Robust Authorship Attribution System

Bhargav Joshi[1], Armin Khayyer[1] and Ye Wang [1]

***Abstract*— An authorship attribution system learns about different authors' writing styles using the extracted features of each author's writing style. The system is made robust by generating features to be trained using multiple feature extraction techniques. The final set of extracted feature is then evolved using genetic evolutionary computation approach for feature extraction (GEFeS) to find the best set of features. To make the system resistant to attacks, different machine learners are trained using the evolved feature set. The system generates final prediction by choosing the most commonly predicted label by most accurately trained machine learners. The system resists different attacks since it is difficult for an adversary to figure out which machine learner would predict the final prediction.**

I. Introduction

Authorship attribution systems (AAS) are machine learning inspired prediction mechanisms to predict the author of a text. The system contains machine learning models such as multi-layer percepetron (MLP) network, linear support vector machines (LSVM), radial basis functions support vector machines (RBFSVM). The goal is to learn how different authors write their texts. When it comes to developing a model about writing styles, feature extraction techniques are used to process written texts by each authors in order to be able to fit a machine learner. Some of the feature extraction techniques include unigram extraction, stylometry extraction, bag of words extraction, n-gram extraction. The details about these feature extractors are further explained in Section II.

The main goal of this project is to make the authorship attribution system more robust and resistant to adversarial attacks. Making the system more robust means the system should be able to make correct predictions even if the provided test samples are tempered with. This could be achieved if machine learning technique used in the system learns thoroughly about an author's writing style. Training a AAS depends on multiple factors, some of the most important factors are: the the number of instances in the training set, the features of each instance in the training set, and to name but a few. The training set of our AAS was created using multiple feature extraction techniques, such as unigrams, stylometry, bag of words, and finally n-grams feature extraction technique, so that the system can define the writing style of an author in more unique way. This creates a diverse training set. The machine learner should learn about the writing style from extracted features rather than memorizing the whole training set. Preventing the overfitting also results into more robust system. A GEFeS method is used to find a mask for the training dataset to distinguish the features which results in a more accurate attribution system. This way the attackers cannot identify which features have been used by the classifiers for the training and testing process.

The system is more resistant to adversarial attacks when the adversary cannot identify the machine learning method used in the system. The authorship attribuition system proposed here is not made with only one machine learning model. It consists of neural MLP, LSVM and RBFSVM. All of these methods share some sort of statistical equivalence but they are trained very differently. Our authorship attribution system predicts the author using each machine learning model. The most common prediction is selected unless all machine learners predict differently. In such case, the prediction generated by the most accurate model is chosen.

The remaining of this paper is structured as follows: Section II explains feature extraction methods, baseline methods, and finally the GEFeS method, Section III provides the experiment setting, Section IV provides the results achieved using the GEFeS method and also answers some questions about the baseline methods, finally Section V conclude the paper and our achieved findings.

II. Methodology

A. Uni(N)-Character-Gram Feature Extraction

Uni-Character-Gram of texts are extensively used in text mining and natural language processing tasks[1]. They are essentially the individual character in a sentence. For example, for the sentence "The cow jumps over the moon.", the Uni-Character-Gram would be: {T, h, e, c, o, w, j, u, m, p, s, o, v, e, r, t, h, e, m, o, o, n,.}. N-Character-Gram is extended version of Uni-Character. They are basically a set of co-occuring characters within a given window and when computing the n-grams you typically move one character forward. For example, for the sentence "The cow jumps over the moon.", the 2-Character-Gram would be: {Th, he, ec, co, ow, wj, ju, um, mp, ps, so, ov, ve, er, rt, th, he, em, mo, oo, on,n.}. N-grams are used for a variety of different task. For example, when developing a language model, n-grams are used to develop not just unigram models but also bigram and trigram models.

In this paper, we use both Uni-Character-Gram and 3-Character-Gram to extract features. We first build a unigram character and 3-gram character lookup table, and then the frequency of element of look up table for the given sentence will be calculated. Finally, we use frequency of all the characters as the extracted feature vectors.

[1]Department of Computer Science and Software Engineering Auburn University Auburn, USA

B. Bag of Words Feature Extraction

The bag-of-words model is a simplifying representation used in natural language processing and information retrieval (IR)[2]. In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. For example, given two simple text documents: {John likes to watch movies. Mary likes movies too.}, {Mary also likes to watch football games.}. Based on these two text documents, a look up is constructed as follows for two documents: {"John", "likes", "to", "watch", "movies", "Mary", "likes", "movies", "too", "also" "football", "game"}. The frequency of word for each docment will be computed. The bag of words features of two documents can be represented as two vectors $[1,2,1,1,2,1,1,0,0,0],[1,1,1,1,0,0,0,1,1,1]$.

The Bag-of-words model is an orderless document representation - only the counts of words matter, and it also can be extend to N-gram model. In this paper, we only focus on Bag-of-words feature extraction, and build the look up table for all documents, which contrains 6118 words.

C. Stylometry Feature Extraction

Stylometry is the application of the study of linguistic style, usually to written language, and it is often used to attribute authorship to anonymous or disputed documents [3]. The use of stylometric features relies on the theory of style by Sanders: "Style is the result of choices made by an author within a context from a range of possibilities offered by the language system". Stylometry expects that stylistic variations depend on the author preferences and competence, the genre, and the communicative context [4]. To build the stylometry feature, we first select 320 topic-free functional words, such as "a" , "about" and "above", and calculate the frequency of functional words. Besides functional words, we also introduce some basic statistics information of text to the stylometry feature, for example the length of sentence and text, number of digit number, and the number of words per sentence. In this paper, our stylometry feature will contain 428 features.

D. Linear SVMs

Support vector machines (SVMs, also support-vector networks[5]) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis[5]. Given a training set of N data points y_k, x_k, where $x_k \in \mathbb{R}^n$ is the kth input pattern and $y_k \in \mathbb{R}$ is the kth output pattern, the support vector method approach aims at constructing a classifier of the form:

$$y(x) = \text{sign}(\sum_{k=1}^{N} \alpha_k y_k \phi(x, x_k) + b) \tag{1}$$

where α_k are positive real constants and b is a real constant. For linear SVM, the $\phi(\cdot,\cdot)$ should be

$$\phi(x, x_k) = x^T \tag{2}$$

The classifier is constructed as follows. One assumes that

$$w^T \phi(x_k) + b \geq 1, \text{if} \quad y_k = +1 \tag{3}$$

$$w^T \phi(x_k) + b \leq -1, \text{if} \quad y_k = -1 \tag{4}$$

which is equivalent to

$$y_k[w^T \phi(x_k) + b] \geq 1, \quad y_k = 1, \cdots N, \tag{5}$$

where $\phi(\cdot)$ is a linear function.

E. Radial Basis Function SVMs

RBF kernel function is a universal kernel functions, and it can be applied to any of the distribution of the samples through the choice of parameters. It has been more and more used in the nonlinear mapping of support vector machine more and more. RBF kernel function expression is:

$$\phi(x, x_k) = \exp(\frac{-||x - x_k||^2}{2\sigma^2}) \tag{6}$$

By introducing radial basis kernel function, we can easily solve the linearly inseparable problem. The kernel is a way of computing the dot product of two vectors x and x_k in some (very high dimensional) feature space, which is why kernel functions are sometimes called generalized dot product.

F. Multi-Layer Percepetron Neural Network

Percepetron neurons are a type of neurons that support supervised learning of binary classifiers using percepetron algorithm developed by Frank Rosenblatt [6]. In the modern sense, the perceptron is an algorithm for learning a binary classifier called a threshold function: a function that maps its input x (a real-valued vector) to an output value $f(x)$ (a single binary value) [7]:

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where w is a vector of real-valued weights, $w \cdot x$ is the dot product $\sum_{i=1}^{m} w_i x_i$, where m is the number of inputs to the perceptron, and b is the bias.

Multi-Layer Percepetron (MLP) network is a neural network consisted of layered architecture of percepetron neurons. It contains many perceptrons that are organized into layers. Unlike the early age perceptron discussed above, percepetron neurons in MLP can have different activation functions such as sigmoid functions, linear functions, ,softmax functions, rectifier linear units [8][9]. Since MLPs are fully connected, each node in one layer connects with a certain weight w_{ij} to every node in the following layer.

The degree of error in an output node j at the nth data point can be denoted as,

$$e_j(n) = d_j(n) - y_j(n) \tag{8}$$

where, d_j is the desired output value and y_j is the value produced by the percepetron. The weights at each node are

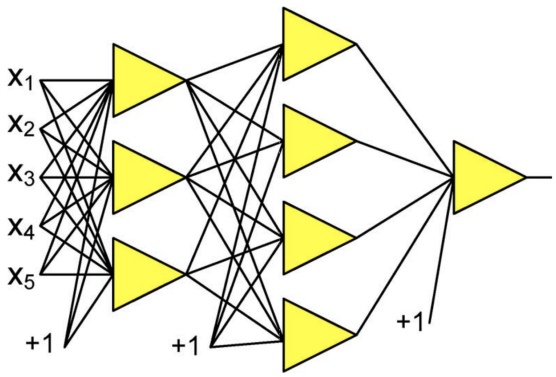

Fig. 1. MLP network with 8 neurons and two hidden layers [10]

targeted be adjusted such that the total error in the entire output is minimized. This error function is computed as,

$$\mathscr{E}(n) = \frac{1}{2}\sum_{j} e_j^2(n) \tag{9}$$

The computation of new weights at each node to minimize the error function is performed through learning algorithms. Some of the examples of training algorithms include error-back propagation (EBP) [11],scaled conjugate gradient descent (SCG) [12], levenberg-marquardt (LM) algorithm [13].

G. Genetic & Evolutionary Feature Selection

The process of finding the most relevant and significant inputs for a predictive model is called feature selection. These techniques can be used to identify and remove unneeded, irrelevant and redundant features that do not contribute or decrease the accuracy of the predictive model. One of the most advanced algorithms for feature selection is the genetic algorithm. This is a stochastic method for function optimization based on the mechanics of natural genetics and biological evolution. In this paper a steady-state genetic algorithm has been utilized to find the best feature set for the classifiers. Each individual in the population represents the predictive model. The number of genes is the total number of features in the data set. Genes here are binary-coded, and represent the inclusion or not of particular features in the model. A binary tournament selection method is used to select 2 parents from the population at each evolutionary cycle. Once the parents are chosen, a uniform crossover operator together with a mutation occurrence rate of 0.01 are used to evolve a new offspring.

In order to evaluate the fitness of each individual, the mentioned baseline methods are trained with those features that have a corresponding gene with "True" value, then the average of accuracy measures of the three baseline methods is used as fitness of each individual. Additionally a criterion-based approach for the fitness assignment is used. Two candidate solutions x and y were compared first according to prediction accuracy on the test set, taking the larger of the two as the “winner”. If the two candidate solutions had equal prediction accuracy, they were then compared according to the number of features used, taking the smaller of the two as the “winner” [14]. By doing so, any plausible tie will be broken. As the genetic algorithm is a stochastic optimization method, the genes of the individuals are initialized at random. In fact for creating the initial population, a user-specified number of individuals (otherwise known as population size) will be generated and added to the population. In this paper for generating each individual at the initialization step, for each gene, a Boolean value is selected uniformly and assigned to that gene.

III. Experiment

A. Testing the system

The authorship attribution system was trained with the dataset made using the first 3 text samples from each 25 authors. All 25 authors are labeled from 1000 to 1024. Since the GEFeS method is rather time consuming, it was only run for 1000 function evaluations. The system was further tested using the last sample form each author, in total 25 text files. The highest accuracy measure we achieved was 92 percent. We then attacked five of the test files using different adversarial techniques as follows:

- advTest00.txt: This test file contains a text that was randomly generated from the four text samples from author 1000.
- advTest05.txt: This test file contains 80% of the text from author 1005 and the rest 20% from other authors’ texts.
- advTest10.txt: This test file contains 60% of the text from author 1010 and the rest 40% from author 1005’s texts.
- advTest15.txt: This test file contains 30% of the text from author 1015 and the rest 70% from other authors’ texts.
- advTest20.txt: This test file contains only one small sentence from the author 1020’s text. This test set has very few words to determine the author. It is likely to fail.

We left the rest of the text files as they were. The results are shown in the following section, Section IV

IV. Results

TABLE I
Expected predictions vs AAS’ predictions of adversarial texts

Adversarial Texts	Predictions	
	AAS	Expected
advTest00.txt	1000	1000
advTest05.txt	1005	1005
advTest10.txt	1010	1010
advTest15.txt	1015	1015
advTest20.txt	1021	1020

As mentioned in the previous section, our AAS was tested with 25 text files which resulted in a 92 percent accuracy, that is it only miss-classified two files out of 25 text files. The AAS then was evaluated using the attacked files, the only file that was miss-classified is the last file (Table I). This dropped the accuracy to 88 percent, which is a 4% drop in the accuracy measure.

V. Conclusion

The results show that the genetic & evolutionary feature selection improves the accuracy of machine learning models by choosing the best characters to have as features. All three machine learning models LSVM, RBFSVM and MLP showed significant increase in accuracy as GEFeS evolved the feature mask. Also, using there different classifiers, and a feature mask make the AAS robust against adversarial attacks.

VI. Breakdown of the Work

Each author performed their roles to complete the project. Armin Khayyer and Bhargav Joshi made the code and developed a method to make system resistant to the attacks. Bhargav Joshi tried to attack the system and Armin Khayyer ran the statistics on the attacks. Each author contributed equally to write the paper.

References

[1] W. B. Cavnar, J. M. Trenkle *et al.*, "N-gram-based text categorization," in *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, vol. 161175. Citeseer, 1994.

[2] H. M. Wallach, "Topic modeling: beyond bag-of-words," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 977–984.

[3] M. Eder, J. Rybicki, and M. Kestemont, "Stylometry with r: a package for computational text analysis." *R journal*, vol. 8, no. 1, 2016.

[4] E. Lex, A. Juffinger, and M. Granitzer, "A comparison of stylometric and lexical features for web genre classification and emotion classification in blogs," in *2010 Workshops on Database and Expert Systems Applications*. IEEE, 2010, pp. 10–14.

[5] B. Boser *et al.*, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 1992, pp. 144–152.

[6] F. Rosenblatt, "The Perceptron—a perceiving and recognizing automaton," *Report 85-460-1*, 1957.

[7] Wikipedia, "Perceptron - Wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Perceptron

[8] X. Glorot, A. Bordes, and Y. Bengio, *Deep sparse rectifier neural networks*, 2011, vol. 15. [Online]. Available: http://jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf

[9] Wikipedia, "Multilayer perceptron - Wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Multilayer_perceptron

[10] D. Hunter, H. Yu, M. S. Pukish, III, J. Kolbusz, and B. M. Wilamowski, "Selection of Proper Neural Network Sizes and Architectures—A Comparative Study," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 2, pp. 228–240, may 2012. [Online]. Available: http://ieeexplore.ieee.org/document/6152147/

[11] B. M. Wilamowski, "How to not get frustrated with neural networks," *Proceedings of the IEEE International Conference on Industrial Technology*, pp. 5–11, 2011.

[12] M. Fodslette Møller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, vol. 6, pp. 525–533, 1993.

[13] D. W. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, jun 1963. [Online]. Available: http://epubs.siam.org/doi/10.1137/0111030

[14] K. Casey, A. Garrett, J. Gay, L. Montgomery, and G. Dozier, "An evolutionary approach for achieving scalability with general regression neural networks," *Natural Computing*, vol. 8, no. 1, pp. 133–148, Mar 2009. [Online]. Available: https://doi.org/10.1007/s11047-007-9052-x