# Machine Learning-Based Authorship Identification in Web Fictions

Fangying Zhan
fzhan4@wisc.edu

Weijia Cao
wcao34@wisc.edu

Yuan Tian
tian74@wisc.edu

## Abstract

*To tell the authors of online web fictions from analyzing the text data using snippets cut from those works, we use machine-learning algorithms to try and identify different authors' writing styles. We collected data from FanFiction.Net, the most popular online archive of fan fictions. Four authors' chapters of fan fictions based off of Inception, Harry Potter, Avengers, and Naruto were cut into 851 snippets to constitute our dataset. All the text data were preprocessed using tokenization, lowercasing, and lemmatization. We compared two approaches of preprocessing the text data: LSM words + punctuations as features and stop word removal, two approaches of train-test splitting: a random 80%-20% split and a split based on themes, and two approaches for feature extraction: bag-of-words and tf-idf models. We employed multinomial naive Bayes classifier, which is empirically proven suitable and effective in its performance with tackling text data. We explored the performance of different approaches under 4 different scenarios to find the best setting for our model. Under Inception-non-Inception splitting, bag-of-words with multinomial naive Bayes classifier gave us the most reasonable and improved results to identify an author's writing style.*

## 1. Introduction

Authorship identification has been one of the popular challenges in the field of Natural Language Processing. But besides eliciting great research interest in the domain of computational linguistics, authorship identification has some significance in many applied areas.

As the use of electronic devices and scientific technology is popularized in the contemporary world, new issues have risen in the area of forensic sciences. For example, it has become more and more knotty to detect criminals who write ransom notes and threatening letters, send libellous or personal attack messages and phishing e-mails, and distribute illegal contents online or in the real world, as people can now take advantage of anonymity or counterfeit their identities and addresses to avoid investigation[1], and words are more often typed rather than written nowadays, leaving traditional criminal investigation techniques, such as authentication of handwriting signatures, nullified. As a result, being able to tell the authorship of pieces of writing from a known pool of candidates by simply analyzing the writing styles becomes rather meaningful in solving such conundrums.

In addition, authorship identification can also help find out most possible authors for literary works whose authors are anonymous or missing. Certainly, we do not want to go against the author's will if they intend to stay anonymous in order to reserve privacy, but the authorship for many literary works went unverifiable for various and complicated reasons. Victor Hugo published the first two editions of his novel *The Last Day of a Condemned* anonymously, more to tantalize his readers as a sales strategy. However, if he had accidentally been unable to ever reveal his authorship, and, in the most extreme case, nobody else had known that it was his work, then we might miss the chance to get to know that the book was "the most real and truthful of everything that Hugo wrote". Another example would be the long-enduring debate and guesses about the author of the most popular Arthurian tales, *Sir Gawain and the Green Knight*, ever since the work was rediscovered in the 19th century.[2] One can imagine how it would contribute to the literary world, if a possible author is found for these masterpieces, using statistically founded authorship identification techniques.

Further, authorship identification tells us not only if someone is likely to be the author of a work or not, but also helps us recognize the characteristics of different authors' writing styles, allowing us to do better in comparative literature and in studying different literary genres and literary movements and archiving literary works. It also helps us compare different author's written work to see if they are more similar to each other than they should be. Take what we will focus on in this project as an example. Performing authorship identification on snippets of fan fictions might help us understand if some fan fictions can truly imitate the writing of the original authors, if there is an issue of plagiarism in fan fictions, etc.

In this project, we will focus on solving the issue of authorship identification implementing machine learning methods. By using Multinomial Naive Bayes classifier and comparing different models, we will try to maximize the accuracy of identifying the correct authors of web fictions, allowing the best model to help identify anonymous articles. Moreover, our project will be devoted to classifying different authors' writing styles. Hence, it is plausible for us to compare writers, checking if some of them who readers formerly believed to be very similar in fact differ significantly in their wording, and if some of them that readers believed to apply totally different writing styles are actually more similar in their ”stylistic features”.

## 2. Related Work

There have been many works in the past that focused on improving the author analysis, including identification, verification, and profiling. Multiple methods have been implemented to achieve the goal such as uses of univariate or multivariate measures that can reflect the style of a particular author, as well as statistical machine learning techniques. But all methods attempted were relatively flawed until Michal Rosen-Zvi presented the first systematic study of authorship identification by using enhanced version of LDA which has the ability to identify all hidden topics from large numbers of features and presents them as LDA topics, thus serving for dimensionality reduction and making it attractive for text analysis problems.[3]

There are people's works applying the author identification into practical use. A K-nearest neighbor (kNN) algorithm that is based on the authorship verification method was introduced by Halvani et al. for identifying authorship and the task of the PAN 2013 challenge.[4] Zamani described authorship identification as an important method to look for the existence of plagiarism in law and journalism.[5] An end-to-end digital investigation (EEDI) was proposed by Ding et al. for a visualizable evidence-driven approach (VEA) in order to smooth the way for cyber investigation.[6] Furthermore, the news authorship identification task was dealt with by Zhou and Wang in different levels, including author, article, word, and sentence, utilizing both deep and non-deep algorithms with the GloVe word vectors that they used as the pre-trained word vectors.[7]

## 3. Proposed Method

### 3.0.1 Tokenization

By performing tokenization, we break up English sentences into words. In our project, we employ spaCy for tokenizing at word-level.

### 3.0.2 Lowercasing

English is a language with two cases (upper and lower). To address any confusion that can be caused by case, we convert every letter to its lower case in this project.

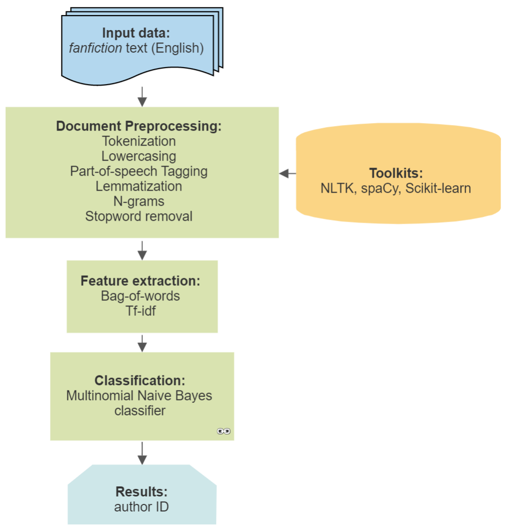


Figure 1. Flowchart

.

### 3.0.3 Lemmatization

For any given word in English, usually there is a group of words that represent the same meaning but are just different forms that derived from an inflected form. Such a group of words is called a lexeme, and the inflected form that conventionally represents the lexeme is the lemma.[8] By lemmatizing the text at word-level, we find the lemma for each word and group them together, so that the words with the same meaning can be treated as a single item.[9]

### 3.0.4 N-grams

An *n*-gram is a grouping of *n* contiguous words or characters.[3] We apply word-level *n*-gram models in our project. Take the phrase ”to be or not to be” as an example, an *n*-gram of size 1 (”unigram) would separate it into ”to, be, or, not, to, be” and an *n*-gram of size 3 (”trigram”) would separate it into ”to be or, be or not, or not to, not to be”.

*N*-gram models are widely used in Natural Language Processing for its ability to help predict what is coming next to a specific sequence of length *n*-1. They help us identify the language structures frequently used by a certain author. An appropriate choice of *n* allows us to capture both the general writing styles of authors and significant differences between them.

#### 3.0.5 Stopword Removal

Stops words are generally defined as the words with highly frequent appearances but minimal lexical meanings. It is possible that removing stop words can help us better predicting the authorship of text since these words are not very meaningful. However, it is also possible that the use of stop words is also a vital component of one's writing style. Therefore, in this project, we try authorship identification both with and without stop word removal to test both possibilities.

## 3.1. Feature Extraction

### 3.1.1 Bag of Words

The bag-of-words model treats text as a bag of words and corresponds each word to its frequency of appearance in the corpus, which is a resource that contains substantial structured texts.This model does not take into account the order or grammar of words.[3][10]

### 3.1.2 TF-IDF

The term frequency-inverse document frequency (tf-idf) model can be considered as a weighted variation of the bag-of-words model. The model balances out the frequency of a given word's occurrence in the text (term frequency) with the number of documents that contains the word in the corpus (inverse document frequency).[11] It is useful also because it can be applied via Naive Bayes classifier, which we will use in this project.

First, we normalize the term frequency by

$$\text{normalized term frequency} = \frac{tf(t,d)}{n_d},$$

where $tf(t,d)$ is the raw term frequency (i.e., how many times term $t$ appears in document $d$, and $n_d$ is the total count of terms in document $d$.

Then, we can calculate the tf-idf value.

$$\text{Tf-idf} = tf_n(t,d) \cdot idf(t)$$

Here, $tf_n(t,d)$ is the normalized term frequency, and $idf$ is calculated as follows

$$idf(t) = \log(\frac{n_d}{n_d(t)}),$$

where $n_d$ is the total number of documents and $n_d(t)$ is the total number of documents containing the term (word) $t$.[12]

## 3.2. Classification Model

### 3.2.1 Multinomial Naive Bayes Classifier

The Naive Bayes classifier for multinomial models is useful for classifying discrete features such as word counts. Although the multinomial distrubution usually requires integer feature counts, it also works for fractional counts such as tf-idf values practice.

To estimate the class-conditional probabilities in the multinomial model, the classifier will use term frequencies to calculate the maximum-likelihood estimate based on the training set:

$$P(\hat{x_i}|w_j) = \frac{\sum tf(x_i, d \in w_j) + \alpha}{\sum N_{d \in w_j} + \alpha \cdot V},$$

where

- $x_i$ is a word from the feature vector **x** of a particular sample
- $\sum tf(x_i, d \in w_j)$ is the sum of raw frequencies of word $x_i$ from all documents in the training sample belonging to class $w_j$
- $\sum N_{d \in w_j}$ is the sum of all term frequencies in the training dataset for class $w_j$
- $\alpha$ is an additive smoothing parameter
- $V$ is the size of the vocabulary (number of different words in the training set).

The class-conditional probability of encountering the text **x** can be calculated as the product from the likelihoods of the individual words (under the *naive* assumption of conditional independence).

$$P(\mathbf{x}|w_j) = P(x_1|w_j) \cdot P(x_2|w_j) \cdot \ldots \cdot P(x_n|w_j) = \prod_{i=1}^{m} P(x_i|w_j)$$

Zhao and Zobel (2005) examined the performances of Naïve Bayesian, Bayesian networks, 1-NN, 3-NN, and decision trees on authorship identification, and found that the Naïve Bayesian classifier outperforms kNN and decision tree classifiers. [13] Since it is empirically tested to be suitable and relatively effective for authorship identification tasks, we decided to apply this approach in our project.

### 3.2.2 Model Output

Same as the input class labels, the output class labels will also be the author ID's (IMP, MIX, SMC, or AVO).

## 4. Experiments

First, our experiment compared two approaches of preprocessing fan fiction text data: LSM words + punctuations as features and stop word removal. According to psycholinguistic points of vies, there are word categories for calculating language style matching, and these categories of words are called LSM words. Table 1 shows some examples of LSM words. Stop word removal, as is aforementioned, is to remove words with substantial occurrences and minor lexical meanings. For both approaches, we will apply tokenization, lowercasing, and lemmatization on our data by default.

| Category | Examples |
|---|---|
| Personal pronouns | I, his, their |
| Impersonal pronouns | it, that, anything |
| Articles | a, an, the |
| Conjunctions | and, but, because |
| Prepositions | in, under, about |
| Auxiliary verbs | shall, be, was |
| High-frequency adverbs | very, rather, just |
| Negations | no, not, never |
| Quantifiers | much, few, lots |

Table 1. Examples of LSM words

Second, we compared two approaches to split our dataset. The first one is to mix all the *Inception* and non-*Inception* snippets for each author, using 80% of them as our training set, and the remaining 20% of them as our test set. The second way is to use all the *Inception* snippets as our training set, and all the non-*Inception* ones as our test set. The second way of splitting might in fact logically makes more sense because in this way we may eliminate the impact of topics/themes and focus more on author's writing style, and thus may have better generalization performance on unseen data. Or say, we want see if our models are making predictions based on writing styles rather than the topics/themes.

Third, we compared two approaches for feature extraction: bag-of-words and tf-idf models, which differ in how they measure and weight the presence of words in text, as is discussed in Section 3.

### 4.1. Dataset

For obtaining our dataset, we manually collected our data from the most popular and famous web fiction archive, FanFiction.Net. For this project, we focused on the famous movie, *Inception*. We identified the top 4 fan writiers of *Inception*: imperfectandchaotic[14], mixed.vinyl[15], S.McCoy[16], and AvocadoLove[17]. Besides *Inception*, these authors also produced fan fictions of some other works, such as *Harry Potter* and *Avengers*. We then randomly selected fan fiction chapters written by these authors. Besides the chapters of *Inception*, the non-*Inception* fan fiction chapters of these authors are relatively topic-wise homogeneous, except for that we also collected imperfectandchaotic's and AvocadoLove's works of *Naruto*, a theme that the other two authors did not write on. These chapters were cut into 400- to 700-word snippets. Each snippet served as a sample. Our total dataset consists of 851 snippets. An overview of the themes of which these snippets are based off is shown in the table below.

| | IMP | MIX | SMC | AVO | **Total #** |
|---|---|---|---|---|---|
| *Inception* | 121 | 195 | 113 | 70 | 499 |
| non-*Inception* | 127 | 23 | 122 | 80 | 352 |
| **Total #** | 248 | 218 | 235 | 150 | 851 |

Table 2. Fan fictions of different authors

### 4.2. Software

For this project, we used Python for processing the data and implementing the machine-learning algorithms. To better tailor our project to Natural Language Processing, we imported the libraries Numpy, Scikit-learn, spaCy, Itertools, Matplotlib, and Pandas. The transformation of our data was supported by CountVectorizer and TfidfVectorizer in Scikit-learn.

## 5. Results and Discussion

In the data processing part, we used two way of splitting: 1.Randomly, 80% train, 20%test; 2.Inception as train, non-Inception as test. Moreover, we implemented two text preprocessing techniques to better looking writing styles: Stopword removal and our defined LSM technique. Hence, we will have four scenarios with different results.

For each scenario, we implemented 5 algorithms to fit our data: Multinomial Naive Bayes, KNN(baseline), Random Forests, XGBoost, and Light GBM. For each classifier, we used grid search to find the best hyperparameter tuning(Note: Naive Bayes doesn't have any hyperparameters to tune), so we can use the best performance of each classifier to do comparison. For feature extraction, we used two techniques: Bag of words and TF-IDF with each algorithm.

In this section, we will see each result with different combination of situations and compare them using 5 different evaluation metrics so we can have the most comprehensive conclusion.

### 5.1. Stopword removal on randomly spilt

#### 5.1.1 Feature Extraction

We began with the comparison between the two feature extractions techniques under splitting method 1 and using the Stopword removal technique. From the table below, we can see the clear results under five evaluation metrics we applied. Under each metric, Bag of words has performed better than Tf-idf though it is a popular technique in NLP. So we can conclude that in this scenarios, Bag of words would be our preferred strategy.

Note that in this table, I simply listed the results under Multinomial Naive Bayers. Though in our Baseline model(KNN), TF-IDF did get a better performance under each evaluation metric, applying other classifiers such as random forest, XGBoost, Light Gbm with Bag of words would allow us to get a improved result.

| | Bag of words | TF-IDF |
|---|---|---|
| Accuracy | 0.977 | 0.883 |
| Precision | 0.979 | 0.918 |
| Recall | 0.979 | 0.842 |
| F1 | 0.979 | 0.848 |
| Mcc | 0.969 | 0.851 |

Table 3. STRM-random: Feature Extraction result on Multinomial Naive Bayers

### 5.1.2 Model Selection

For model selection, as we have tuned each classifier, the comparisons are of their best performance. We can see from the table below to see that Multinomial Naive Bayers(0.977) had the best performance under Bag of words, though worst under TF-IDF and KNN got the best performance under TF-IDF, which is not surprising as we mention above that KNN performed reasonably well under TF-IDF.

Though due to limited space, We only listed results on accuracy, other evaluation metrics showed almost consistent results. Moreover since Bag of words is our preferred technique, we would consider Multinomial Naive Bayers as our best classifier.

| | Bag of words | TF-IDF |
|---|---|---|
| Multinomial Naive Bayes | 0.977 | 0.883 |
| Random forest | 0.953 | 0.947 |
| XGBoost | 0.959 | 0.947 |
| Light GBM | 0.959 | 0.942 |
| KNN | 0.912 | 0.959 |

Table 4. STRM-random: Model Selection result on Accuracy

### 5.1.3 Model Evaluation

Then we do model evaluation on training accuracy on our best classifier: Mulitnominal Naive Bayer and best features: Bag of words, applying .632+ Bootstrap method to best avoid bias. Results are in the table below:

| | |
|---|---|
| Mean Bootstrap score | 98.55% |
| Score std | 0.00510 |
| Confidence interval | [0.975, 0.994] |

Table 5. STRM-random: Model Evaluation on .632+

## 5.2. Stopword removal on Inception,non-Inception

### 5.2.1 Feature Extraction

In this scenario, we compare the two feature extraction techniques under Stopword removal and splitting method 2:Inception,non-Inception. Again under this different way of splitting, we compared the scores of each technique using Multinomial Naive Bayers as representative. We can see from the table below that though Bag of words had worse performance compared to under spillting 1, it is much better than TF-IDF's performance. Other evaluation metric also presented the same result except for KNN as in the last senario. There is huge gap between the performance of these two techniques. So it is with no doubt that Bag of words is again a preferred technique.

| | Bag of word_inc/rev | TF-IDF_inc/rev |
|---|---|---|
| Accuracy | 0.727 / 0.457 | 0.081 / 0.400 |
| Precision | 0.741 / 0.340 | 0.265 / 0.221 |
| Recall | 0.734 / 0.546 | 0.264 / 0.426 |
| F1 | 0.604 / 0.421 | 0.055 / 0.282 |
| MCC | 0.647 / 0.370 | 0.085 / 0.296 |

Table 6. STRM-inception: Feature Extraction result on Multinomial Naive Bayers

### 5.2.2 Model Selection

For model selection, from the table below it is Multinomial Naive Bayers had the best performance under Bag of words, far head the others, and KNN got the best performance consistent with last scenario (Note: the results came from before we made the split into strictly 8:2, but it is substantial for model selection). Also, other evaluation metrics showed almost the same results results as accuracy. Then as Bag of words is our preferred technique, we would consider Multinomial Naive Bayers as our best classifier again in this scenario.

| | Bag of words | TF-IDF |
|---|---|---|
| Multinomial Naive Bayes | 0.707 | 0.074 |
| Random forest | 0.239 | 0.580 |
| XGBoost | 0.332 | 0.233 |
| Light GBM | 0.347 | 0.256 |
| KNN | 0.361 | 0.580 |

Table 7. STRM-inception: Model Selection result on Accuracy

### 5.2.3 Model Evaluation

Then we do model evaluation on training accuracy on our best classifier: Mulitnominal Naive Bayer and best features: Bag of words. This time we evaluate on balanced accuracy.

| | Inc | Rev |
|---|---|---|
| Average per class accuracy | 86.36% | 72.86% |

Table 8. STRM-inception: Average per class accuracy

## 5.3. LSM on randomly spilt

### 5.3.1 Feature Extraction

Then We switch into the comparison between the two feature extractions techniques under splitting method 1 and using different a different technique this time: LSM. From the table below, it is clear that under each evaluation metrics we applied, though we were using a different text preprocessing method this time, Bag of words still had a much better performance than Tf-idf. So we can conclude that in this scenarios, Bag of words would be our preferred strategy.

Almost consist with results on former settings, though in KNN and XGBoost, TF-IDF did get a better performance, under each evaluation metric, applying other classifiers Bag of words gave higher scores. In fact, contrast to in Multinomial Naive Bayer Bag of words took the large lead, other classifiers presented closer results between the two.

| | Bag of words | TF-IDF |
|---|---|---|
| Accuracy | 0.930 | 0.556 |
| Precision | 0.932 | 0.278 |
| Recall | 0.924 | 0.490 |
| F1 | 0.927 | 0.355 |
| Mcc | 0.906 | 0.446 |

Table 9. LSM-random: Feature Extraction result on Multinomial Naive Bayers

### 5.3.2 Model Selection

For model selection, Multinomial Naive Bayers, as usual, had the best performance under Bag of words and with huge advantage over TF-IDF. XGBoost got the best performance under TF-IDF. . Since Bag of words is our preferred technique and other evaluation metrics showed consistent results as accuracy, we would consider Multinomial Naive Bayers as our best classifier.

### 5.3.3 Model Evaluation

Then we do model evaluation on training accuracy on our best classifier: Mulitnominal Naive Bayer and best features: Bag of words under .632+ Bootstrap method. Results are in the table below:

| | Bag of words | TF-IDF |
|---|---|---|
| Multinomial Naive Bayes | 0.930 | 0.556 |
| Random forest | 0.877 | 0.871 |
| XGBoost | 0.877 | 0.883 |
| Light GBM | 0.918 | 0.906 |
| KNN | 0.725 | 0.760 |

Table 10. LSM-random: Model Selection result on Accuracy

| | |
|---|---|
| Mean Bootstrap score | 93.94% |
| Score std | 0.00907 |
| Confidence interval | [0.922, 0.957] |

Table 11. LSM-random: Model Evaluation on .632+

## 5.4. LSM on Inception,non-Inception

### 5.4.1 Feature Extraction

This time, we compare the two feature extraction techniques under LSM and splitting method 2:Inception,non-Inception. Again under this way of splitting, we use Multinomial Naive Bayers as an example to compare scores under different evaluation metric. We can see from the table that similar to scenario 2 under splitting Bag of words had worse performances, it is still much reliable than TF-IDF.

| | Bag of word_inc/rev | TF-IDF_inc/rev |
|---|---|---|
| Accuracy | 0.717 / 0.471 | 0.061 / 0.329 |
| Precision | 0.692 / 0.365 | 0.015 / 0.178 |
| Recall | 0.778 / 0.593 | 0.250 / 0.353 |
| F1 | 0.665 / 0.431 | 0.029 / 0.219 |
| MCC | 0.646 / 0.404 | 0.000 / 0.188 |

Table 12. LSM-inception: Feature Extraction result on Multinomial Naive Bayers

### 5.4.2 Model Selection

For model selection, from the table below it is still Multinomial Naive Bayers had the best performance under Bag of words, only one with score over 0.5, and KNN got the best performance consistent with last scenario (Note: the results came from before we made the split into strictly 8:2, but it is substantial for model selection). Though KNN has a even higher absolute score under TF-IDF than Multinomial Naive Bayer under Bag of words. Its relative advantage under TF-IDF is not great enough. Moreover, as we prefer using Bag of words, then we would still prefer Multinomial Naive Bayers as our best classifier again in this scenario.

| | Bag of words | TF-IDF |
|---|---|---|
| Multinomial Naive Bayes | 0.648 | 0.065 |
| Random forest | 0.432 | 0.452 |
| XGBoost | 0.506 | 0.568 |
| Light GBM | 0.557 | 0.594 |
| KNN | 0.557 | 0.653 |

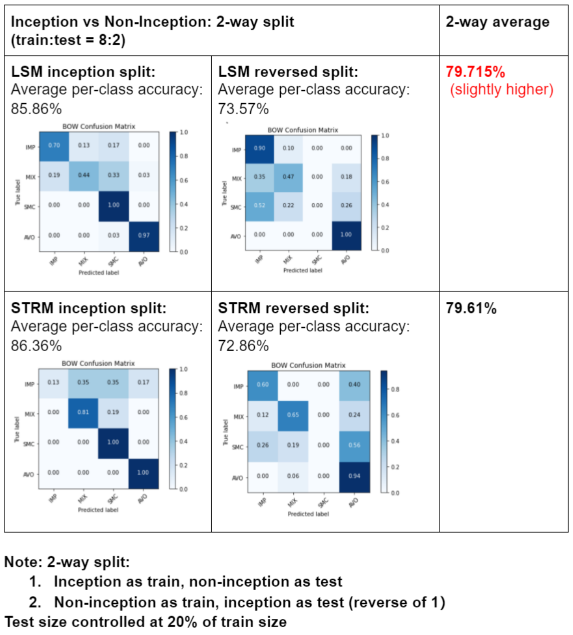
Table 13. LSM-inception: Model Selection result on Accuracy

### 5.4.3 Model Evaluation

Hence, after deciding to implement Multinomial Naive Bayers, Bag of words would be a better choice. Then we do model evaluation on training accuracy on this combination. The evaluation is on Balanced accuracy. Results are in the table below:

| | Inc | Rev |
|---|---|---|
| Average per class accuracy | 85.86% | 73.57% |

Table 14. LSM-inception: Average per class accuracy



| Inception vs Non-Inception: 2-way split (train:test = 8:2) | | 2-way average |
|---|---|---|
| **LSM inception split:** Average per-class accuracy: 85.86% | **LSM reversed split:** Average per-class accuracy: 73.57% | 79.715% (slightly higher) |
| **STRM inception split:** Average per-class accuracy: 86.36% | **STRM reversed split:** Average per-class accuracy: 72.86% | 79.61% |

BOW Confusion Matrix
IMP 0.70 0.13 0.17 0.00
MIX 0.19 0.44 0.33 0.03
SMC 0.00 0.00 1.00 0.00
AVO 0.00 0.00 0.03 0.97
True label
Predicted label
BOW Confusion Matrix
IMP 0.90 0.10 0.00 0.00
MIX 0.35 0.47 0.00 0.18
SMC 0.52 0.22 0.00 0.26
AVO 0.00 0.00 0.00 1.00
BOW Confusion Matrix
IMP 0.13 0.35 0.35 0.17
MIX 0.00 0.81 0.19 0.00
SMC 0.00 0.00 1.00 0.00
AVO 0.00 0.00 0.00 1.00
BOW Confusion Matrix
IMP 0.60 0.00 0.00 0.40
MIX 0.12 0.65 0.00 0.24
SMC 0.26 0.19 0.00 0.56
AVO 0.00 0.06 0.00 0.94
IMP MIX SMC AVO

**Note: 2-way split:**

1. **Inception as train, non-inception as test**
2. **Non-inception as train, inception as test (reverse of 1)**

**Test size controlled at 20% of train size**

Figure 2. LSM vs. Stopword Removal

## 5.5. LSM vs. Stopword Removal

In this section, we compare the results of these two text preprocessing technique. As our focus is to examine whether our models are making predictions based on writing styles rather than the topics/themes, we applied our second way of splitting to eliminate the impact of topics/themes. Though we can generally see the results from above that random splitting in fact gave us a better score, implementing the same classifier and features on a different splitting caused such a huge gap made us think that there should exist overfitting. So for this comparison, we will use the result from inception splitting as representative.

In this figure below, we used balanced accuracy to combine the results from two cases. We can see that though the difference is very small, LSM slightly defeat Stopword Removal as a better text processing technique.

## 5.6. Discussion

Though we tried our best to present most comprehensive and generalized model, there are still some limitations of our project. Since our dataset are manually collected from FanFiction.net, the size it is quite limited. For example, our dataset only contained four authors' most famous work. There might be situations that one author's master piece are not that representative of his/her style while some other writings represent the essence of the writing style. These four authors are selected since their works are relative popular and share similar themes, so our model can not be generalized to other topics yet.

The quality of data also matters. An author's writing style is in fact not static. Their writing style would change with their experiences over time, so even for the same author, his/her writing style might be completely different. On the other hand, different authors in the same period might have a similar writing styles based on similar education and experiments. These are situations our model are unable to fix right now. Our goal on identifying author's writing style would be largely influenced by these two possibilities.

## 6. Conclusions

### 6.1. Conclusions

From our comparisons on different scenarios and dimensions, we can conclude that for feature extraction technique, Bag of words is no doubt our best choice. In each scenario, Bag of words is far head TF-IDF. Similar to Bag of words, for model selection, Multinomial Naive Bayes has won each of the comparison under different cases. Though KNN performed surprisingly good under TF-IDF, due to fact that Bag of words would be a prior choice, we will take Multinomial Naive Bayes as our best classifier. However, it is not that obvious which text preprocessing technique we should use. LSM finally got a slightly better performance over Stopword Removal, so we can say Writing styles are more likely to be found in LSM words, whereas themes/topics are more likely to be found after removing stopword.

In conclusion, in our NLP project, we impletmented several text processing techniques listed in provious section and splitted into Inception, Non-Inception sets as our data to explore an author's writing style. For feature extraction and model selection, Bag of words with Multinomial Naive

Bayers gave us the most reasonable and improved result. The final accuracy of our model on Author Identification is about 80%. In other words, 80% of unknown text example can be correctly identified to its author.

### 6.2. Future directions

As one of the limitations of our project is the failure to take into account the changes in one's writing style over time, future studies can focus more on time as a factor and maybe introduce time series data. Nevertheless, our dataset is still relatively small. Under ideal circumstances where we have a significant amount of data, we can first perform unsupervised learning, grouping fan fictions of similar writing styles together and then perform further authorship identification.

Although we first intended to apply *n*-grams for preprocessing our data, we did not do so due to a lack of time. Future research can probe in whether making use of this technique will lead to a more reasonable dataset and eventually more desirable performances.

Also, although our project focused on identifying the authors of online fan fictions, authorship identification is in fact much more versatile, as we discussed in the first section of this report. Therefore, future work can explore the use and performance of authorship identification in more fields, such as law, journalism, and forensic sciences.

Last but not least, the way we collected our data was to manually copy and split them from the website FanFiction.Net. Future studies can try to develop approaches to automatically and randomly capture such chapters from the internet and cut them into snippets of desirable sizes, so that an exhaustiveness and a randomness can be better guaranteed.

## 7. Contributions

During the process of carrying out the project, every team member made contributions to each part, with different foci on different tasks. Fangying is mainly dedicated to collecting data and running the models. Weijia and Yuan are mainly responsible for result interpretation and the project report. Every group member contributed to background literature research as well.

## References

[1] L. Srinivasan and C. Nalini. An improved framework for authorship identification in online messages. *Cluster Computing*, 22(5):12101 – 12110, 2019.

[2] BookRags. Sir gawain and the green knight author/context.

[3] Imran Sarwar Bajwa Waheed Anwar and Shabana Ramzan. Design and implementation of a machine learning-based authorship identification model. *Hindawi*, 2019(7):14, 2019.

[4] Oren Halvani, Martin Steinebach, and Ralf Zimmermann. Authorship verification via k-nearest neighbor estimation. *Notebook PAN at CLEF*, 2013.

[5] Hamed Zamani, Hossein Nasr Esfahani, Pariya Babaie, Samira Abnar, Mostafa Dehghani, and Azadeh Shakery. Authorship identification using dynamic selection of features from probabilistic feature set. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 128–140. Springer, 2014.

[6] Steven HH Ding, Benjamin CM Fung, and Mourad Debbabi. A visualizable evidence-driven approach for authorship attribution. *ACM Transactions on Information and System Security (TISSEC)*, 17(3):1–30, 2015.

[7] L Zhou and H Wang. News authorship identification with deep learning. In *Conference and Labs of the Evaluation Forum, Portugal*, 2016.

[8] Lemma (morphology), Nov 2020.

[9] Lemmatisation, Oct 2020.

[10] Text corpus, Dec 2020.

[11] Tf–idf, Dec 2020.

[12] Sebastian Raschka. Naive bayes and text classification, Oct 2014.

[13] Ying Zhao and Justin Zobel. Effective and scalable authorship attribution using function words, Oct 2005.

[14] imperfectandchaotic, 2020.

[15] mixed.vinyl, 2020.

[16] S. mccoy, 2020.

[17] Avocadolove, 2020.