Authorship Attribution: NLP

David Pogrebitskiy, Jacob Ostapenko

Northeastern University, Boston, MA, USA

April 17, 2024

Introduction

Authorship attribution, the task of identifying the author of a given text or document, holds considerable importance across multiple domains. Whether discerning the origin of leaked information, establishing ownership of creative works, or maintaining academic integrity by detecting plagiarism, the ability to determine authorship is invaluable. In this project, we focused on evaluating the efficacy of different machine learning models in author classification. Our investigation not only sheds light on model performance but also contributes to the understanding of document embeddings and model hyperparameters.

Related Work

Authorship identification has been approached from various angles in the literature. Abbasi et al. (2022) employed ensemble learning techniques in their study of authorship identification. Their work utilized TFIDF (Term Frequency-Inverse Document Frequency) as a feature representation method, coupled with ensemble learning algorithms. By combining multiple base classifiers, ensemble learning enhances predictive performance and robustness. This approach demonstrated the effectiveness of ensemble methods in authorship identification tasks, providing valuable insights into alternative methodologies. This gave us the inspiration to try other features representations for the same task.

Methods

Data Acquisition and Pre-processing

We chose a dataset from Kaggle, named “All the news”; it was put together using news articles from 15 publishers (Thompson 2017). It contains 143,000 articles, belonging to 15,000 authors. Relevant information in one entry contains a publisher, the author, and the full text of an article.

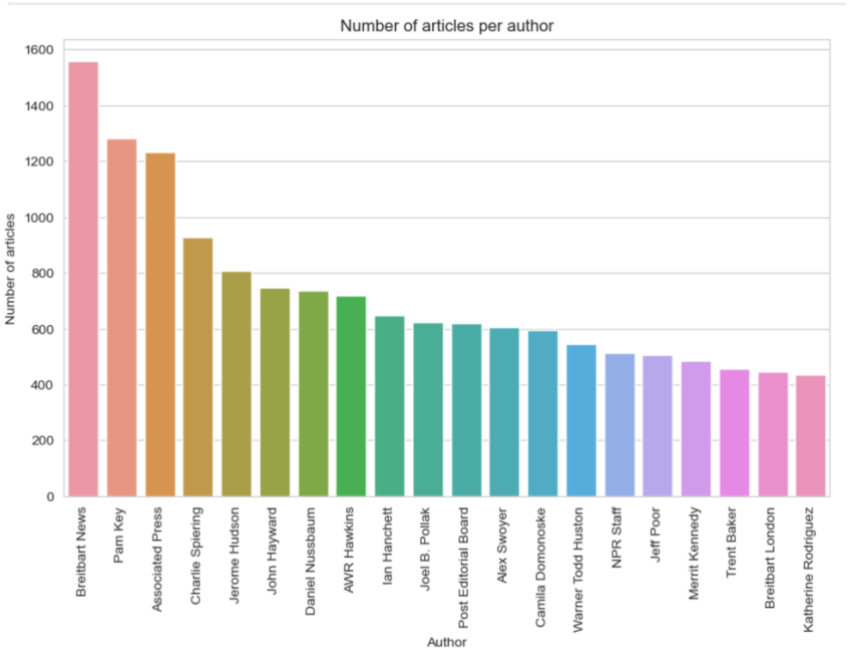
As with any corpus for a NLP task, the first step was to clean the data and retain only useful information. We removed data with missing authors or publishers and removed all stop words to remove any ‘information-less’ words or phrases. Next, the data was filtered to only include articles from the top 20 authors with the most articles. This was done because some authors have many articles in the original dataset while others do not – this led to a class imbalance that would make it very difficult to properly train and test models using this data. Furthermore, there are a total of 15,000 authors and we don’t have the computational power to

train a model to predict from a set of 15,000 authors. Figure 1 below depicts the top 20 authors and their corresponding number of articles.

Feature Engineering

For featurization, we explored two primary approaches: Doc2Vec and BERT. Doc2Vec, an extension of Word2Vec, enables the generation of dense embeddings for entire documents, leveraging neural network architectures. In contrast, BERT, equipped with a transformer architecture, offers powerful pre-trained tokenizer and embedding capabilities, particularly adept at capturing bi-directional context. We standardized features (embeddings) before training and testing each model. These approaches were chosen because they provide dense vectors that efficiently store numerical representations of each corpus. These dense vectors will then be able to be fed into any of the models we choose.

Figure 1



Histogram visualizing number of publications per author.

Model Evaluation

We evaluated 6 different models: Logistic Regression, Random Forest, SVM, Naïve Bayes, KNN, and Feed-Forward Neural Networks. The following section is focused on explaining each one of these in more detail. Each of the models' best hyperparameters can be seen in Table 1. In each of the models, we employed SMOTE during training to provide performance improvement and reduce the effect of class imbalance (Chawla et al. 2002).

Logistic Regression

A logistic regression model linearly combines features and their corresponding weights, multiplies them by an activation function, and outputs probabilities of each class using a SoftMax function. During training, there is an iterative process to update the weights of the features correctly – during each iteration, the loss function and its gradient are calculated, followed by the update of the feature weights. For training, a maximum iteration of 10000 was used and RandomizedSearchCV was used to fine-tune and find the best hyperparameters.

Random Forest

A random forest model is an ensemble-learning method based on decision trees; it outputs the most common class prediction across many individual decision trees. To train, the Random Forest selects a random sample of training data and features (bootstrap sampling) and then grows a decision tree for this data; this happens iteratively for an x number of decision trees which combine to make a random forest. For training, RandomizedSearchCV was used to fine-tune and find the best hyperparameters.

SVM

A SVM works by finding the best 'hyperplane' that divides the feature space into the different classes. They use different kernel functions to find patterns and map data into higher dimensional space where separation between classes is easier. This was optimized by maximizing the distance between the hyperplane and nearest data points. For training, RandomizedSearchCV was used to fine-tune and find the best hyperparameters (C value, kernel, degree, and gamma).

Naïve Bayes

A NB model use a probabilistic approach based on Bayes' Theorem and conditional independence between features. The model trains by estimating probabilities of features/embeddings given a class; during prediction, the model predicts probabilities of classes given a specific feature. We used a Gaussian Naïve Bayes model, under the assumption that the dataset follows a Gaussian distribution.

K-Nearest Neighbors

A KNN model doesn't require training – instead, it stores all available instances and then classifies new instances based on their similarities to the k-nearest neighbors in the 'training' set. We used RandomizedSearchCV was used to fine-tune and find the best hyperparameters (number of neighbors, weights, algorithm, leaf size, and p).

Feed-Forward Neural Network

A FFNN model contains layers of interconnected 'neurons', meaning inputs flow from layer to layer. Each layer is connected to another layer with a weighted connection and an activation function. FFNNs use backpropagation and gradient descent to update weights and minimize loss during training. We chose a 2 layer network with ReLU activation functions, and added 'batchnorm' to normalize the inputs to each layer, as well as a default dropout rate of 0.5 to

reduce overfitting. We started with one epoch and attempted increasing the number while observing the training and validation accuracy and loss.

Table 1

Model	Doc2Vec Best Hyperparameters	BERT Best Hyperparameters
LR	C: 100	C: 10
RF	n_estimators: 500, min_samples_split: 5, min_samples_leaf: 1, max_features: sqrt, max_depth: None	n_estimators: 500, min_samples_split: 5, min_samples_leaf: 1, max_features: sqrt, max_depth: None
SVM	kernel: rbf, gamma: scale, C: 100	kernel: poly, gamma: scale, degree: 4, C: 10
NB	No hyperparameters to tune	No hyperparameters to tune
KNN	weights: distance, p: 1, n_neighbors: 7, leaf_size: 10, algorithm: kd_tree	weights: distance, p: 1, n_neighbors: 7, leaf_size: 10, algorithm: kd_tree
FFNN	Hidden layer size=128, activation=relu, epochs=30, lr=1e-4, dropout=0.5	Hidden layer size=128, activation=relu, epochs=30, lr=1e-4, dropout=0.5

The best hyperparameters found for each of the models using the Doc2Vec and BERT features, respectively.

Results and Conclusions

We found that the Feed-Forward Neural Network performed the best, although marginally compared to the SVM model and the Logistic Regression model. All these 3 models had similar accuracy, F1, precision, and recall scores as can be seen in Table 2 and Figure 2 below. One important thing to note is that all the models performed significantly better on data that was tokenized and embedded using BERT, as opposed to Doc2Vec. This is likely because BERT uses a powerful transformer model that takes into context surrounding tokens, whereas Doc2Vec is a Neural-Network based model. Although both very powerful, BERT is also trained on significantly more data than Doc2Vec and can be expected to produce more complicated embeddings which capture info at a higher ability.

Error Analysis

After close examination of the errors produced by the Feed-Forward Neural Network trained on the BERT embeddings, we concluded that some of the author categories didn't represent individuals, but rather larger corporations that produce publications written by various unnamed authors. For example, articles written by Breitbart News and Associated Press were the two most misclassified classes in the testing set. We hypothesize that because many of the articles 'written' by Breitbart News or Associated Press were written by different people, the model couldn't identify a specific linguistic style or writing pattern that could be associated with the label.

Future Work

Going forward, we would remove these larger names from the dataset and focus on only individuals. We would also like to diversify the data being used. Authorship attribution could be

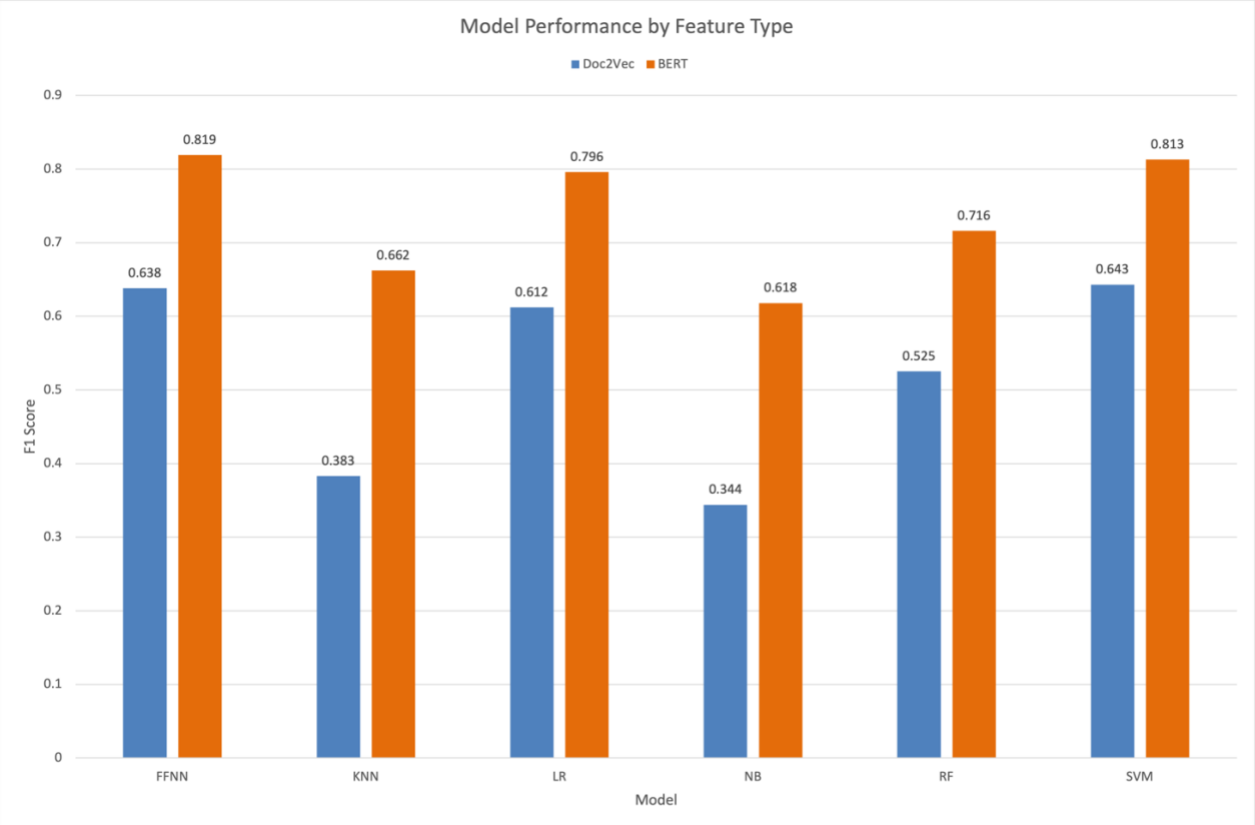
useful for detecting plagiarism or verifying authorship of legal documents/copyrights. In this project we looked at a dataset of news articles – in the future, we would like to extend to other literary styles, such as academic papers and legal documents. In addition to the current models we've experimented with, we could try deeper, more intricate neural network architectures like recurrent, convolution, attention mechanisms, and other kinds of Transformer-based Architectures. Past this, we would also like to explore text generation for specific literary styles mentioned earlier. Author attribution can be useful in areas mentioned above, but text generation also can be useful in many other areas, such as LLMs.

Table 2

Model	**Doc2Vec**				**BERT**			
	Accuracy	F1	Precision	Recall	Accuracy	F1	Precision	Recall
FFNN	0.640	0.638	0.661	0.640	0.816	0.819	0.827	0.816
KNN	0.376	0.383	0.455	0.376	0.660	0.662	0.705	0.660
LR	0.611	0.612	0.615	0.611	0.795	0.796	0.799	0.795
NB	0.346	0.344	0.399	0.346	0.613	0.618	0.635	0.613
RF	0.540	0.525	0.535	0.540	0.714	0.716	0.726	0.714
SVM	0.645	0.643	0.650	0.645	0.813	0.813	0.815	0.813

The model performance for both types on both types of training features.

Figure 2



This figure shows the F1 scores of each of the model on Doc2Vec and BERT features, respectively.

References

Abbasi, A., Javed, A.R., Iqbal, F. *et al.* Authorship identification using ensemble learning. *Sci Rep* **12**, 9537 (2022). https://doi.org/10.1038/s41598-022-13690-4

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 321–357. https://arxiv.org/abs/1106.1813

Thompson, A. All the news: 143,000 articles from 15 american publications. https://www.kaggle.com/snapcrack/all-the-news (2017).