Genetic & Evolutionary Feature Selection To Achieve Improved Accuracy of LSVM, RBFSVM and MLP

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

***Abstract*— In this paper, the unigram feature extractor was used on CASIS-25 dataset to build a unigram characters lookup table and extract feature vectors in terms of the frequency of each character in the given sentence. Machine learning models: Linear SVM, Radial Basis Functions SVM and Multi-Layer Percepetron (MLP) neural network were trained using the extracted feature. It was noticed that using appropriate feature masks improved the accuracy of the training. Hence, The feature selection was performed using a genetic algorithm to optimize the feature selection for the machine learning models in order to improve the accuracy of the training. This paper also carries discussion of the results obtained by performing the experiment 30 times.**

I. Introduction

Feature selection, or input selection, is the process of finding the most relevant inputs for a predictive model. 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. In this paper a Genetic & Evolutionary Feature Selection (GEFeS) method has been utilized to find the most important features that results in a good accuracy measure for the three baseline machine learning algorithms named as Linear Support Vector Machines (LSVM), Radial Basis Function Support Vector Machines (RBFSVM), and Multi-layer Perceptron (MLP). The Baseline methods are used to create an Authorship Attribution (AA) system. Authorship Attribution is the process of attempting to identify the likely authorship of a given document, given a collection of documents whose authorship is known. For evolving the best feature set for the baseline models, a dataset of 100 text file with their corresponding author is used. In total there exist 25 authors in the dataset. For preprocessing, a uni-gram feature extraction method has been utilized to extract the features of each text. The masked features of each text together with its ground truth label, corresponding author, are then fed to the baseline models for training the classifiers. Once the classifiers are trained over the training set, the accuracy of them is then calculated over the test set and is further used as fitness value of the GEFeS algorithm. The remaining of this paper is structured as follows: Section II explains Uni-gram feature extraction method, 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.

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

II. Methodology

A. Unigram Feature Extraction

N-grams of texts are extensively used in text mining and natural language processing tasks[1]. They are basically a set of co-occuring words within a given window and when computing the n-grams you typically move one word or character forward. For example, for the sentence "The cow jumps over the moon.", if we focus on word level and $N = 2$ (known as bigram), then the bigram would be: {the cow, cow jumps, jumps over over the, the moon}. When $N = 1$, this is referred to as unigrams and this is essentially the individual words in a sentence. 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 unigram to extract features by focusing on character level. We first build a unigram characters lookup table which contrains all the characters (95 characters), and then the frequency of each character in the given sentence will be calculated. Finally, we use frequency of all the character as the extracted feature vectors.

B. Linear SVM

Support vector machines (SVMs, also support-vector networks[2]) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis[2]. 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. We have assumed the labels are $y \in \{1, -1\}$ rather than $\{0, 1\}$, and we use the hinge loss

as the loss function, which is defined as

$$\ell(\eta) = \max(0, 1 - \eta \cdot y) \tag{6}$$

Note that η should be the "raw" output of the classifier's decision function, not the predicted class label. For instance, in linear SVMs, $\eta = \mathbf{w} \cdot \mathbf{x} + b$, where $\mathbf{w}, b$ are the parameters of the hyperplane and $\mathbf{x}$ is the input variable(s). When y and η have the same sign (meaning η predicts the right class) and $|\eta| \geq 1$, the hinge loss $\ell(\eta) = 0$. When they have opposite signs, $\ell(\eta)$ increases linearly with y, and similarly if $|\eta| < 1$, even if it has the same sign (correct prediction, but not by enough margin). The hinge loss is a convex function, so many of the usual convex optimizers used in machine learning can work with it.

C. 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:

$$\phi(x, x_k) = \exp\left(\frac{-||x - x_k||^2}{2\sigma^2}\right) \tag{7}$$

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.

D. Multi-Layer Percepetron Neural Network

Percepetron neurons are a type of neurons that support supervised learnig of binary classifiers using percepetron algorithm developed by Frank Rosenblatt [3]. 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) [4]:

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

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. Figure 1 shows a 5-3-4-1 MLP network which is interpreted as a network having a layer of 5 inputs and a layer of an output connected through two hidden layers of 3 neurons and 5 neurons respectively. 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 [5][6]. Since MLPs are fully connected, each node in one layer connects with a certain weight w_{ij} to every node in the following layer.

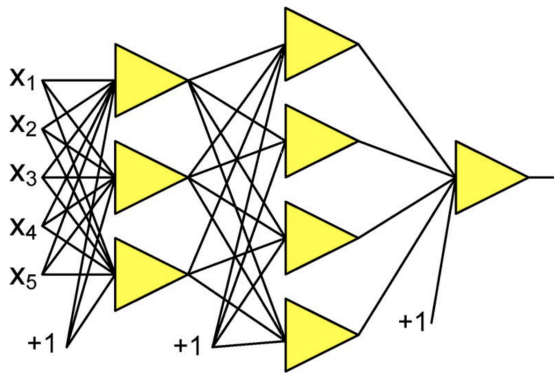

Fig. 1. MLP network with 8 neurons and two hidden layers [7]

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

$$e_j(n) = d_j(n) - y_j(n) \tag{9}$$

where, d_j is the desired output value and y_j is the value produced by the percepetron. The weights at each node are targeted be adjusted such that the total error in the entire output is minimized. This error function is computed as,

$$\mathscr{E}(n) = \frac{1}{2} \sum_j e_j^2(n) \tag{10}$$

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) [8],scaled conjugate gradient descent (SCG) [9], levenberg-marquardt (LM) algorithm [10].

E. 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. Mathematically, inputs selection is formulated as a combinatorial optimization problem. Here the function to optimize is the generalization performance of the predictive model, represented by the error on the selection instances of a data set. The design variables are the inclusion (1) or the exclusion (0) of the input variables in the above-mentioned baselines method. An exhaustive selection of features would evaluate lots of different combinations (2^N, where N is the total number of features). This process requires lots of computational work and, if the number of features is big, becomes impracticable. Therefore, we need intelligent search methods that allow the selection of features in practice.

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

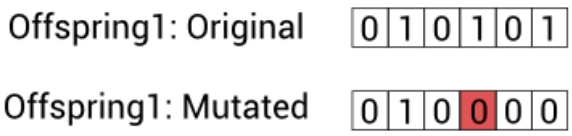

Fig. 2. Mutation Operator [11]

genetics and biological evolution. In this paper a steady-state genetic algorithm has been utilized to find the best feature set for the baseline methods. Each individual in the population represents the baseline predictive models. 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 is used to evolve a new offspring. The crossover operator can generate offspring that are very similar to the parents. This might cause a new generation with low diversity. The mutation operator solve this problem by changing the value of some genes in the offspring at random. To decide if a gene will be mutated, we generate a random number between 0 and 1. If this number is lower than a user-specified value called the mutation rate, that variable is flipped. Since a steady-state genetic algorithm is used, the evolved and mutated offspring will replace the worst-fit individual at each evolutionary cycle.

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” [12]. 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. Since a uniform random process is utilized for each gene, the number of features with true value are distributed based on the Central Limit theorem is following a Normal distribution with a mean value equal to the half of the population size. Therefore as a uni-gram feature extraction with 95 characters is used, the number of active features are usually in the range of 36 to 60, $(\mu \pm 3\sigma)$ where $\mu = 48, \sigma = 4.5$. This could be problematic if the number of active features in the best feature set lies out of this range.

III. Experiment

As mentioned in the last section, Section II, a GEFeS algorithm is wrapped around the baseline methods to evolve the best feature set that results in the highest average accuracy. Each individual of the GEFeS represents a boolean mask with length of 95, corresponding to the number of uni-gram characters, that identifies whether a feature is used or not. Since at each function evaluation of GEFeS algorithm, three baseline models need to be trained, the GEFeS algorithm is rather time-consuming, therefore we let the algorithm cycle for 1000 function evaluations. Also since a steady-state genetic algorithm is used and one offspring is generated at a time, in order to preserve the variety in the population, the initial population is randomly generated with 50 individuals. Moreover, a mutation rate of 0.01 is used to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next. SK-Learn Python package [13] is used to code the baseline models.

IV. Results

Table I shows accuracies of machine learning models obtained from 30 runs of GEFeS. It also shows how many features were selected to train the shown machine learning models. From the results of Table I, it is observable that the MLP and RBFSVM showed better accuracy when compared with that of LSVM. The result of T-test between the accuracy of RBFSVM and the accuracy of MLP shows class-1 equivalance.

Table II shows the mean and standard deviation of the frequency of characters chosen as features during the 30 runs of the experiment. The characters with the value of mean being 1 or close to 1 indicates that those characters were picked the most during all 30 runs of the experiment by GEFeS. These characters (i.e. ’-’, ’.’, ’A’, etc.) are classified as the most consistent characters for the feature mask in order to achieve better accuracy for the machine learning models.

In order to show the improvement achieved by using GEFeS, we ran the baseline code 30 times and collected the accuracy of each baseline model, LSVM , RBFSVM , and MLP. A T-test was performed over each pair of data, collected using GEFeS and baseline model, the achieved p-value for all three pairs are very small (≤ 0.001) which implies that the GEFeS model improves the accuracy measure by evolving the feature set and eliminating the irrelevant features.

A. TF-IDF

Typically, the TF-IDF weight is composed by two terms: the first computes the normalized Term Frequency (TF), otherwise known as the number of times a word appears in a document, divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number

TABLE I

THE FINAL FITNESS , ACCURACIES OF MACHINE LEARNING MODELS AND NO. OF FEATURES SELECTED AFTER 1000 FUNCTION EVALUATIONS OF THE GENETIC ALGORITHM

Run No.	Fitness GA	Accuracy LSVM	Accuracy RBFSVM	Accuracy MLP	Features Selected
1	0.80	0.79	0.79	0.82	52
2	0.80	0.75	0.82	0.83	55
3	0.79	0.75	0.81	0.81	56
4	0.81	0.78	0.82	0.83	51
5	0.81	0.77	0.83	0.84	52
6	0.79	0.78	0.78	0.81	44
7	0.81	0.78	0.81	0.83	59
8	0.81	0.77	0.84	0.82	62
9	0.81	0.80	0.81	0.81	50
10	0.81	0.77	0.82	0.83	46
11	0.81	0.77	0.82	0.83	45
12	0.79	0.73	0.85	0.80	57
13	0.80	0.78	0.78	0.83	52
14	0.80	0.76	0.79	0.84	49
15	0.81	0.79	0.82	0.81	50
16	0.80	0.76	0.81	0.82	62
17	0.81	0.77	0.83	0.82	47
18	0.80	0.75	0.82	0.83	60
19	0.78	0.74	0.81	0.84	50
20	0.80	0.78	0.78	0.84	51
21	0.80	0.79	0.82	0.79	53
22	0.80	0.76	0.83	0.82	54
23	0.82	0.81	0.80	0.85	46
24	0.81	0.77	0.82	0.83	52
25	0.80	0.74	0.83	0.82	58
26	0.79	0.75	0.82	0.81	56
27	0.80	0.76	0.81	0.82	58
28	0.80	0.73	0.84	0.83	51
29	0.81	0.75	0.83	0.84	50
30	0.80	0.77	0.82	0.82	47

TABLE II

MEAN AND STANDARD DEVIATION OF THE FREQUENCY OF CHARACTERS CHOSEN AS FEATURES FOR 30 RUNS OF THE EXPERIMENT

Char	Mean	Std.	Char	Mean	Std.	Char	Mean	Std.
	0.40	0.50	@	0.27	0.45	‘	0.93	0.25
!	0.57	0.50	A	1.00	0.00	a	0.50	0.51
"	0.97	0.18	B	1.00	0.00	b	0.77	0.43
#	0.47	0.51	C	1.00	0.00	c	0.90	0.31
$	0.27	0.45	D	0.07	0.25	d	0.60	0.50
%	0.43	0.50	E	0.70	0.47	e	0.47	0.51
&	0.00	0.00	F	0.43	0.50	f	0.60	0.50
’	0.93	0.25	G	0.27	0.45	g	0.27	0.45
(	0.60	0.50	H	0.37	0.49	h	0.67	0.48
)	0.63	0.49	I	1.00	0.00	i	0.57	0.50
*	0.47	0.51	J	1.00	0.00	j	0.97	0.18
+	0.43	0.50	K	0.90	0.31	k	0.37	0.49
,	0.73	0.45	L	0.00	0.00	l	0.70	0.47
-	1.00	0.00	M	1.00	0.00	m	0.40	0.50
.	1.00	0.00	N	1.00	0.00	n	0.47	0.51
/	0.00	0.00	O	0.80	0.41	o	0.57	0.50
0	0.07	0.25	P	1.00	0.00	p	0.40	0.50
1	0.00	0.00	Q	0.73	0.45	q	0.43	0.50
2	0.50	0.51	R	0.50	0.51	r	0.73	0.45
3	0.03	0.18	S	1.00	0.00	s	0.53	0.51
4	0.23	0.43	T	0.93	0.25	t	0.33	0.48
5	0.27	0.45	U	0.97	0.18	u	0.70	0.47
6	0.10	0.31	V	0.23	0.43	v	0.83	0.38
7	0.07	0.25	W	1.00	0.00	w	0.63	0.49
8	0.30	0.47	X	0.33	0.48	x	0.30	0.47
9	0.33	0.48	Y	0.47	0.51	y	0.87	0.35
:	0.37	0.49	Z	0.40	0.50	z	0.00	0.00
;	0.70	0.47	[	0.53	0.51	{	0.53	0.51
<	0.40	0.50	\	0.53	0.51	|	0.40	0.50
=	0.60	0.50	]	0.83	0.38	}	0.30	0.47
>	0.47	0.51	^	0.53	0.51	∼	0.30	0.47
?	0.93	0.25	_	0.40	0.50			

of the documents in the corpus divided by the number of documents where the specific term appears. Thus, the TF-IDF weight is the product of these two quantities. In order to check the effect of TF-IDF, since training the MLP model is stochastic and will result in different accuracy value at each trial of training, we run the baseline models 10 times with the TF-IDF weight and 10 times without it, this way we can statistically conclude if the results are significantly affected or not. A T-test is performed over the results of the MLP model, a p-value of 0.06 is achieved which is higher than $\alpha = 0.05$, therefore we can conclude that TF-IDF does not have a significant effect on the accuracy of the MLP model. additionally, for the other two baseline models, LSVM and RBFSVM, the accuracy value at each iteration remains the same whether the TF-IDF is used or not.

B. Standardization

Feature standardization makes the values of each feature in the data have zero-mean (when subtracting the mean in the numerator) and unit-variance. A simillar approach to the previouly mentioned approach is taken to conclude statistically if standardization affect the accuracy of the machine learning baseline models. For the first two model, LSVM and RBFSVM, the accuracy measures are the same at each of the 10 iterations that we ran the "Baseline" code without the standardization method and those values are almost 20% less than the values we achieved using the

standardization method. Additionally, for the MLP model since its training process is stochastic, we performed the T-test and found a very small p value, which implies that the mean of the accuracy of the MLP model achieved by using standardization is significantly higher than that of the MLP model achieved by not using standardization.

C. Normalization

Normalization scales all numeric variables in the range [0,1]. Using the similar statistical analysis mentioned in the previous sections, the small p value achieved by the t-test between accuracy of LSVM, RBFSVM and MLP showed significant improvement when compared to the results of when the normalization was used.

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. The statistical analysis of the final accuracies of LSVM, RBFSVM and MLP shows that RBFSVM and MLP had similar accuracy during each run and they were better than LSVM for the most cases. GEFeS showed significantly better accuracy when compared to baselineMLT but it also proved to be a lot more expensive in computational terms hence it took long time to finish executing when compared to baseline code.

VI. Breakdown of the Work

Each author performed their roles to complete the project. Armin Khayyer merged the GA code with baseline code to develop GEFeS. Bhargav Joshi analyzed the results and performed the statistical analysis. Ye Wang executed the GEFeS code on super computer and generated results of 30 runs. 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] 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.

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

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

[5] 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

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

[7] 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/

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

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

[10] 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

[11] NeuralDesigner.com, "Genetic algorithms for feature selection | Machine learning blog."

[12] 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

[13] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.