King Saud University
College of Computer and Information Sciences
Information Technology Department

IT469
Human Language Technologies

The Catcher: Author Identification

Prepared by:

Group#5	
Leader Email: 438201991@student.ksu.sa	
Nada Almutairi	[redacted]
Lama Alsulaiman	[redacted]
Sara Alrasheed	[redacted]
Reem Alghannam	[redacted]

Supervised by:
Dr. Afnan AlSubaiheen
L. Nora Almadi

Table of Contents

Table of Figures

Table of Tables

Roles and Responsibilities

Student name	Role	Review section	Correction
Reem AlGhannam	Programming & Data Analyst	Introduction, Data Section & Conclusion	Nada
Nada AlMutairi	Programing & Data Analyst	Data Cleaning, Preprocessing & Conclusion	Reem
Lama AlSulaiman	Programming & Data Analyst	Modeling, Evaluation & Future Work	Sara
Sara AlRasheed	Programming & Data Analyst	Features Engineering & Future Work	Lama

Table 1 - Roles and Responsibilities

Introduction

In recent years, authorship analysis of anonymous texts in the Internets has received some attention in cyber forensic and data mining communities. Authorship analysis is the study of linguistic and computational characteristics of the documents written by known or unknown authors.

The problem is determining the true author of texts was a task of social interest from the moment it was possible to attribute the authorship of words. With the development of statistical techniques and due to the wide availability of data that can be accessed from the internet, authorship analysis has become a very practical option.

The aim of this project is to classify texts into classes based on the stylistic choices of their authors, the key idea is to exploit the differences of the writing styles of the authors and use this information to build our models. We will use such word embedding, then we will compare the performance of TFIDF and Pre-trained word embeddings on classic Machine Learning models. This will be performed on an author identification dataset.

We will present an overview and analysis of the processing, we follow the first 5 steps of the Natural Language Processing (NLP) pipeline in this phase starting with Data Acquisition, Text Cleaning and Preprocessing, Feature Engineering and Evaluation. Each pipe will be illustrated in a separate section.

Data

We have been used the Router_50_50 datasets which is an English-Language dataset. Initially, the data was given to us separated in two folders one for the training, and the other one for the testing. Each folder has 50 sub-folder naming by authors names, each of which has 50 text files naming randomly.

Since the text files were naming randomly and we cannot distinguish the texts of each author when collecting the texts in one document, we start manually renaming each text file with the author name following by a number from 1 to 50 and then we combine all of them in one folder. After reading these texts, our data contains two columns: the author's name and the text. The previous steps were done in both training and testing folders.

Given a text, we are trying to predict the author's name from a list of 50 classes. Each class represents an author.

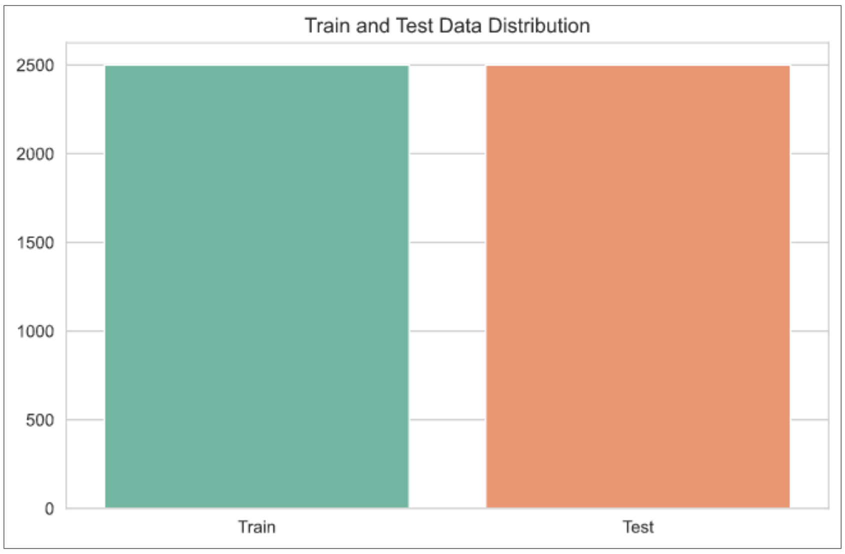

Figure 1- Train and Test Distribution

Figure 1 shows the distribution of our data. We have a total of 5000 observations divided equally for train data and test data.

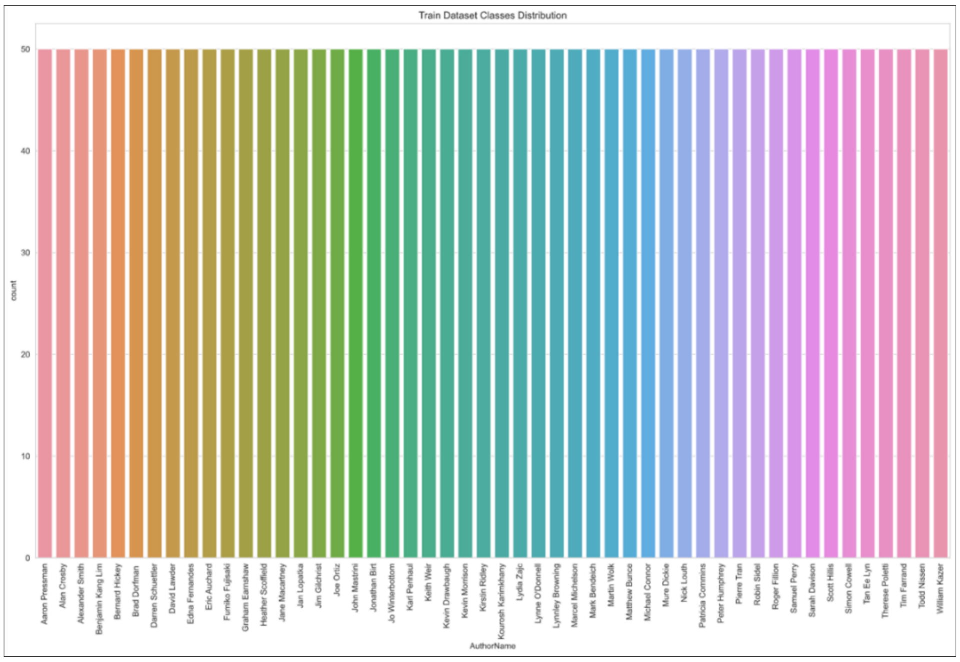

Figure 2 - Train Data Distribution

Figure 2 shows that our training data is well distributed, and we have 50 classes in total and each class has 50 observations.

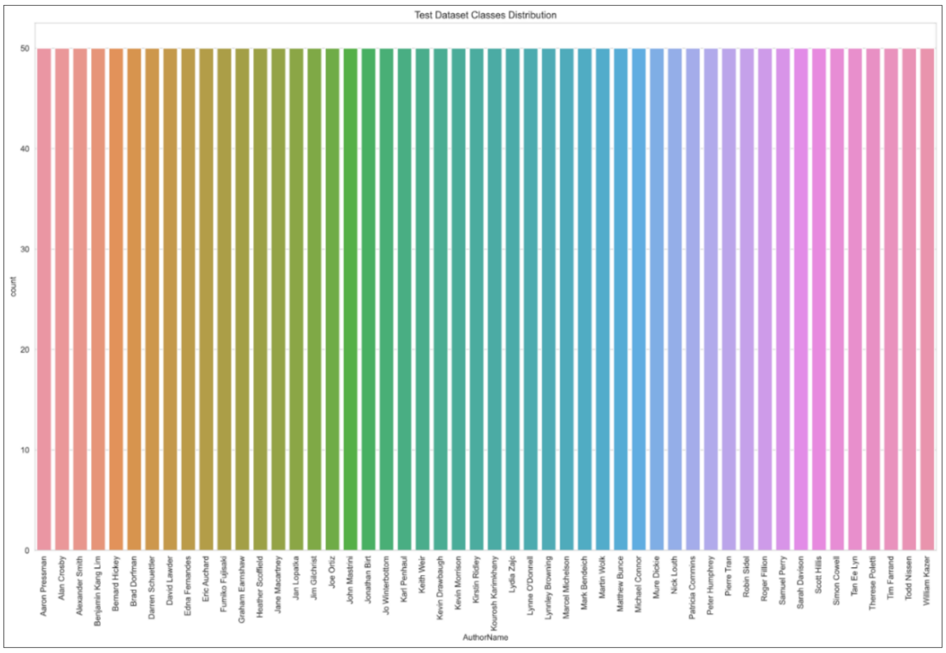

Figure 3 - Test Data Distribution

Figure 3 shows that our testing data is well distributed, and we have 50 classes in total and each class has 50 observations.

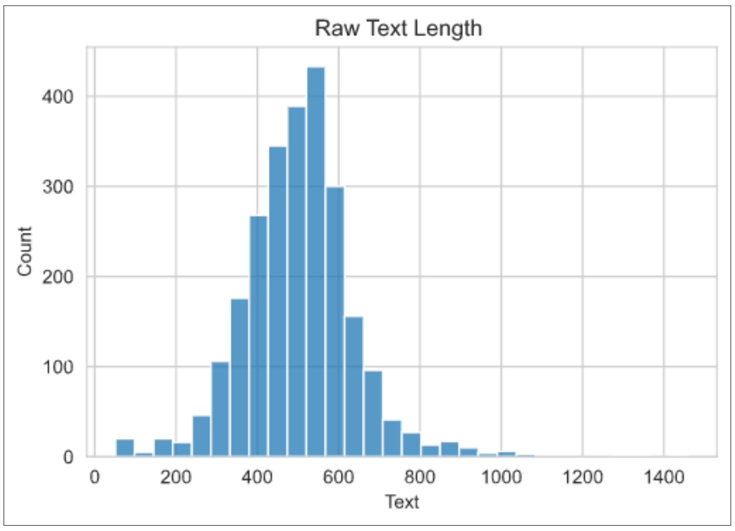

Figure 4 - Raw-Text Length Distribution

Figure 4 shows the total number of words for each observation in our training set and this is was done to help us understanding our dataset better.

Exploratory Data Analysis

Based on our research in the domain of author identification [1][2]. We found out that in order to understand our dataset and to perform initial investigations on them, we tried to explore the following features listed below:

- Word Cloud for the Data
- Length of raw text
- Number of punctuations
- Sentence length
- Number of digits
- Average word Length
- Average sentence Length

For each feature above, we observed the relationship between the class and that feature by plotting them to find such a pattern. Doing that will further help us to understand our dataset better and to check whether these features will be helpful as an input to our model or not.

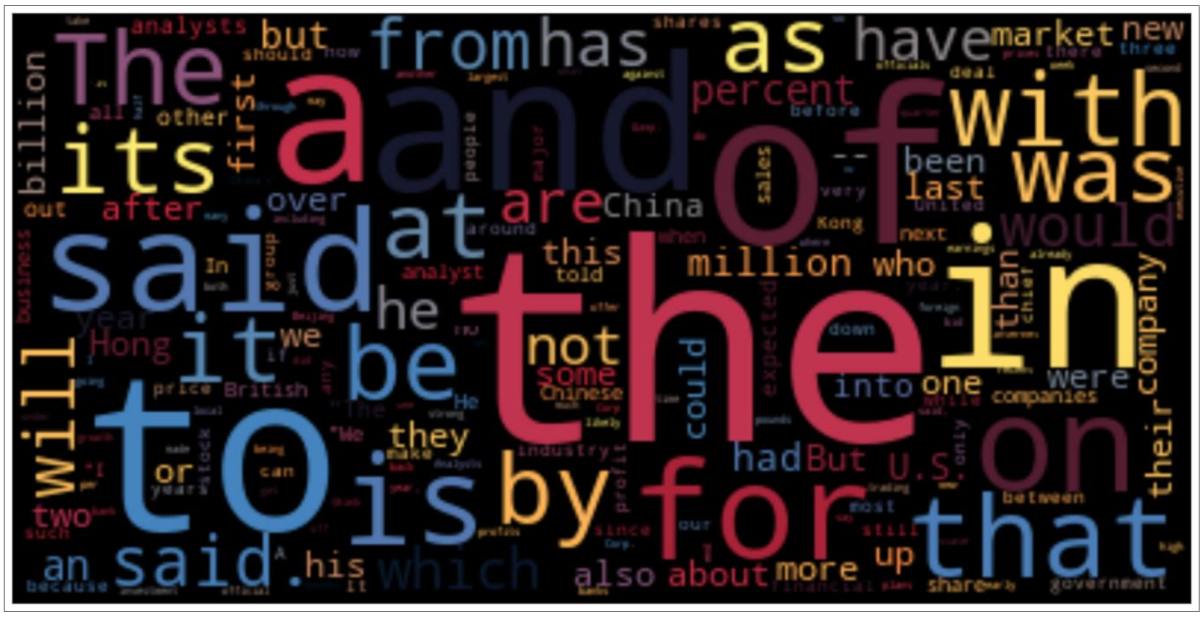

Figure 5 - Word Cloud of Raw Text

Figure 5 shows the generation of word cloud to check the top 200 most frequent words in our dataset. Immediately we can see that most of the mentioned words are stop words. Therefore, we have to remove these stop words.

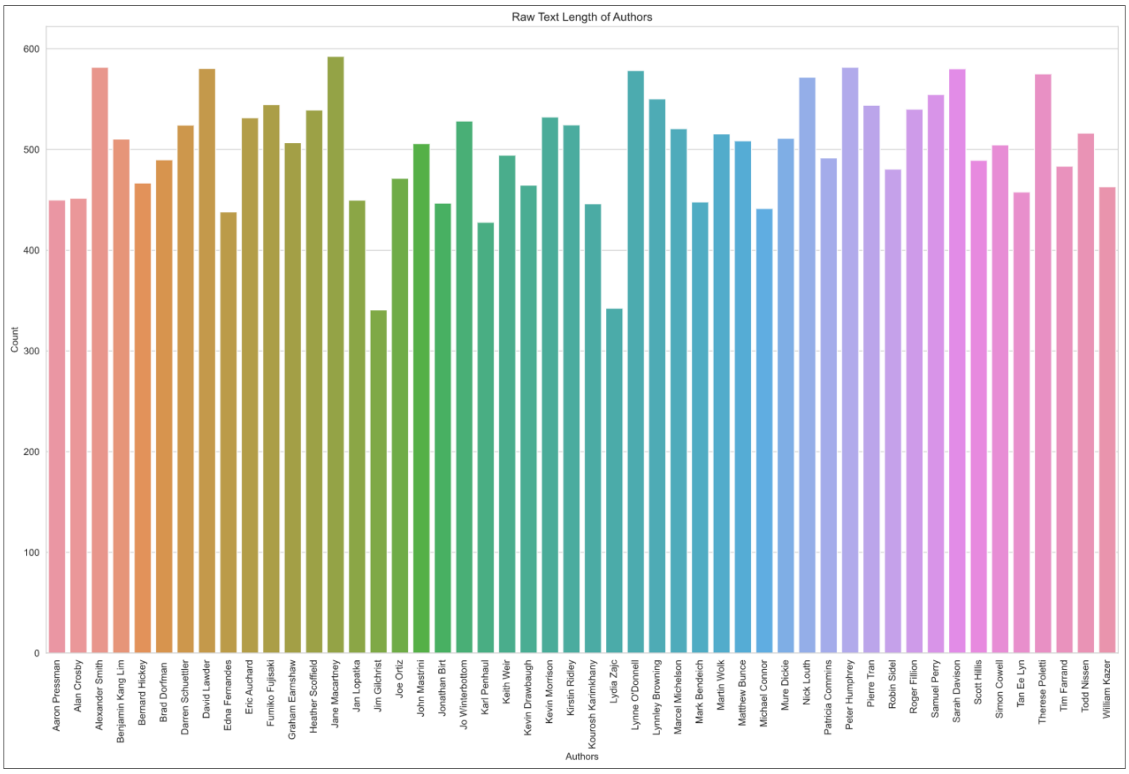

Figure 6 - Average Text Length for each Author

Figure 6 shows the average text length for each author in the dataset. Since some authors have similar average text length and some of them have completely different average text length, we can conclude that there is no connection between the average length of the text and the author itself. Hence, taking this as a feature to our model will not make it perform any better.

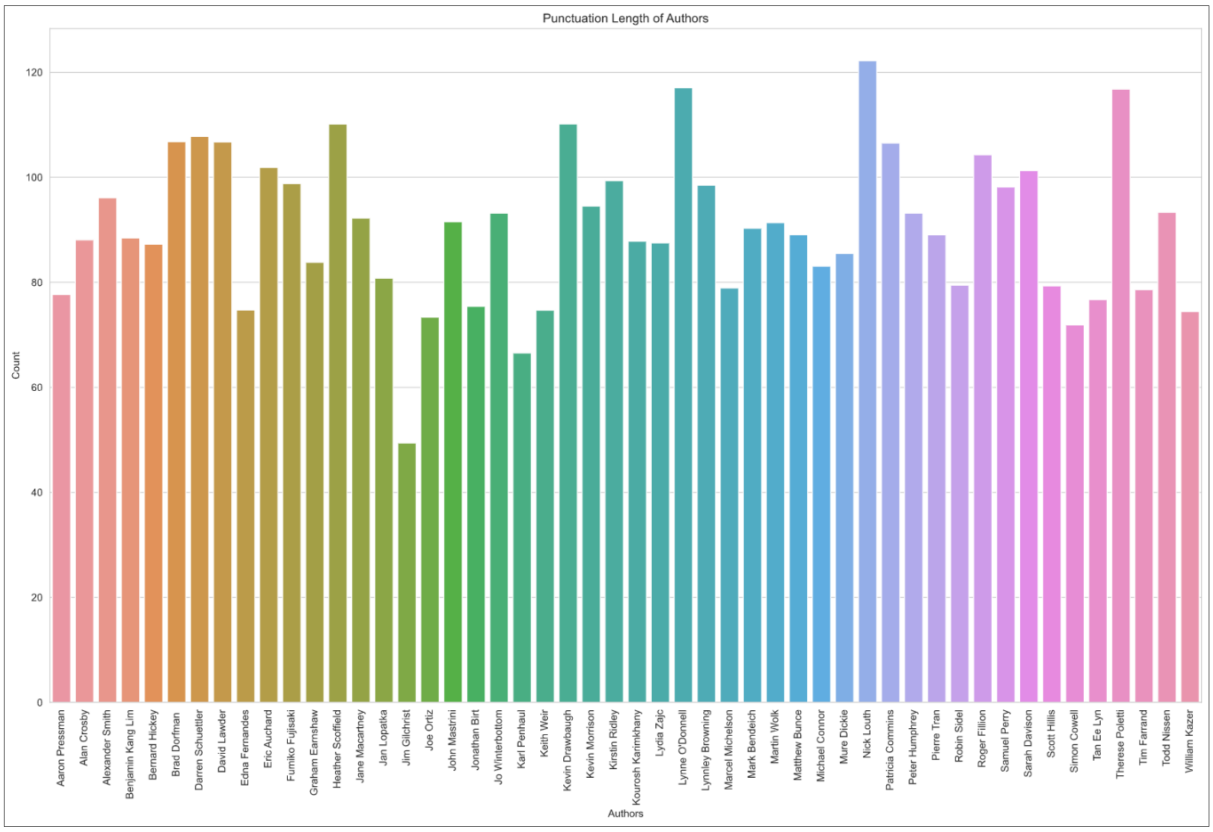

Figure 7 - Average Punctuation Length for each Author

Figure 7 shows the average punctuation length for each author in the dataset. Since some authors have similar distribution and others have a very random distribution, we can conclude that the average punctuation length will not be helpful for our model.

In general, punctuation is used to tell the reader how the sentence is structured and how it supposes to be read. Since we are dealing with English text, the punctuations would be similar across all the texts and this will not be a good feature for our dataset.

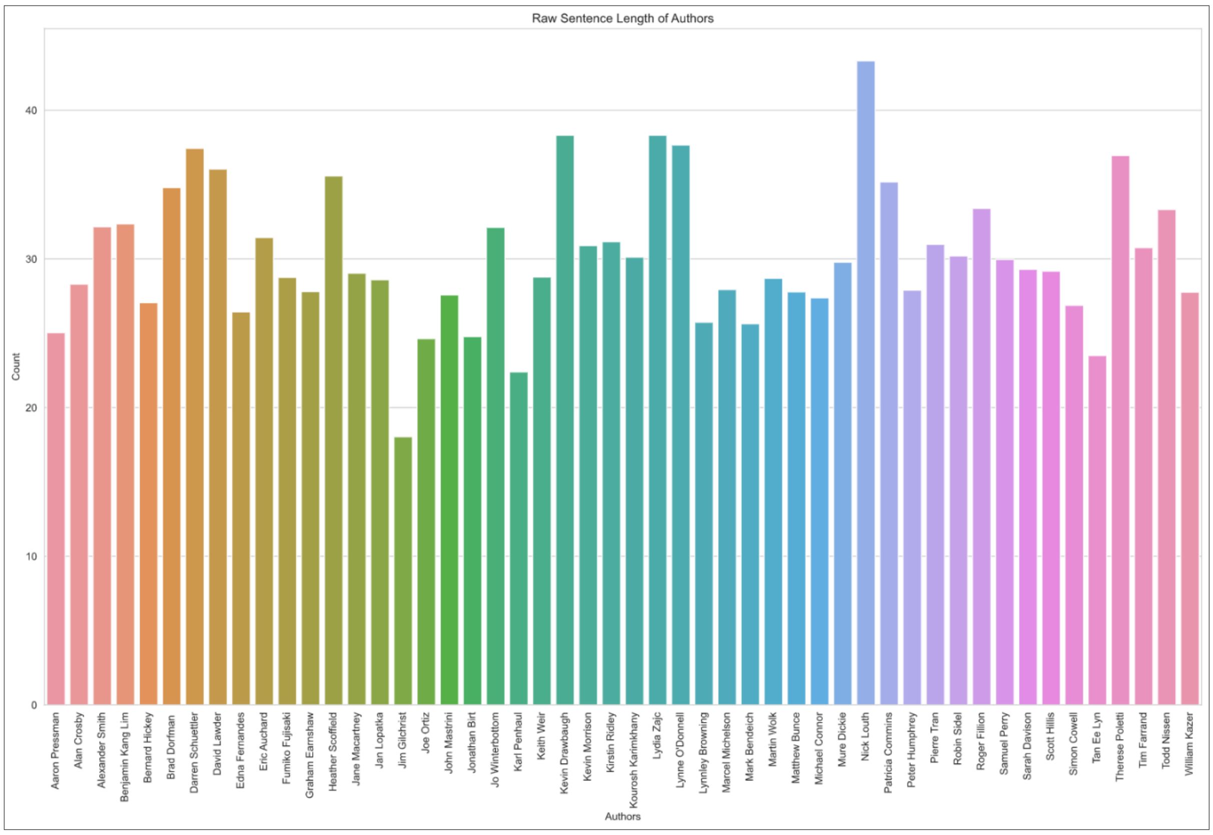

Figure 8 - Average Sentence Length for each Author

Figure 8 shows the average sentence length for each author. Since some authors have similar distribution and others have a very random distribution, we can conclude that the average sentence length cannot help us to tell which author wrote a specific text.

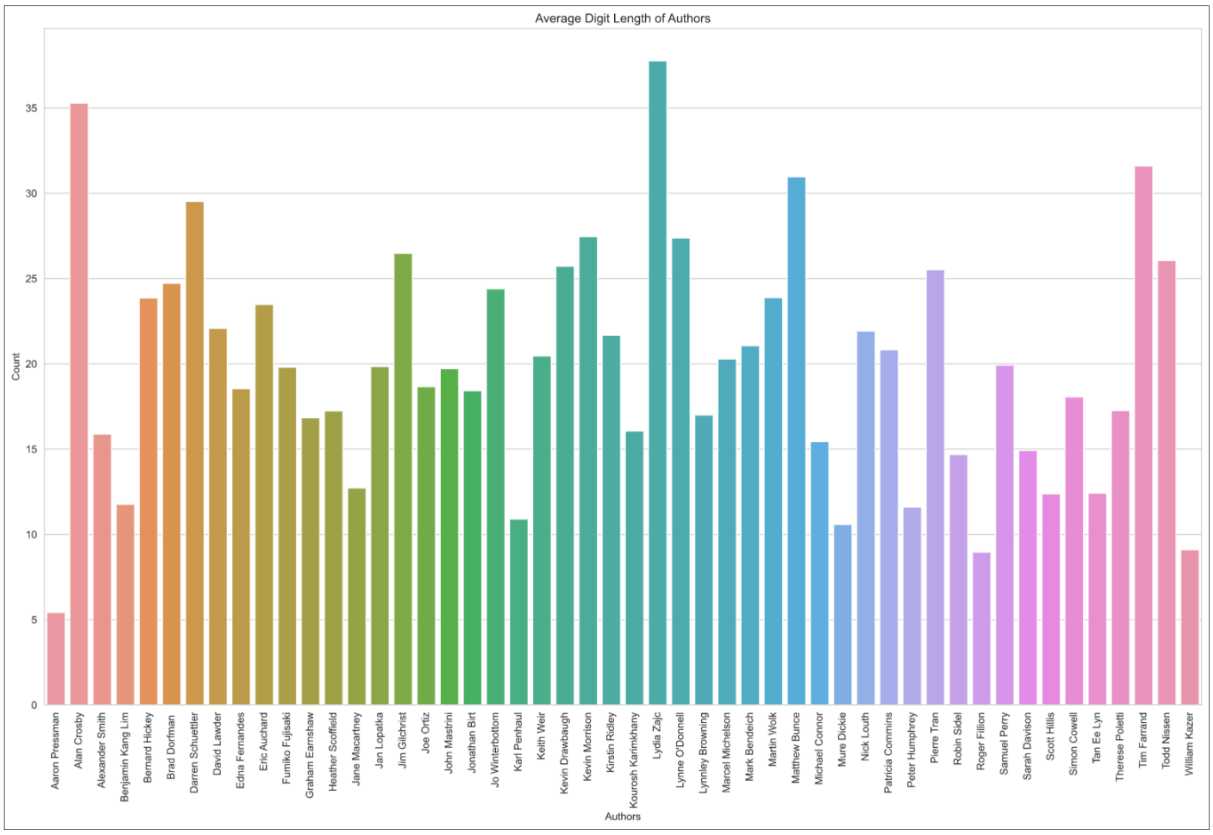

Figure 9 - Average Digit Length for each Author

Figure 9 shows average digit length for each author. Digits do not help us in classifying the author for this dataset as they are randomly distributed.

Text Cleaning

Text data usually has a lot of noise and for building ML model that performs well, our data needs to be cleaned. There are several steps that can be done to clean text data. Most data cleaning tasks are domain specific [3].

We used the NLTK (Natural Language Toolkit) which is one of the oldest and most used Python libraries for Natural Language Processing that supports stop word removal [4]. The regular expressions have been used in the cleaning process.

In this section, we will describe in detail our cleaning steps.

- Removing "\n" and "\" from the text:

Because our data was read through text files, many new line tags were added to our data frame. We would be utilizing pythons string library and the replace function to get rid of them.

- Removing URLS from the text:

Because our text may contain references or other resources which are not helpful to our task because the URL itself does not hold any important information and it will not help us in the classification task [5].

- Removing punctuations:

When preprocessing text data, we often remove punctuations because it can add noise to our data [1]. For example, a word like “hello” and “hello!” might have the same meaning to us, but for the machine learning model, they are 2 different words. In return, this will not help the machine learning model in learning process. As shown previously in figure 6, we can see that there is no connection between the authors and the average punctuation length, and we found out that it did not give us any useful information.

- Removing stop words:

Stop words occur in abundance, hence providing no unique information and will bring noise to the data. We used the NLTK stop word list to remove the stop words from our dataset.

- Normalization

We were not sure which normalization technique will perform better with our task, so we apply different models with both stemming and lemmatization. The only difference between stemming and lemmatization, apart from brining the word to its root form [6], is that stemming is a very rough way of bringing words down to their root form. Mostly, the root form does not exist in the dictionary. In lemmatization, we bring the word down to their base form and that form is usually available in the dictionary.

These techniques are usually performed on text data to reduce the number of features. Compared to stemming, lemmatization is slower to perform because it is a dictionary-based approach [7]. For example, if we have the word “Studies”, stemming will bring it down to “studi” and lemmatization will bring it down to “study”.

Figure 10 below shows the cleaned text length distribution.

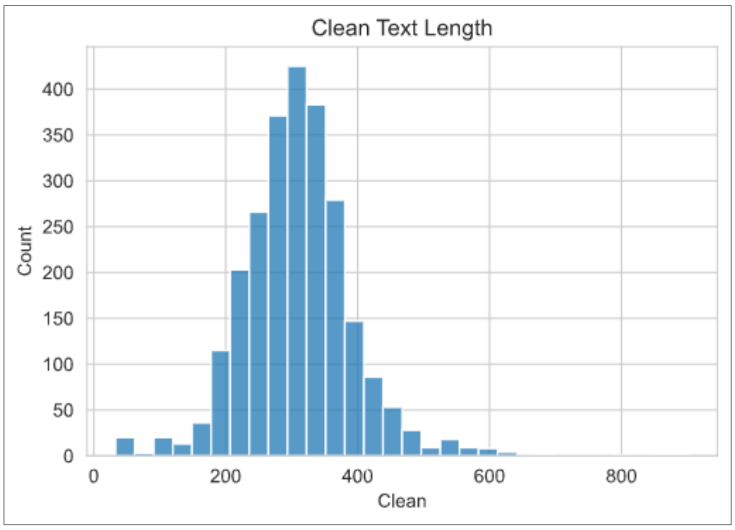

Figure 10 - Clean-Text Length Distribution

Compared to figure 4, we can see that the length of the data has been reduced from 1500 to 800. We can see most of our text length falls between 200 to 400.

Figure 11 shows the word cloud generated after cleaning the data.

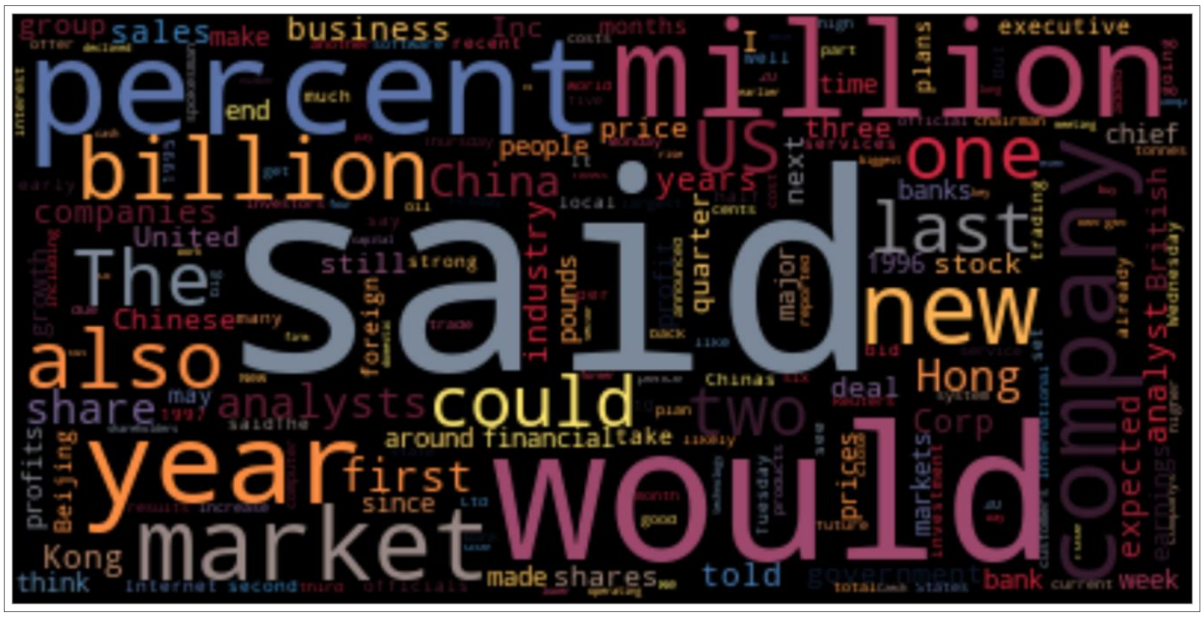

Figure 11 - Word Cloud for Clean Text

Compared to figure 5, this word cloud contains no stop words, and we see a set of new words.

Feature Engineering

Machine learning models understand numerical data only since we are dealing with a multi-class text classification problem [8]. We have to convert our data to numerical representation. The two techniques that have been used in this project are the Term Frequency Inverse Document Frequency (TFIDF) and Word Embeddings.

In the Exploratory Data Analysis part, we analyzed different features (figure 6 to figure 9) that could be considered as features but we have already explained why they will not be helpful in the learning process of our machine learning models.

Most importantly, our texts will be tokenized. Tokenization is the step where each sentence is divided into multiple tokens or words and then each word can act as a feature to our machine learning model [9].

We will now talk in detail about the TFIDF and Glove approaches that will be used in this project.

TFIDF

In TFIDF, the sentences are tokenized and then the words that appear in most documents are penalized and given a less score compared to words that are not frequent. The words that are not frequent (unique words) are given a high score. It is similar to one-hot encoding representation [10].

In the case of our dataset, our feature space contained 11738 unique words. This means that each review in our dataset contains 11738 features. This leads to a very high dimension space.

For this step, we used the Sklearn library that has a Tfidfvectorizer class. This class automatically converts our data and assigns the TFIDF score to each feature. It also tokenizes and converts everything to lowercase [11].

Glove Pre-trained Embeddings

The whole idea behind word embedding is that it is the type of representation that allows words with similar meanings to have similar representation. The words are mapped to one vector and their vector values are learned by using neural networks.

The advantage of this approach is that the similar words have similar representation. Another advantage of this approach is that we reduce the feature space. They are also known as feature reduction techniques and have been widely used [12].

For this project, we used a pre-trained embedding Glove which stands from Global Vectors. These pre-trained embeddings are trained on the Wikipedia dataset and are available in various dimensions. The 100 dimensions embeddings have been used.

Because Genism library deals with word2vec, we first had to convert our Glove embeddings to word2vec format. Once they were loaded, we mapped the vectors to our vocabulary and took the average. The advantage of this approach is that our data is now represented in less dimensions [13].

Modeling

For the machine learning models, we used Multinomial Naïve Bayes, Support Vector Machine, Random Forest, and Logistic Regression. Also, we used TFIDF and Glove Pretrained embeddings.

- Multinomial Naïve Bayes

Naïve Bayes is a fast algorithm that is based on probabilities. It also tends to not overfit on data [14]. Naïve Bayes treats each feature as independent and considered that there is no relation or connection between them.

- Support Vector Machine

Support Vector Machine is also a very popular algorithm for text classification. It works well on both linear and nonlinear problems. It create a line that separates the data into classes. By using the Kernel trick, the SVM classifier transforms the data into a higher dimension and this makes it suitable for text data along with its overfitting protection [14].

- Random forest

Random forest, which is a type of decision tree, is used to fix the problem of overfitting in decision trees [15]. This is done by taking random samples from the training set and then making small trees.

- Logistic regression

Logistic regression is a great starter algorithm, this is because it performs well on text data. Usually, it is used for binary classification but by using the one vs rest approach, it can work on multiclass classification as well [16].

Evaluation Results

TFIDF Results

TFIDF			
Model	Precision	Recall	Accuracy
Multinomial Naïve bayes	0.63	0.61	0.61
Support Vector Machine	0.69	0.51	0.51
Random Forest	0.69	0.51	0.53
Logistic Regression	0.65	0.61	0.61

Table 2 - Results for TFIDF on Clean Dataset

TFIDF (LEMMATIZED)			
Model	Precision	Recall	Accuracy
Multinomial Naïve bayes	0.64	0.62	0.62
Support Vector Machine	0.69	0.56	0.56
Random Forest	0.69	0.56	0.56
Logistic Regression	0.65	0.63	0.63

Table 3 - Results for TFIDF on Clean (Lemmatized) Dataset

TFIDF (STEMMED)			
Model	Precision	Recall	Accuracy
Multinomial Naïve bayes	0.64	0.62	0.62
Support Vector Machine	0.69	0.54	0.54
Random Forest	0.69	0.54	0.54
Logistic Regression	0.66	0.62	0.62

Table 4 - Results for TFIDF on Clean (Stemmed) Dataset

From the above tables, we can compare the results that we achieved after applying machine learning models on clean, lemmatized, and stemmed datasets.

In terms of precisions, we can see that the Support vector machine and random forest performed well on all three datasets. Precision tells us how many times our model was able to find the correct class.

In terms of recall, both Multinomial Naïve Bayes and logistic regression have performed well. Whereas SVM and Random forest achieved low recall scores. Surprisingly enough, when it came to accuracy, our logistic regression performed the best.

If we talk about accuracy, then we achieved 61, 63, 62 percentage for clean, lemmatized, and stemmed dataset.

Consequently, there was not much difference in the results in terms of using the clean dataset, stemmed and lemmatized.

The table below shows the best results achieved by model on each dataset.

Dataset Performance by Model			
Dataset	Precision	Recall	Accuracy
Cleaned	RF	MNB, LR	MNB, LR
Clean (Stemmed)	SVM, RF	MNB, LR	MNB, LR
Clean (Lemmatized)	SVM, RF	LR	LR

Table 5 - Dataset Performance by Model

Glove Embedding Results

Glove Pre Trained Embeddings 100 Dimensions			
Model	Precision	Recall	Accuracy
Support Vector Machine	0.02	0.02	0.02
Random Forest	0.02	0.02	0.02
Logistic Regression	0.02	0.02	0.02

Table 6 - Pre-Trained Embedding results

Our trained embeddings results were very bad, and this is because we did not have enough training data. When we use pre-trained word embedding models with ML models, we take the average of word vectors. Doing this will normally lead to information loss, and it makes it difficult for our model to learn the data. The loss of information is not recoverable.

Conclusion

The problem of author identification that we are dealing with in this project can be clearly stated in "Given Text, identify who is the author of that text" which considered a multiclass classification problem.

The dataset we are dealing with in this project contained 50 classes of authors for train and test data each of which has total of 50 observations.

We performed exploratory data analysis to have a good understanding of our data in order to make a better decision when choosing the input features. We also stemmed and lemmatized our data.

For the models, we applied Multinomial Naïve Bayes, Support Vector Machine, Random Forest, and Logistic Regression on the clean, lemmatized, and stemmed data along with the comparison of the performance. We found out that logistic regression performed the best when using the TFIDF representations on all the datasets.

We compared the performance of TFIDF with Glove Pre-trained embedding on our dataset. The results showed that the TFIDF performed better compared to the pre trained Glove embeddings on the traditional machine learning model.

When using pre-trained word embeddings on traditional machine learning algorithms, we were required to average word vectors which results in loss of information and this is the reason of having low scores with them.

Even though our dataset was balanced, but for 50 classes, the data was very less. Having more observations would made the models perform better, but at the same time it is very difficult.

Future Work

Traditional machine learning models do not perform well with word embeddings. In order to improve the performance, we can use deep learning models with pre-trained word embeddings. The reason of that is because pre-trained word embeddings are trained on a large corpus which covers a richer vocabulary than what our dataset contains. Apart from that we can use an RNN based neural network like Long Short-Term Memory (LSTM) to improve the performance.

Recent development in transformer-based model can be applied to achieve better results. Compared to RNN and LSTM's, the transformer-based model performs better.

Another advantage of these models is that they take an entire sequence of text rather than word by word and this makes these models faster.

Finally, we can try to improve the dataset by using different data augmentation techniques to increase the number of observations that we have for each class and this will result in having more data to train the model on.

Reference

[1] HaCohen-Kerner Y, Miller D, Yigal Y (2020) The influence of preprocessing on text classification using a bag-of-words representation. PLOS ONE 15(5): e0232525. https://doi.org/10.1371/journal.pone.0232525

[2] S. (2017, December 2). Simple Feature Engg Notebook - Spooky Author. Kaggle. https://www.kaggle.com/sudalairajkumar/simple-feature-engg-notebook-spooky-author

[3] Uysal, A.K. and Gunal, S., 2014. The impact of preprocessing on text classification. Information Processing & Management, 50(1), pp.104-112

[4] Malik, U. (n.d.). Removing Stop Words from Strings in Python. Stack Abuse. https://stackabuse.com/removing-stop-words-from-strings-in-python/

[5] Borad, A. (2020, June 15). NLP text pre-processing. Crazyblog. https://www.einfochips.com/blog/nlp-text-preprocessing/

[6] Sarkar, D., 2019. Text Analytics With Python. 2nd ed. APRESS.

[7] Balodi, T. (n.d.). What is Stemming and Lemmatization in NLP? | Analytics Steps. Analytics Steps. https://www.analyticssteps.com/blogs/what-stemming-and-lemmatization-nlp

[8] Kalaivani, K.S., Uma, S. and Kanimozhiselvi, C.S., 2020, March. A Review on Feature Extraction Techniques for Sentiment Classification. In 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC) (pp. 679-683). IEEE.

[9] Duque, T. (2020, November 9). How to turn Text into Features. To Wards Data Science. https://towardsdatascience.com/how-to-turn-text-into-features-478b57632e99

[10] Rezaeinia, S.M., Rahmani, R., Ghodsi, A. and Veisi, H., 2019. Sentiment analysis based on improved pre-trained word embeddings. Expert Systems with Applications, 117, pp.139-147

[11] Borcan, M. (2020, June 8). TF-IDF Explained And Python Sklearn Implementation. Towards Data Science. https://towardsdatascience.com/tf-idf-explained-and-python-sklearn-implementation-b020c5e83275

[12] Hartmann, J., Huppertz, J., Schamp, C. and Heitmann, M., 2019. Comparing automated text classification methods. International Journal of Research in Marketing, 36(1), pp.20- 38

[13] Brownlee, J. (2020, September 3). How to Develop Word Embeddings in Python with Gensim. Machine Learning Mastery. https://machinelearningmastery.com/develop-word-embeddings-python-gensim/

[14] Kowsari, Jafari Meimandi, Heidarysafa, Mendu, Barnes and Brown, 2019. Text Classification Algorithms: A Survey. Information, 10(4), p.150.

[15] Yiu, T. (2019). Understanding Random Forest. [online] Medium. Available at: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

[16] Ishaq, A., Umer, M., Mushtaq, M.F., Medaglia, C., Siddiqui, H.U.R., Mehmood, A. and Choi, G.S., 2020. Extensive hotel reviews classification using long short term memory. Journal of Ambient Intelligence and Humanized Computing, pp.1-11.