Authorship Attribution with GatorCAAT using Machine Learning Techniques

Rahul Alapati
Department of Computer Science and Software Engineering
Auburn University
rza0037@auburn.edu

Sutanu Bhattacharya
Department of Computer Science and Software Engineering
Auburn University
szb0134@auburn.edu

***Abstract*— Authorship Attribution is the science of identifying the author from the characteristics of articles written by that author. It can be applied to tasks such as identifying anonymous author, detecting plagiarism or finding ghost writer. In this paper, we explore the use of Machine Learning Techniques for the identification of the author of a text. Three machine learning techniques, namely K-Nearest Neighbor, Distance Weighted K-Nearest Neighbor and General Regression Neural Network, were developed for authorship attribution over the CASIS-25 and SEC sports writers' datasets. We analyze the performance of these basic instance based methods and also discuss strategies to improve their accuracies.**

Keywords—Authorship Attribution, GatorCAAT, kNN, Distance Weighted kNN, GRNN, CASIS-25, SEC Sports Writers

I. INTRODUCTION

Every author usually has a unique writing style, which is to some extent irrelevant with the writing genre. The process of identifying the author from a group of candidates, given a collection of his/her writing samples is Authorship Attribution. The importance of authorship attribution lies on its wide range of applications ranging from plagiarism detection to resolving historical or disputed authorship [1]. Naturally, authorship attribution is a very active and lucrative field of machine learning. It is both a difficult and somewhat open-ended task, with numerous approaches and success rates [2]. In our work, two labelled datasets, namely CASIS-25 and SEC Sports Writers are used to train and test the instance based machine learning methods.

Instance based learning [21] is a family of learning algorithms that, instead of performing explicit generalization, compares new test instances with the training instances, which have been stored in memory. They do not create a model. Instead, they use the training set each time a new test instance is to be classified. One major advantage of instance based learning is its ability to adapt to previously unseen data. In our work, we have implemented three simple, but effective instance based methods, namely K-Nearest Neighbor [3], Distance Weighted K-Nearest Neighbor [4] and General Regression Neural Network [5] to perform authorship attribution.

The organization of the paper is as follows: In section 2, we elaborate the approach of our work, including the datasets and feature representations we used, as well as the instance based methods, we developed for authorship attribution. In section 3, the experiments we did, and the corresponding results are shown. Section 4 ends the paper with discussion on the breakdown of work among the two authors.

II. METHODOLOGY

In this section, our approaches for authorship attribution are elaborated in detail in the following order: datasets, feature extraction and instance based learning algorithms.

A. Datasets

In our work, two different datasets are exploited to train and test our model. The two datasets are CASIS-25 and SEC Sports Writers. The CASIS-25 dataset is composed of 25 blog entries. Each blog entry is broken into 4 writing samples (4 writing samples per author). It is a static dataset with fixed number of writing samples. Whereas, the SEC Sports Writers dataset is a dynamic dataset where writing samples are added to existing dataset each week. As a part of the data collection process, we copy pasted 49 Football Game Recaps of Florida Gators for the first 7 weeks, from 9 daily newspapers [6-15] within the state of Florida written by 15 different sports writers.

Some of the new challenges faced during the data collection process are:

1. The game recap of the Week 4 Football game between Florida and Kentucky was not published in the Palm Beach Post [11] for some unknown reasons. Hence, we acquired the game recap from CBS Sports [13].
2. The game recap of the Week 6 Football game between Florida and LSU was written by the same author in Local 10 [12] and St. Augustine Record [6]. Hence, we replaced the Local 10 game recap with the one in CBS Sports [13].
3. The game recap of the Week 7 Football game between Florida and Vanderbilt was written by the same author in Tampa Bay Times [10], Local-10 [12] and Palm Beach Post [11]. Hence, we replaced them with the game recaps acquired from the Tennessean [14] and Alligator Army [15].

The above mentioned challenges have led to a mismatch in the number of writing samples per author in the SEC Sports Writers Dataset.

The use of both static and dynamic datasets for authorship attribution using machine learning techniques, gives us a chance to analyze the changes in the decision boundaries of different algorithms and to study their effects on the accuracies over dynamic datasets when compared to static datasets.

B. Feature Extraction

After acquiring the game recaps, we generate the raw and normalized feature vectors using the Feature Extractor developed as a part of Assignment-1 [16]. In addition to the

raw and normalized feature vectors, we apply Tf-idf term weighting to our raw feature vectors, in order to prevent the very frequent characters in our character unigram to shadow the frequencies of rarer yet interesting characters. Tf-idf means term-frequency times inverse document-frequency. We use the scikit learn python library to apply the Tf-idf term weighting to our raw feature vectors [17].

C. Instance Based Learning Methods

In our work, we have implemented three simple, but effective instance based methods, namely K-Nearest Neighbor, Distance Weighted K-Nearest Neighbor and General Regression Neural Network to perform authorship attribution.

i. K-Nearest Neighbor

K-Nearest Neighbor (KNN) has been one of the most well-known algorithms in pattern classification, since it was first introduced. This rule simply retains the entire training set during learning and assigns to each query a class represented by the majority label of its k-nearest neighbors in the training set. KNN has several main advantages: simplicity, effectiveness, intuitiveness and competitive classification performance in many domains [18].

Let $\{(t_1,d_1), (t_2,d_2), \dots , (t_n,d_n)\}$ be the training set where t_n represents the training instance n and d_n represents its class label. In KNN, a query instance t_q is classified into class d_q by the following:

Step 1. Select k nearest neighbor training instances from t_q with Euclidean distance measure. Let the neighborhood of k training samples be represented by $\{(t_{c[1]}, d_{c[1]}), (t_{c[2]}, d_{c[2]}), \dots , (t_{c[k]}, d_{c[k]})\}$. Where c is an array of indexes of the closest distances to q using a distance function dist(q,i), that returns the Euclidean distance between t_q and t_i.

Step 2. Classify t_q into class d_q, which is set to most common d_i in the neighborhood.

The accuracy of the KNN is relatively low when compared to other classification algorithms for several reasons. One of them is that the determination of new data classes is based on a simple vote majority system where the majority vote system ignores the closeness between data. Furthermore, there will be a possibility of double majority class caused by the class determination system for new data based on the vote majority [18].

ii. Local Mean based K-Nearest Neighbor:

In our work, we propose a replacement to the vote majority system in KNN using the local mean based K-Nearest Neighbor (LMKNN) method [19]. This method is simple, effective and resilient nonparametric classification. This method has been proven to improve classification performance and also reduce the effect of existing outliers. The process of LMKNN can be described as follows:

Step 1. Determination of the k value

Step 2. Compute the distance between the query instance and all the training instances using the Euclidean distance measure.

Step 3. Sort the distances in ascending order as much as k for each data class.

Step 4. Calculate the local mean vector of each class.

Step 5. Compute the distance between the query instance and the local mean vectors of all the classes using the Euclidean distance measure.

Step 6. Classify the query instance with the label of the local mean vector which is closest to the query instance.

The LMKNN classification is equal to 1-Nearest Neighbor if the value of k=1. The value of k is different from that in the KNN, in KNN the value of k is the number of nearest neighbors from all the training data, whereas in LMKNN the value of k is the number of nearest neighbors from each class in the training data [19].

In the class determination of the query instance, LMKNN uses the closest distance to each local mean vector of each data class, which is considered effective in overcoming the negative effects of outliers [19].

iii. Distance Weighted K-Nearest Neighbor

One of the major reasons for the relatively low accuracy of KNN algorithm is that the determination of new data classes is based on a simple vote majority system, where the closeness between the data is totally ignored, which is unacceptable when the distance of each nearest neighbor differs greatly against the distance of the test data. Distance Weighted K-Nearest Neighbor (WKNN) has been proposed to overcome this problem [4]. This method specifies a new data class based on the weight value obtained from the distance between data. This weighting method increases the performance of KNN because it can reduce the influence of outliers and also influence of the distribution of the unbalanced datasets.

Let $\{(t_1,d_1), (t_2,d_2), \dots , (t_n,d_n)\}$ be the training set where t_n represents the training instance n and d_n represents its class label. In WKNN, a query instance t_q is classified into class d_q by the following:

Step 1. Select k nearest neighbor training instances from t_q with Euclidean distance measure. Let the neighborhood of k training samples be represented by $\{(t_{c[1]}, d_{c[1]}), (t_{c[2]}, d_{c[2]}), \dots , (t_{c[k]}, d_{c[k]})\}$. Where c is an array of indexes of the closest distances to q using a distance function dist(q,i), that returns the Euclidean distance between t_q and t_i.

Step 2. Calculate the weights from the distances using the following equation:

$$w_i = \text{dist}(q,c[i])^{-b}$$

$$w_i = 1, \text{ when } \text{dist}(q,c[i])^{-b} = 0$$

where w_i is the weight of nearest neighbor i

dist(q,c[i]) is the distance between the nearest neighbor i and query instance q.

b value is optimized to achieve higher accuracies.

Step 3. Classify t_q into d_q, which is set to the most weighted d_i in the neighborhood.

WKNN is called as Local Method when k < n and as global method when k = n.

WKNN increases the accuracy of KNN by weighting the close neighbors more heavily according to their distances from the query instance [4].

iv. Distance Weighted K-Nearest Neighbor using Local Means

Due to the curse of dimensionality and the existing outliers in small datasets, the classification accuracy of WKNN leads to a dramatic degradation. We propose the following algorithm to improve the accuracy of WKNN [19]:

Step 1. Determination of k value.

Step 2. Compute the distance between the query instance and all the training instances using the Euclidean distance measure.

Step 3. Sort the distances in ascending order as much as k for each data class.

Step 4. Calculate the weights from the distances using the following equation:

$$w_i = dist(q,c[i])^{-b}$$

$$w_i = 1, \text{ when } dist(q,c[i])^{-b} = 0$$

where w_i is the weight of nearest neighbor i

dist(q,c[i]) is the distance between the nearest neighbor i and query instance q.

b value is optimized to achieve higher accuracies.

Step 5. Calculate the average weight for each class.

Step 6. Classify t_q into d_q, which is set to the class with the highest average weight.

Generally, the classification performance of KNN results in the estimate of the conditional class probabilities from training set in a local region of data space, which contains k nearest neighbors from the query instance. This estimate is heavily affected by the sensitivity of the selection of the neighborhood size k [19].

v. Distance Weighted K-Nearest Neighbor using the dual distance weighted function

To reduce the effect of sensitivity of the selection of the neighborhood size k on the KNN-based classifiers, we propose the Distance Weighted K-Nearest Neighbor algorithm with dual distance weighted function (DWKNN) [18]. The proposed DWKNN algorithm can be described as follows:

Step 1. Compute the distances of all the training instances from the query instance using the Euclidean Distance measure.

Step 2. Sort the distances in ascending order.

Step 3. Select the closest k-nearest neighbors.

Step 4. Calculate the dual weights of k-nearest neighbors using the following:

for i = 1 to k do

if $dist(t_q, t_k^{NN}) \neq dist(t_q, t_1^{NN})$ then

$$w_i = \frac{dist(t_q,t_k^{NN}) - dist(t_q,t_i^{NN})}{dist(t_q,t_k^{NN}) - dist(t_q,t_1^{NN})} * \frac{dist(t_q,t_k^{NN}) + dist(t_q,t_i^{NN})}{dist(t_q,t_k^{NN}) + dist(t_q,t_1^{NN})}$$

else

$w_i = 1$

where $dist(t_q, t_k^{NN})$ is the distance between the query instance and its k^{th} nearest neighbor.

Step 5. Classify t_q into d_q, which is set to the majority weighted class d_i.

vi. General Regression Neural Network

General Regression Neural Networks (GRNN) [5] are global methods that consist of a hidden layer of Gaussian neurons (one neuron for each training instance t_i), a set of weights w_i , where $w_i = d_i$ the desired output vector and a set of standard deviations σ_i for each training instance i.

The query instance t_q can be classified into d_q using the following equation:

$$d_q = f(t_q) = (\Sigma hf_i(t_q,t_i)d_i) / \Sigma hf_i(t_q,t_i)$$

where hidden function hf_i can be calculated as follows:

$$hf_i(t_q,t_i) = \exp(- (||t_q - t_i||^2)/2\sigma_i^2)$$

The value of the standard deviation σ can be optimized to obtained higher accuracies.

When the d_i is a desired output vector then the resulting network is called a multiple output GRNN.

The hidden function values are calculated using Gaussian kernels, whereas the weights in the Shepard's method (k = n) are calculated based on the distance function used to determine the k-nearest neighbors.

Generally, the desired output vectors are identity matrices corresponding to the authors of the writing samples. But we can evolve the desired output vectors using evolutionary computation and genetic algorithms to improve the accuracy of our basic GRNN.

vii. Evolving Desired Output Vectors for GRNN using Genetic Algorithm

To improve the accuracy of our basic GRNN, we propose the following Genetic Algorithm [20]:

Step 1. Initialize the population using random gaussian distribution consisting of candidate solutions for the desired output vectors.

Step 2. Evaluate the fitness of the all the individuals in the population with the classification error of the GRNN as its evaluation metric.

Step 3. Select two parents using tournament selection.

Step 4. Generate two children from the two parents using blend crossover.

Step 5. Apply Gaussian Mutation on the two children to introduce some diversity into the population.

Step 6. Evaluate the fitness of the two children

Step 7. Replace the two worst fit individuals in the population with the two children. This type of GA is called as Steady State GA.

Step 8. Repeat from Step 3 to Step 7 for n generations until a stopping criterion is met.

The use of an optimized sigma value and evolved desired output vectors in GRNN, improves its classification performance relatively.

III. Experiments and Results

1. *Tf-idf transformation on raw feature vectors*

Tf-idf means term-frequency times inverse document-frequency. We apply Tf-idf transformation on our raw feature vectors to prevent the very frequent characters in our character unigram to shadow the frequencies of rarer yet interesting characters [17]. In this experiment, we compare the accuracies of the instance based learning methods on the raw, normalized and transformed feature vectors of CASIS-25 dataset.

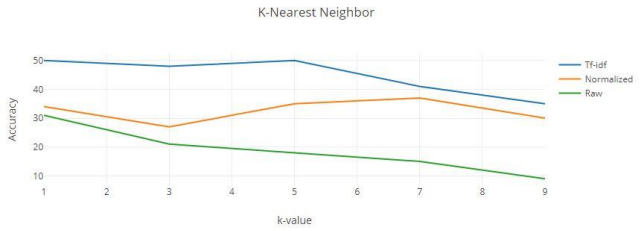

Figure 1: Accuracy Analysis for KNN over transformed, normalized and raw features of CASIS-25 Dataset

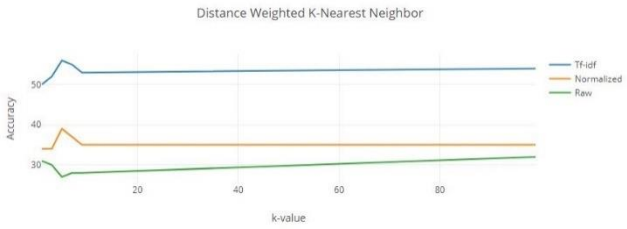

Figure 2: Accuracy Analysis for WKNN over transformed, normalized and raw features of CASIS-25 Dataset

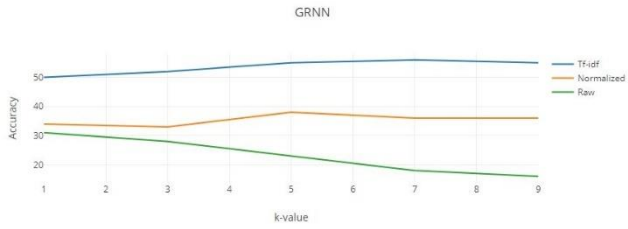

Figure 3: Accuracy Analysis for GRNN over transformed, normalized and raw features of CASIS-25 Dataset

Figures 1, 2 and 3 show the effect of Tf-idf transformation on the accuracies of KNN, Distance Weighted KNN and GRNN algorithms, respectively. It can be noted that Tf-idf transformation results in the highest accuracy, followed by the normalized features and the raw features when observed over k-values 1,3,5,7 and 9. These results demonstrate the importance of Tf-idf transformation in reducing the dominance of frequent characters over the rare characters. Hence, the transformed features have been used for authorship attribution in all the remaining experiments.

2. *Improving the accuracy of KNN using Local Mean Vectors*

We investigated the effect of simple majority based voting system on the classification performance of KNN [3] over CASIS-25 and SEC Sports Writers Datasets during Authorship Attribution. We observed that the accuracy of baseline KNN algorithm decreases as the k-value increases on the CASIS-25 dataset. The highest accuracy achieved by the basic KNN on this dataset is 50% for k=1 and 5. While, in case of the SEC Sports Writers Dataset, the accuracy of the baseline KNN algorithm decreases and then constantly increases as the k-value increases. The highest accuracy achieved on this dataset is 45% for k=1.

To reduce the effect of the simple majority based voting and thereby improve the accuracy of basic KNN, we propose the LMKNN method [19]. With LMKNN, we observed that the accuracy increases as the k-value increases and the highest accuracy achieved during this process was 63% for k=5,7 and 9 on CASIS-25 dataset and 45% for k=1 on SEC Sports Writers dataset.

Figure 4: Comparison between the classification performances of KNN and LMKNN over the CASIS-25 Dataset

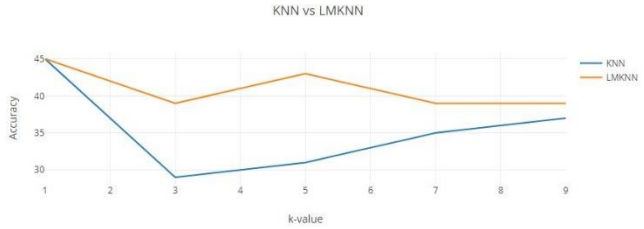

Figure 5: Comparison between the classification performances of KNN and LMKNN over the SEC Sports Writers Dataset

From figures 4 and 5, we can observe that on an average there is an improvement of 20% in the classification performance of KNN when Local Mean Vectors are used. Also, it can be noted that the performance of LMKNN algorithm is equal to that of 1-NN algorithm when k=1, as the local mean vector is equal to the closest neighbor itself when k=1.

3. *Improving the accuracy of Distance-Weighted KNN using Local Mean Vectors and Dual Distance Weighted Function*

Firstly, we optimize the value of b, which is involved in the weight calculation of the WKNN [4], by varying the b value from 0.5 to 10 with a step size of 0.5. We observed that, the b value of 5 for local method and the b value of 10 for Shepard's method result in the highest accuracies on CASIS-25 dataset. Whereas, the b value of 10 for local method and Shepard's method results in the highest accuracies on SEC Sports Writers dataset.

To investigate the effect of curse of dimensionality and outliers on WKNN algorithm, we compare the classification

performance of WKNN and WKNN with Local Mean Vectors [19] on CASIS-25 and SEC Sports Writers Datasets.

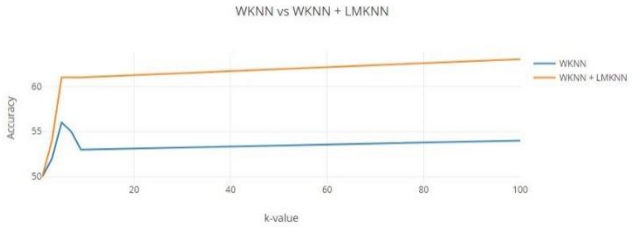

Figure 6: Comparison between the classification performances of WKNN and WKNN with local means over the CASIS-25 Dataset

From figure 6, we can observe that, the classification performance of WKNN is increased by 7% on an average for k =1, 3, 5, 7, 9, 100 on the CASIS-25 dataset, by reducing the effect of outliers using the local means.

Figure 7: Comparison between the classification performances of WKNN, WKNN with local means and DWKNN over the SEC Sports Writers Dataset

From figure 7, we can observe that the local means in case of SEC Sports Writers Dataset decreases the performance of the WKNN. This is due to the small size of the dataset i.e. 49 writing samples and the less number of writing samples per author i.e. 8 authors out of 15 have only one or two writing samples which thereby aggravate the effect of outliers. Also, as the writing samples are game recaps, there is an increased usage of the same characters in all the writing samples which effects the distance calculations and means in the algorithms, thereby effecting their accuracy. Hence, to overcome these, we apply a dual weighted function in WKNN named as DWKNN [18], which minimizes the above mentioned effects and increases the performance of WKNN by 6% on an average.

4. Improving the accuracy of GRNN by optimizing sigma and evolving the desired output vectors

Firstly, we optimize the value of sigma, by varying it from 0.05 to d_{max}, where d_{max} is the maximum distance between training instances, with a step size of 0.005. From Figure 8, we can observe that for CASIS-25 dataset, the sigma value of 0.2 results in highest accuracy of 54%, whereas a sigma value of 0.1 results in highest accuracy of 47% on SEC Sports Writers Dataset.

Figure 8: Optimize sigma values for CASIS-25 and SEC Sports Writers Dataset

Using sigma value, we evolve the desired output vectors using Steady State Genetic Algorithm [20] and obtain the best desired output vector. While using blend crossover BLX-0.5 for generating children, we obtain an improved accuracy of 61% over CASIS-25 dataset and 53% over the SEC Sports Writers dataset. Whereas, while using blend crossover BLX-0.0 for generating children, we obtain an improved accuracy of 60% over CASIS-25 dataset and 49% over the SEC Sports Writers Dataset.

The following figure shows the performance of GRNN over CASIS-25 and SEC Sports Writers Datasets for different k-values.

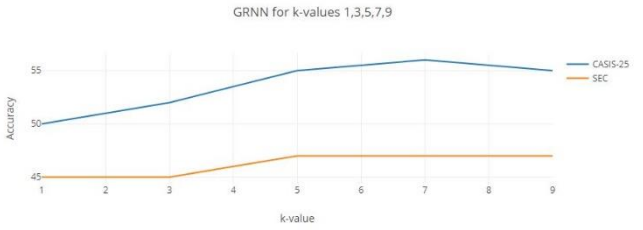

Figure 9: Performance of GRNN for different k-values over CASIS-25 and the SEC Sports Writers Datasets

IV. Breakdown of the Work

Rahul was responsible for the collection of game recaps from newspapers. He was also responsible for implementing the baseline versions of Distance Weighted K-Nearest Neighbor (WKNN) and GRNN. He also proposed improvements such as WKNN with Local Means and WKNN with Dual Distance Weighted Function for WKNN. He performed the optimization of sigma and evolved the desired output vectors to improve the performance of GRNN.

Sutanu was responsible for implementing the baseline version of KNN. He also implemented the improvised version of KNN using Local Mean Vectors.

References

[1] Qian, Chen, Ting He and Rao Zhang. “Deep Learning based Authorship Identification.” Machine Learning Final Projects (2017).

[2] Stanko, Sean, Devin Lu, and Irving Hsu. "Whose book is it anyway? using machine learning to identify the author of unknown texts." Machine Learning Final Projects (2013).

[3] T. M. Cover, P. Hart, "The nearest neighbor decision rule", IEEE Trans. Inform. Theory, vol. IT-13, pp. 21-27, Jan. 1967.

[4] Dudani, Sahibsingh A. "The distance-weighted k-nearest-neighbor rule." IEEE Transactions on Systems, Man, and Cybernetics 4 (1976): 325-327.

[5] Specht, Donald F. "A general regression neural network." IEEE transactions on neural networks 2.6 (1991): 568-576.

[6] http://www.staugustine.com/.

[7] http://www.ocala.com/.

[8] https://www.alligator.org/.
[9] https://www.orlandosentinel.com/.
[10] http://www.tampabay.com/.
[11] https://www.palmbeachpost.com/.
[12] https://www.local10.com/.
[13] https://www.cbssports.com/.
[14] https://www.tennessean.com/.
[15] https://www.alligatorarmy.com/.
[16] R Alapati, S Bhattacharya, S Dsouza. “Authorship Attribution with GatorCAAT: Data Collection and Feature Extraction”.
[17] http://scikitlearn.org/.
[18] Gou, Jianping, et al. "A new distance-weighted k-nearest neighbor classifier." J. Inf. Comput. Sci 9.6 (2012): 1429-1436.
[19] K U Syaliman et al 2018 J. Phys.: Conf. Ser. 978 012047.
[20] G. Dozier, A. Homaifar, E. Tunstel, and D. Battle, "An Introduction to Evolutionary Computation" (Chapter 17), *Intelligent Control Systems Using Soft Computing Methodologies.*
[21] Tom Mitchell, “Machine Learning”, McGraw-Hill.