [1]. It is thought as a multi-class, single-label text categorization task. The main problem of authorship identification is to extract characteristics which can represent author’s style from a collection of text. Since in an era when lots of electronic text can be rapidly copied and disseminated, authorship identification can also play an important role in information security such as copyright protection and identification of malicious messages.

In this study, we focused on finding a more suitable vectorizer and classifier in authorship attribution. Different vectorizers and classifiers were compared and representative errors were analyzed.

**Corpus**

The corpus is a collection of newswire stories of 50 authors from Reuters Corpus Volume 1 for authorship attribution. Texts in this corpus are all short texts, the average number of words of each article is 445. Since in the real-world author attribution tasks only short texts of each author are available, findings from this corpus will have realistic significance. Besides, all the texts are labeled with at least one subtopic of the class CCAT (corporate/industrial) [2]. Thus, the topic control strategy has minimized the influence of topics and the classification would be more about authorship differences.

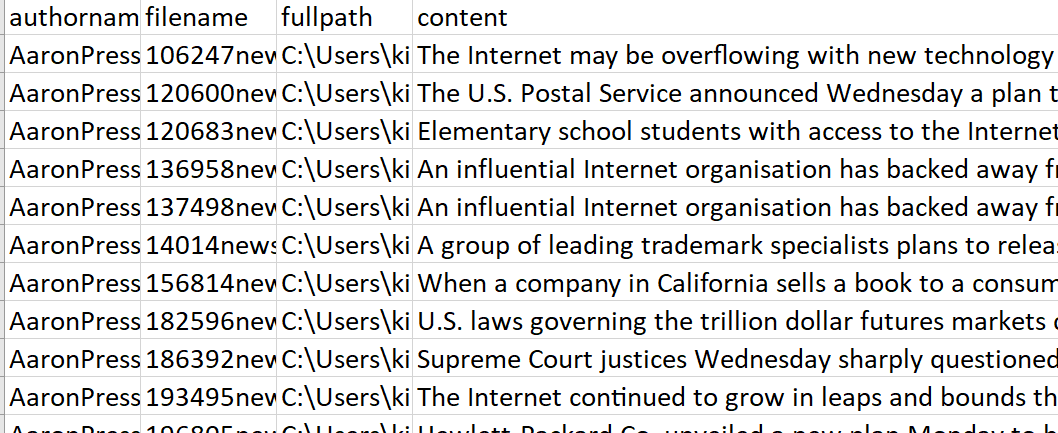
The corpus has been divided into two folders – training and testing. The training corpus consists of 2,500 texts (50 texts per author) and the testing corpus includes other 2,500 texts (50 texts per author) non-overlapping with training texts. Each folder contains 50 subfolders of author where 50 text files are saved. Figure 1 is an example of corpus files of each author.



*Figure 1. Example of corpus files of each author*

**Method**

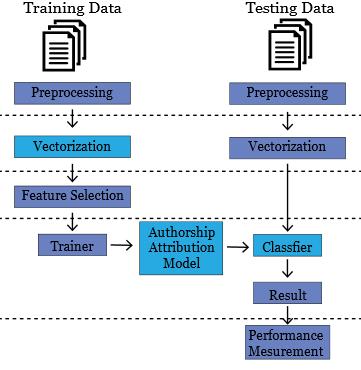
The study was accomplished based on Python. The task is to classify every document into 50 categories according to authors. For better analysis, we converted the text files into csv format, with author’s name as the first column and the content of each text as the fourth column. Figure 2 shows an example of the converted files.



*Figure 2. Example of converted files*

Different vectorizers were used in this study. We tried four kinds of word-level vectorizer – unigram term frequency, unigram Tf-idf, both unigram and bigram term frequency, both unigram and bigram Tf-idf, and a character level vectorizer – character 3-gram. In the entire experiment, we didn’t remove stop words, considering some authors may like to use many stop words in their articles which may reflect authors’ style. The minimum term frequency was fixed to 8 to get rid of some useless words. Lowercase setting and features to keep were tuned to figure out if these parameters would have an impact on the task. Since numbers usually occurred in the articles, and that would generate many useless features after vectorizations, we used a special symbol: ‘@’ to replace every digit of number. We want to find if this preprocessing strategy would be helpful in authorship attribution.

After vectorization, features extracted from the training dataset were used to build models. Three classifiers – Multinomial Naïve Bayes, SVM, KNN, and Decision Tree were tested on the testing dataset. The overall accuracy of each model was used as the evaluation measurement to compare their performance. The workflow is shown in Figure 3.



*Figure 3. Workflow of the experiment*

**Result**

After some experiments, we recorded the accuracy as our key metrics to measure the performance. Table 1 shows how algorithms and word vectorizers with different features affect the overall accuracy.

*Table 1. Accuracy with using different vectorizers and classifiers*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Algorithm | Features | unigram\_count | unigram\_tfidf | gram12\_count | gram12\_tfidf | char3grams |
| Naïve Bayes | lowercase=True | 0.6688 | 0.6296 | 0.6812 | 0.6456 | 0.6644 |
| lowercase = False | 0.6696 | 0.6324 | 0.6844 | 0.652 | 0.686 |
| lowercase = False, number to '@' | 0.6696 | 0.6292 | 0.6812 | 0.6468 | 0.67 |
| SVM | lowercase=True | 0.682 | 0.6936 | 0.6996 | 0.7084 | 0.6812 |
| lowercase = False | 0.6884 | 0.6956 | 0.7044 | 0.7096 | 0.6704 |
| lowercase = False, number to '@' | 0.684 | 0.6928 | 0.702 | 0.7104 | 0.6916 |
| Decision tree | lowercase = False, number to '@' | 0.4432 | 0.4536 | 0.49 | 0.4696 | 0.42 |
| KNN | k =3, lowercase = False, number to '@' | 0.3932 | 0.5536 | 0.3872 | 0.5596 | 0.466 |
| k=4, lowercase = False, number to '@' | 0.3976 | 0.5584 | 0.3888 | 0.5744 | 0.4608 |
| K=5, lowercase = False, number to '@' | 0.4072 | 0.5632 | 0.3888 | 0.5824 | 0.4672 |
| k=6, lowercase = False, number to '@' | 0.402 | 0.552 | 0.382 | 0.58 | 0.4532 |

**Comparing Features**

1. We compared word vectorizers (first four columns) and a character vectorizer (fifth column: character 3-gram). Word vectorizers had the best accuracy and character n-gram has some good results. We tested character n-gram with n=3 only in the task because the algorithm took significant amount of time to build sparse matrix than others when we tried to increase n value. We believe that character n-gram could be a reasonable option to distinguish authors’ writing styles if there is no time limitation.

2. For unigram and unigram+bigram, we find that unigram+bigram generally had better performance than only using unigram.

3. Compared with using word count, using Tf-idf to build SVM model improved the overall accuracy, but with Naïve Bayes model, word count had better performance.

4. For uppercase and lowercase, most results with keeping capital words had higher accuracy than changing all words into lowercase. We think one reason that may affect uppercase and lowercase results is that different authors have their own styles of writing long or short sentences (average words in sentences). Authors prefer short sentences would have more uppercase words, which may affect the results.

5. Changing digits into ‘@’ seemed had little impact on the task since it usually led to identical results. Considering we set the minimum term frequency as 8, many number features may be filtered out by this setting, which can interpret why preprocessing is unnecessary in our experiment.

**Comparing Algorithm**

Comparing the accuracy, overall performance: SVM > Naïve Bayes > KNN > Decision Tree

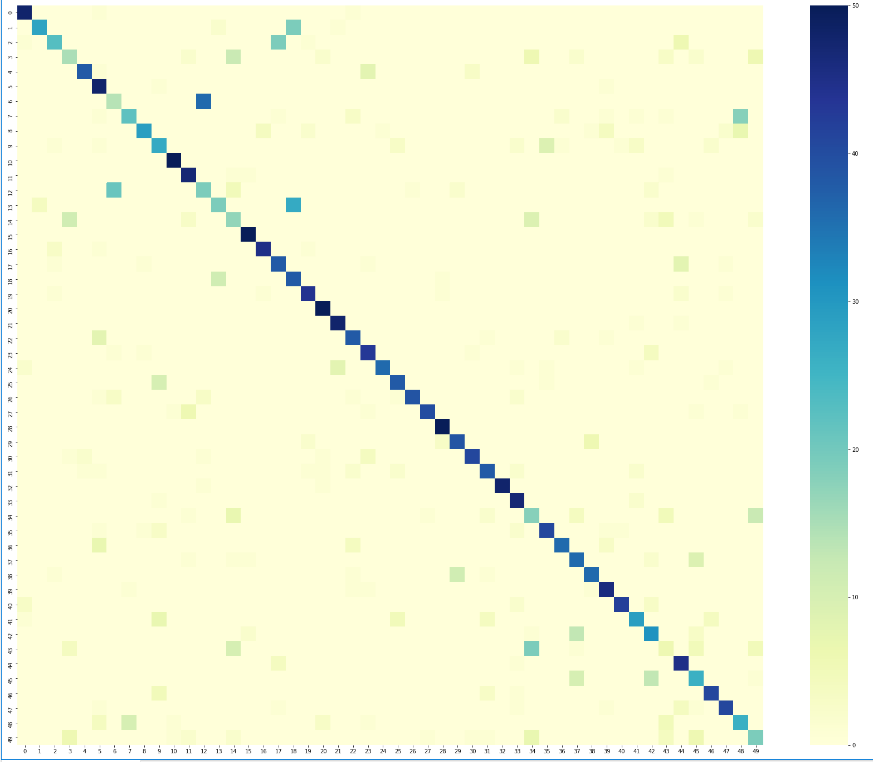
The task is a single-label and multi-class problem, which contains 50 authors in the dataset. The baseline accuracy is only 2%. In federal list papers problem, we could use decision tree to distinguish two authors, Hamilton and Madison using function word “upon”. We thought we might use the algorithm to find the tree that can classify 50 authors. However, the result is not good compared to other algorithms. Decision Tree algorithm does not perform well in tasks with so many classes. Also, KNN does not achieve good performance compared to Naïve Bayes and SVM algorithms. But we can find from the table that when k is 5 and vectorizer is ngram1,2 with TFIDF, KNN has its best accuracy which is nearly 60%. SVM performs best among these four algorithms.

Thus, we draw the conclusion that unigram and bigram with TFidf normalization can best represent texts for authorship attribution, and SVM is the most suitable classifier for the task since it consistently achieved good performance in the experiment.

**Error Analysis**

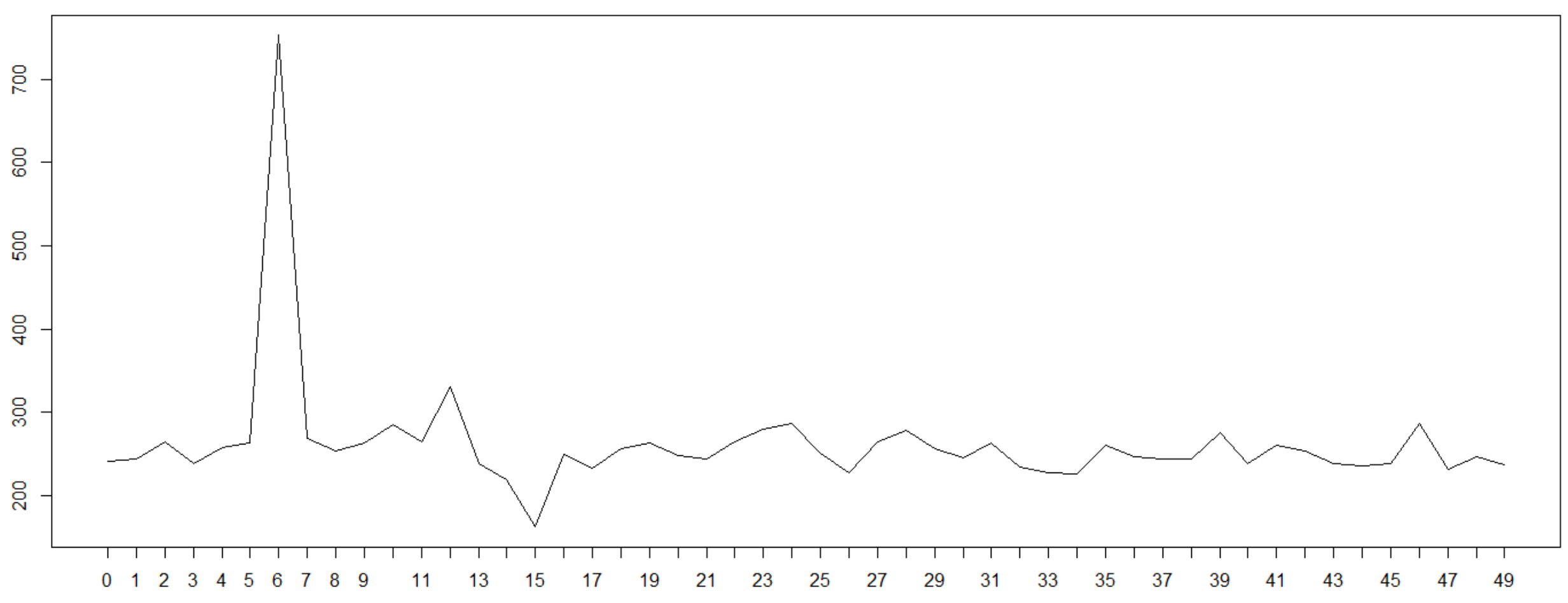
In the experiment, we also found some authors were more easily to be misclassified than others. We generated confusion matrix to get more insights. Figure 4 is an example of confusion matrix.

This is the confusion matrix for gram12\_tfidf using SVM algorithm. The color indicates that the degree of how many documents belong to the author are misclassified into the other author. The point means that the algorithm cannot distinguish author 6 and author 12. Thus, we try to find what happened between these two authors.



*Figure 4. Example of confusion matrix*

We extracted the word features for gram12\_tfidf using SVM for each author and compared the size of interaction set of features between author 6 and the rest authors (seen in Figure 5).



*Figure 5. Comparison of feature set*

We can find that the highest point is the interaction set size of comparing author 6 with himself. Author 6 has 753 features and author 12 has 529 features. The second highest point is with author 12, which means that author 6 have more features in common with author 12, compared to the rest authors. They have 330 features in common. This could tell why the algorithm cannot distinguish between author 6 and author 12 in some degree. The features in common only considered the presence of the features, while the algorithm uses the TFIDF instead.

**Conclusion**

By analyzing 2,500 texts from 50 authors, different types of vectorizers and classifiers were compared to find the most suitable model for authorship attribution. Comparing the overall accuracy, unigram and bigram with TF-idf normalization can best represent texts for authorship attribution, and SVM suits for the task since it generally had better performance than other classifiers.

We also found there is no need to select specific features, since preprocessing and weighting of features leads to identical results. Moreover, keeping capital words is helpful and the possible reason is some authors like to use short sentences which may generate more words with capital letters. In real-world, these typical vectorizers and classifiers can be used to help improve the accuracy of identifying the author. Besides, authors with similar vocabulary would be more easily to be misclassified using word-level features. Better feature selection strategy is required to solve this sort of problem.

**References**

1. Sebastiani F. Machine learning in automated text categorization. ACM Computing Surveys, 2002, 34(2): 1-47.

2. Houvardas J. and Stamatatos E. N-gram feature selection for authorship identification. International Conference on Artificial Intelligence: Methodology, Systems, and Applications, 2006, 12: 77-86