International Journal of
Intelligent Systems and Applications in Engineering

ISSN:2147-6799 www.ijisae.org Original Research Paper

An Empirical Study of the Extreme Learning Machine for Twitter Sentiment Analysis

Onder COBAN*[1], Buse Melis OZYILDIRIM[2], Selma Ayse OZEL[3]

Accepted : 08/08/2018 Published: 29/09/2018

Abstract: Extreme Learning Machine (ELM) method is proposed for single hidden layer feed-forward networks (SLFNs). The ELM employs feed-forward neural network architecture and works with randomly determined input weights. In this respect, ELM depends on the principle that enables to determine weights and biases in the network. In the first phase of ELM which can be named as feature mapping, random values are used differently from other methods such as Support Vector Machines (SVM) and Deep Neural Networks that employ a kernel function for this purpose. After the feature mapping step, the main goal of the ELM is to learn weights between hidden and output layers by minimizing the error. Therefore, the ELM has gained much more popularity recently; and can be applied for solving various problems like classification, regression, and dimension reduction. In this study, our aim is to apply the basic ELM for making sentiment analysis from Twitter messages as it is considered as a classification task in the literature. To evaluate the performance of the ELM for sentiment analysis, we compare it with the SVM which is one of the most successful machine learning algorithms used for sentiment analysis. To our knowledge, this paper is the first study that employs the ELM for sentiment analysis from Twitter messages written in Turkish. Experimental evaluation of the two methods are done by using two different Turkish Twitter messages datasets. The experimental results showed that the performances of the two methods are slightly different, however SVM outperforms basic ELM.

Keywords: *Twitter, Sentiment Analysis, Support Vector Machine, Extreme Learning Machine*

1. Introduction

Today, social media is not only just a communication tool but also it has become an integral part of our everyday life. Social media users can generally express their thoughts, feelings, ideas, and experiences about any subject by using tools like Google+, Twitter, and Facebook [1]. Social media usage have been increasing day by day therefore, users also employ it for sharing and organizing news about popular entities [2]. In this aspect, social media provides a rich data source that can be used in many research fields such as commerce, politics, economy, and opinion mining.

Sentiment analysis deals with the textual data generated in social media to make analysis about entities like products, movies, companies, people, brands, etc. [3]. Among the social media platforms, Twitter is one of the most popular and it has 313 million monthly active users. It is proved that Twitter is the most used tool for sharing the breaking news, with 40 million tweets posted on the day of the 2016 U.S. Presidential election [4]. As it provides an easy way to access and download published tweets, Twitter is considered as one of the largest data source of user generated content [5]. Therefore, in this study, Twitter is selected as data source and sentiment analysis is performed on Turkish Twitter feeds using Support Vector Machine (SVM) and Extreme Learning Machine (ELM).

The ELM is a kind of single layer feed-forward neural networks (SLFNs) and can be employed for regression and classification. It determines the weights and biases randomly between input and hidden neurons but never updates these values [6]. The ELM learns only these weights by solving a linear model. However, compared to traditional feed-forward neural networks, it provides better performance in generalization, and its training time is remarkably efficient. It is also quite popular recently, due to its successful results in many other fields such as image segmentation, medical diagnosis applications, and forest cover-type prediction [7]. However, in this study, we utilize the basic ELM in sentiment analysis performed on Turkish tweets. Our aim is to investigate the applicability of the ELM in this field. Therefore, we compare the results of the ELM and SVM in terms of accuracy. We prefer the SVM in comparison due to it is generally being the most successful traditional machine learning algorithm in sentiment analysis [8].

The rest of this paper is organized as follows. The related works are summarized in Section 2. The datasets used in the experimental evaluation are briefly described in Section 3. Methods applied for pre-processing and classification stages are described in Section 4. Our experimental results are presented and discussed in Section 5, and finally, summarized results and conclusions are given in the last section.

2. Related Works

Twitter sentiment analysis is generally performed by using traditional machine learning algorithms. The sentiment analysis is usually considered as a text classification problem which is a supervised machine learning task. The ELM has been firstly proposed for supervised learning tasks such as classification and regression. Moreover, researchers have developed the variants of the ELM by making some extensions to use it for supervised [9] and unsupervised [10] learning, and feature selection [11]. It has

[1,2,3]*Computer Eng. Cukurova University, Adana – 01330, TURKEY*
* *Corresponding Author: Email: onder.cbn@gmail.com*

also been employed to build applications in many other research fields such as computer vision, system modeling, control and robotics, and image processing [7].

Both in sentiment analysis and text classification, there are many studies which utilize the ELM algorithm for different purposes. Poria *et al.* employed the ELM to build a new version of SenticNet and created dependency-based rules for concept level polarity detection [12]. Cambria *et al.* proposed an ELM based architecture for emoticon categorization [13]. Zheng *et al.* presented a text categorization framework combining the Latent Semantic Analysis and Regularization Extreme Learning Machine [14]. Zhao *et al.* employed the ELM algorithm for XML document classification and proposed a voting ELM algorithm to improve the accuracy of the ELM [15]. To our best knowledge, there is no previous work utilizing the ELM for sentiment analysis of Turkish tweets. Therefore, in this paper, we performed the sentiment analysis using the ELM on Turkish tweets. Our aim is to investigate the applicability of the ELM for this domain by comparing it with the SVM which is the most successful classifier in this field in general.

3. Datasets Used in the Experiments

In this section, we describe the datasets used for the evaluation of the ELM. Sentiment analysis on Twitter is generally performed on two different types of datasets including subject-dependent and subject-independent [16]. Subject-dependent means the dataset is composed of tweets that related to any topic, whereas subject-independent represents the dataset containing tweets that does not have relation with any topic. In this study, we use two different Turkish Twitter datasets from which the Dataset1 (DS1) is subject-dependent and the second one (DS2) is subject-independent. The DS1 [17] has manually labeled tweets which are related with a private company in the telecom sector. It contains totally 3000 tweets in three categories that are positive, negative, and neutral. The second dataset (DS2) is a subset of another dataset [18] which is automatically collected and labeled by using emoticons contained in the tweets (i.e., distant supervision) with the help of Twitter API. This dataset contains 32000 tweets which are equally distributed in positive and negative categories. However, we use a subset of this dataset to form DS2 as the size of the original dataset is out of the scope for this paper. The DS2 is composed of 10000 tweets in total which have equal distribution for positive and negative classes. The distributions of samples both in DS1 and DS2 datasets are given in Table 1.

Table 1. Class distribution of tweets for DS1 and DS2 datasets

Category	**Dataset**	
	DS1	**DS2**
Positive	756	5000
Negative	1287	5000
Neutral	957	N/A

4. Methods

We apply five main steps to perform sentiment analysis. These steps consist of (i) pre-processing, (ii) feature extraction, (iii) term weighting, (iv) classification, and (v) evaluation.

In the pre-processing step, we clean the Twitter feeds from the incomprehensible words and characters. Then, we extract features in two different ways including Bag of Words (BoW) and n-gram. After the feature extraction phase, weights are assigned to features and document vectors are computed. Then we build a learning system by training a classifier. In the final phase, we employ this learning system to make polarity classification of the tweets. In the learning system, we just employ the ELM and SVM algorithms as classifiers to compare their performances with each other.

4.1. Pre-processing

As the pre-processing task we apply the following steps to extract meaningful information from the tweets:

- Cleaning noise from the tweets by removing the incomprehensible words and meaningless characters (only in BoW model) such as punctuation marks and digits.
- Lowercase conversion to remove case sensitivity of features.
- Applying text normalization to prevent from having high dimensional feature space by removing the repeating letters and reducing them to one letter as in [19].
- Removing the tweets that have less than one feature (i.e., word or n-gram) by applying minimum term count filter.
- Stripping the features, whose length is less than two characters, out from the tweets.
- Removing the Twitter specific terms (e.g., emoticons, URLs, hashtags, etc.) to make the classifier learn only from the meaningful textual content [18, 20].

Sentiment analysis has sparse and high dimensional feature space as it is considered as text classification task. Especially, the high dimensionality of feature space stems from the traditional feature extraction models. Therefore, to prevent from having high dimensional feature space we also apply the following steps only in BoW model:

- Stop-words removal, and
- Stemming

We remove the stop-words from the tweets by using the default list in Lucene API [21]. We also perform word stemming by utilizing the Zemberek that is an open source Turkish NLP tool [22]. After the completion of the pre-processing steps which are illustrated in Table 2, we tokenize the tweets and extract features to classify the datasets.

4.2. Feature Extraction

In Vector Space Model (VSM), the data instances are converted to numerical vectors by using different methods which have high

Table 2. An example for pre-processing steps applied to a tweet

Pre-processing Step	**Tweet**
No (Original Content)	*“@i***n Yakın olsak ben derdimmm :D Ama ne yazık ki, güzel insanlarla komşu olamadık hiççç..!”*
Cleaning	*“Yakın olsak ben derdimmm Ama ne yazık ki güzel insanlarla komşu olamadık hiççç”*
Lowercase Conversion	*“yakın olsak ben derdimmm ama ne yazık ki güzel insanlarla komşu olamadık hiççç”*
Text Normalization	*“yakın olsak ben derdim ama ne yazık ki güzel insanlarla komşu olamadık hiç”*
Stop-words Removal	*“yakın olsak derdim yazık güzel insanlarla komşu olamadık”*
Stemming	*“yakın ol de yazık güzel insan komşu ol”*

impact on the accuracy of the classification system applied. Data instances (i.e., tweets) are represented as numerical vectors by using the extracted features and assigning weights to these features. In this study, we use two different methods that are BoW and n-gram (i.e., trigram) to extract features. We apply both methods to the two datasets, therefore we have 2 different representations for each dataset. We call them according to the feature extraction method used as $DS1_{BoW}$, $DS1_{Trigram}$, $DS2_{BoW}$, and $DS2_{Trigram}$ respectively.

In BoW model, numerical vector representation of a document is often done by associating a word with a numerical weight which is generally proportional to its frequency on the document [23]. Therefore, each word is taken as a feature in BoW model.

The n-gram model, on the other hands, is an alternative feature extraction technique. It can be applied in two different ways: i) character level, and ii) word level n-grams. In character level n-gram model, the features are formed by taking *n* consecutive characters in the text content. For example, the character level n-grams of the string "*opinion mining*" are obtained as follows:

- 2-gram (bigram): |op|, |pi|, |in|, |ni|, |io|, |on|, |n_|, |_m|, |mi|, |in|, |ni|, etc.
- 3-gram (trigram): |opi|, |pin|, |ini|, |nio|, |ion|, |on_|, |n_m|, |_mi|, |min|, |ini|, |nin|, etc.

where '_' represents whitespace character, and '|' is used to show the n-gram boundary. In word level n-gram representation, the adjacent *n* words are taken as a feature. For the same example above, word level 1-grams and 2-gram are as follows:

- 1-gram (unigram): |opinion|, |mining|
- 2-gram (bigram): |opinion mining|

where, we do not have any 3-gram. Character level n-grams are language independent and they can handle with misspelling and abbreviations [24, 25]. Therefore, in this study, we use character level trigrams.

4.3. Term Weighting

After all features are extracted from the whole document collection by using either BoW, or n-gram methods, weight of each feature for each document is computed to form the document vector. Term weighting is a process to determine the importance of a term for a document. For this reason, it has an important role in the correct and effective representation of textual data. In this study, we use Term Frequency-Inverse Document Frequency (TF*IDF) that is the most widely used unsupervised and traditional weighting method [27] to assign weights to terms. The TF*IDF can be formulated as follows:

$$W_{TF*IDF(t,d)} = TF_{(t,d)} * log \frac{N}{|\{d \in D : t \in d\}|} \quad (1)$$

where, $t, d, N,$ and D represent any term, document, number of documents (i.e., tweets) in the collection, and the document collection, respectively. The TF also corresponds to the observed raw frequency of term t in document d.

In VSM, a document vector is formed for a document *i* as shown in equation 2 [26]:

$$d_i = (t_{1i}, t_{2i}, \dots, t_{mi})^T \quad (2)$$

where d_i is document vector for document i and t_{ji} represents weight of term j in document i, and m is the total number of features extracted from the whole dataset. Therefore, all document collection can be represented as a matrix A as in equation 3.

$$A = [d_1, d_2, \dots, d_n] \quad (3)$$

4.4. Extreme Learning Machine

The ELM was proposed by Huang *et al.* [28] and its architecture is shown in Figure 1. As it can be seen from Figure 1, ELM is a type of SLFNs. The main idea of the ELM is to initialize the neural network weights and biases (between input and hidden neurons) randomly. Then weights between the hidden and output layer are obtained analytically [29]. For N arbitrary distinct samples (X_i, t_i), where X_i is an $n \times 1$ input vector $X_i = [x_{i1}, x_{i2}, x_{i3}, \dots, x_{in}]^T$, and t_i is an $m \times 1$ target vector $t_i = [t_{i1}, t_{i2}, t_{i3}, \dots, t_{im}]^T$, the output of a standard ELM with L hidden nodes and an activation function $g(x)$ can be modelled mathematically as in equation 4.

$$\sum_{i=1}^{L} \beta_i g(w_i \cdot x_j + b_i) = o_j, \quad \text{for } j = 1,2,\dots,N \quad (4)$$

where $g(x)$ is nonlinear infinitely differentiable function in any interval, $w_i = [w_{i1}, w_{i2}, w_{i3}, \dots, w_{in}]^T$ is the weight vector that represents connections between input neurons and ith hidden neuron, $\beta_i = [\beta_{i1}, \beta_{i2}, \beta_{i3}, \dots, \beta_{im}]^T$ is the weight vector that represents connections between the ith hidden neuron and output neurons, and b_i corresponds to the bias of the ith hidden neuron respectively [30]. $w_i \cdot x_j$ indicates the inner product of w_i and x_j.

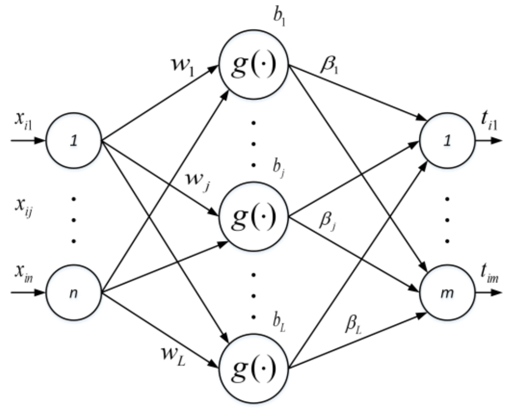


Figure 1. ELM architecture

The ELM can approximate these N samples with zero error, which means that $\sum_{j=1}^{L} ||o_j - t_j|| = 0$, that is, there exist (w_i, b_i) and β_i [9] such that:

$$\sum_{i=1}^{L} \beta_i g(w_i \cdot x_j + b_i) = t_j, \quad \text{for } j = 1,2,\dots,N \quad (5)$$

For N sample, the above equation can be written in a more compact format as:

$$H\beta = T \quad (6)$$

where,

$$H = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_L \cdot x_1 + b_L) \\ \vdots & \ddots & \vdots \\ g(w_1 \cdot x_N + b_1) & \cdots & g(w_L \cdot x_N + b_L) \end{bmatrix}_{N \times L}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L\times m} \quad \text{and} \quad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{N\times m}$$

In equations above, H, β, and T are matrices represent the output matrix of hidden layer, the output weight matrix, and the target matrix, respectively. The ith column of H is the ith sample output vector for samples and the ith row is the ith sample output vector for all hidden neurons. As activation functions, different nonlinear functions can be used such as Sigmoid, Sinusoid, and Gaussian. Finding the least-squares solution $\hat{\beta}$ of the linear system given in equation (6) is simply equivalent to train an ELM. Then, the smallest norm least squares of equation (6) is obtained using the Moore-Penrose generalized inverse definition [31].

$$\hat{\beta} = H^{\dagger}T \tag{7}$$

where, $H^{\dagger}$ is Moore-Penrose generalized inverse of the matrix H.

4.4.1. Activation Function

Traditional gradient-based learning algorithms only work with differentiable activation functions. However, the ELM can work with all bounded nonconstant piecewise and continuous activation functions [32]. In ELM, the H that is the output matrix of hidden layer is calculated by using an activation function. In this study, we use five different activation functions including Sigmoid, Sine (Sin), Hard-limit (Hardlim), Radial-basis (Radbas), and Linear transfer (Purelin) to analyse the effect on performance of the ELM. These functions are formulated as follows:

$$Sigmoid(n) = 1/(1 + exp(-1 \times n)) \tag{8}$$

$$Sine(n) = \sin(n) \tag{9}$$

$$Hardlim(n) = \begin{cases} 1, & if\ n \geq 0 \\ 0, & otherwise \end{cases} \tag{10}$$

$$Radbas(n) = exp(-1 \times n^2) \tag{11}$$

$$Purelin(n) = n \tag{12}$$

In above equations, n is the output of each neuron in the hidden layer which is represented as $w_i \cdot x_j + b_i$.

4.4.2. Example

To apply ELM for classification, first of all, features extracted in the pre-processing step and their computed weight values for each tweet are taken as shown below:

$$x = \begin{bmatrix} (Tweet_1) & 0 & 0 & 1 & 1 & 0 & 0 \\ (Tweet_2) & 1 & 0 & 0 & 0 & 0 & 1 \\ (Tweet_3) & 0 & 1 & 0 & 0 & 1 & 0 \\ (Tweet_4) & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad t = \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}$$

where we have 4 tweets and 6 features such that matrix x represents feature weights for the training dataset such that in each row we have a document vector for each tweet; and t denotes the class labels of each tweet in matrix x. In the training phase, the matrices for input weights (w) and the biases of hidden neurons (b) are randomly initialized as shown below. In our example, the number of neurons in the hidden layer is taken as $L = 4$, which corresponds to the number of tweets in the training set. Therefore, the input weight matrix has 4 (i.e., # of tweets) rows and 6 (i.e., # of features) columns.

$$w = \begin{bmatrix} 0.78 & 0.41 & 0.37 & 0.74 & 0.37 & 0.21 \\ 0.85 & 0.46 & 0.76 & 0.36 & 0.44 & 0.13 \\ 0.73 & 0.66 & 0.17 & 0.95 & 0.72 & 0.26 \\ 0.55 & 0.43 & 0.05 & 0.11 & 0.71 & 0.14 \end{bmatrix} \quad b = \begin{bmatrix} 0.98 \\ 0.40 \\ 0.21 \\ 0.30 \end{bmatrix}$$

After randomly initialization of the input weights and biases, the matrix for the output of the hidden layer H is calculated with the following equation:

$$\mathrm{H} = \sum_{i=1}^{N} g(w_i \cdot x_j + b_i) \tag{13}$$

In above equation, g represents the activation function. In this example, we use the sigmoid activation function. Following the calculation of H, the output weights could then be calculated by using equation (4). The matrix H and resulting β are as follows:

$$H = \begin{bmatrix} 0.89 & 0.87 & 0.85 & 0.85 \\ 0.82 & 0.80 & 0.78 & 0.77 \\ 0.79 & 0.77 & 0.83 & 0.72 \\ 0.61 & 0.73 & 0.81 & 0.70 \end{bmatrix}$$

$$\beta = \begin{bmatrix} -57.693 & 482.609 & -116.98 & -299.08 \\ 66.3809 & -569.61 & 118.394 & 371.656 \\ 5.53886 & 64.7862 & 0.40749 & -68.685 \\ -10.575 & -23.298 & 9.16500 & 23.6135 \end{bmatrix}$$

After the output weights are computed, the training phase is completed. Then, we can classify a previously unseen test data (with the same number of hidden neuron and feature space) by using the computed output weights. In the testing phase, if the following vector x for a tweet with class label t which is 3 is given to the trained ELM system,

$$x = [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0] \quad \text{and} \quad t = [3]$$

then, the H matrix for the test tweet is calculated by applying the same steps as in the training phase. Therefore, H_{Test} is computed as follows:

$$H_{Test} = \begin{bmatrix} 0.85 \\ 0.77 \\ 0.72 \\ 0.70 \end{bmatrix}$$

After that the class label of the test tweet is predicted by using the equation $Y_{Out} = H_{Test}\beta$ as shown below:

$$Y_{Out} = \begin{bmatrix} -1.00 \\ -0.99 \\ -1.00 \\ 1.00 \end{bmatrix}$$

The resulting Y_{Out} shows the prediction of the ELM. As it can be seen from Y_{Out} the last value is the maximum of all values, therefore the predicted class label is equal to last class which is 3 (see matrix t in the training phase for all possible classes). This prediction is correct for the given test instance.

In this study, we also use SVM with linear kernel, which is a kernel-based and robust machine learning algorithm for sparse data, to make comparison with ELM. It has two types in practice including linear and non-linear [33] SVM. Linear SVM aims to find a hyperplane that has maximum margin between classes without increasing dimension of the feature space. Non-linear SVM transforms the data to a higher dimension by a kernel function (e.g., Radial basis, Sigmoid) and performs classification in this space. SVM is mainly concerned with solving binary class problems, however, it can be employed successfully in multiclass classification by different methods such as one-against-one and one-against-all [34].

4.5. Performance Evaluation Measures

In this section, we describe the validation approach and performance evaluation measures that we use to compare the classification methods. We employ n-fold cross validation (CV) approach [36] to validate the predictive model. We prefer this method especially by considering the ELM. Since the ELM uses random values in the computations, even if we use the same activation function and the same number of hidden neurons, the input weights and the hidden layer biases may change. Therefore, the ELM should be run multiple times. In the n-fold CV, the dataset is randomly divided into n disjoint subsets, then one of these subsets is taken as the test set whereas the rest are used as the training set. This process is repeated for each subset. The overall success of the classification model is calculated by taking the average of these n iterations. In this study, we perform 10 folds CV.

Table 3. The feature reduction in pre-processing phase

Features	Dataset			
	DS1		DS2	
	# of features	Percent	# of features	Percent
None	35732	100%	93665	100%
(-) Usernames	N/A	0%	N/A	0%
(-) URLs	N/A	0%	1445	1.54%
(-) Hashtags	380	1.06%	1539	1.64%
(-) Norm. Terms	351	0.98%	1590	1.69%
(-) Emoticons	684	1.91%	8701	9.28%
(-) All	1415	3.95%	13275	14.15%

Table 4. Total number of samples and unique features for DS1 and DS2 datasets after pre-processing

Dataset	Statistic	
	# of samples	# of unique features
$DS1_{BoW}$	2927	2620
$DS1_{Trigram}$	2996	6781
$DS2_{BoW}$	9585	3962
$DS2_{Trigram}$	9957	9110

To compute success rate of the classification models we use precision (P), recall (R), F-measure ($F1$), and accuracy (Acc) values that are computed as in equations 14, 15, 16, and 17, respectively.

$$P = TP/(TP + FP) \tag{14}$$

$$R = TP/(TP + FN) \tag{15}$$

$$F1 = 2 \times (P \times R)/(P + R) \tag{16}$$

$$Acc = (TP + TN)/N \tag{17}$$

where N represents the number of samples in test dataset, TP, TN, FP, and FN values have the following definitions [37] as shown in Figure 2.

- TP: Correctly classified tweets (e.g., a positive tweet is classified as positive)
- TN: Correctly rejected tweets (e.g., a negative tweet is classified as negative)
- FP: Incorrectly classified tweets (e.g., a negative tweet is classified as positive)
- FN: Incorrectly rejected tweets (e.g., a positive tweet is classified as negative)

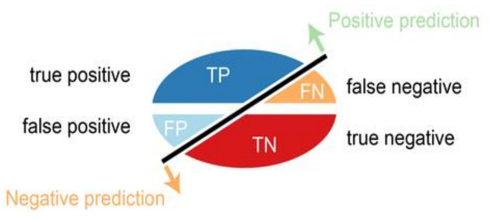
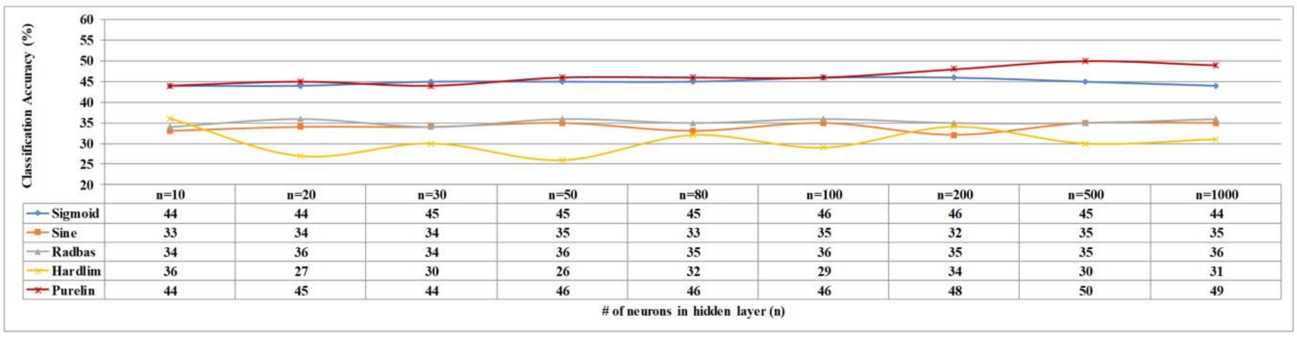


Figure 2. The four outcomes of a classification [35]

5. Experimental Results

To perform sentiment analysis, first we apply pre-processing to the datasets by removing the emoticons, short terms having length less than two characters, and other Twitter specific terms. In this way, we use only the textual content to train the classifier. Therefore, the size of the feature space is reduced by 3.95% and 14.15% for DS1 and DS2 respectively as shown in Table 3. We also remove some samples, which does not have enough number of terms, from the datasets. After the preprocessing phase, the total number of samples for each datasets is given in Table 4. Then we extract the features by using BoW and character level trigram methods. As it can be seen from Table 4, the number of BoW features is less than that of trigram features for each dataset. The main reason for this is that we apply stemming and stop-words removal only in BoW model. In the last step of the pre-processing, we compute term weights. Finally, we perform the classification to obtain the experimental results.

Our experiments consist of two phases. First, we conduct the experiments on $DS1_{BoW}$ to find the best combination of the number of neurons in the hidden layer and the activation function for ELM. We use five different activation functions and nine different values for the number of neurons in the hidden layer resulting in a total of 45 combinations. The five different activation functions are *sigmoid*, *sine*, *radbas*, *hardlim*, and *purelin*. The nine different values of the number of neurons in the hidden layer are 10, 20, 30, 50, 80, 100, 200, 500, and 1000. According to the results of the first phase, we found that performance of the ELM is not so sensitive for the number of neurons in the hidden layer. However, the most successful result is generally obtained when the number of neurons in the hidden layer is selected as 500. We also observed that the activation function is more effective on the performance of the ELM with respect to the number of neurons in the hidden layer. As it can be seen from Figure 3, the most successful activation function is purelin with one exceptional case. Therefore, we decided to use the purelin function, and 500 neurons in the hidden layer for ELM in subsequent experiments.



	n=10	n=20	n=30	n=50	n=80	n=100	n=200	n=500	n=1000
Sigmoid	44	44	45	45	45	46	46	45	44
Sine	33	34	34	35	33	35	32	35	35
Radbas	34	36	34	36	35	36	35	35	36
Hardlim	36	27	30	26	32	29	34	30	31
Purelin	44	45	44	46	46	46	48	50	49

Figure 3. Classification accuracies of ELM on DS1$_{BoW}$ for different activation functions and the number of neurons in the hidden layer

In the second phase of our experiments, our aim is to compare the basic ELM with SVM. For this purpose, we perform classification using both ELM (using best parameter configuration) and SVM on DS1 and DS2 datasets under different feature extraction models. We report our results in Table 5 and Table 6, respectively. In Table 6, each letter (A, B, C, etc.) corresponds to the related classifier and dataset combination used in Table 5. As an example, letter A in Table 5 refers to the accuracy of classification done on DS1$_{BoW}$ by using ELM, however letter A in Table 6 refers to the same accuracy value, also the row starting with letter A in Table 6 presents precision, recall, and F-measure of classification on DS1$_{BoW}$ using ELM. According to our results, the performance of both classifiers are quite close to each other. However, SVM is generally more successful than basic ELM. We also observe that both classifiers produce more successful results on DS2 dataset. In addition, it becomes clear that ELM generally produces better results on BoW features for each dataset, whereas vice versa for SVM.

Table 5. Comparison of two classifiers when ELM has 500 neurons and employs *purelin* activation function in its hidden layer

Dataset	**Classification accuracy (Acc)**	
	ELM	**SVM**
DS1$_{BoW}$	**0.50** (A)	0.49 (B)
DS1$_{Trigram}$	0.49 (C)	**0.54** (D)
DS2$_{BoW}$	0.68 (E)	0.72 (F)
DS2$_{Trigram}$	**0.70** (G)	**0.74** (H)

Table 6. Weighted average values of evaluation measures for the classification accuracies given Table 5

Evaluation Measure			
Acc	**P**	**R**	**F1**
A (**0.50**)	0.567	0.652	0.606
B (0.49)	0.500	0.492	0.495
C (0.49)	0.491	0.497	0.494
D (**0.54**)	0.540	0.541	0.541
E (0.68)	0.689	0.689	0.689
F (0.72)	0.730	0.730	0.730
G (**0.70**)	0.705	0.705	0.705
H (**0.74**)	0.749	0.749	0.749

6. Conclusion and Future Works

In this study, we employ the basic ELM in Twitter sentiment analysis. Our main goal is to investigate the applicability of the ELM for sentiment analysis by comparing it with SVM which is generally the most successful traditional machine learning algorithm. For this purpose, we use two different Twitter datasets which consist of Turkish tweets. According to experimental results, we report that SVM slightly has better classification performance than the basic ELM on the datasets that we used. We think that the reason for this is the robustness of the SVM to the data sparsity. The data sparsity stems from Twitter specific circumstances such as the length restriction and use of informal language in tweets. The success of ELM is not so sensitive to the number of neurons in the hidden layer. However, we observed that performance of the ELM generally rises when the number of neurons in the hidden layer increases. On the contrary, the activation function has a remarkable effect on the success of the ELM. Among the activation functions, the purelin is the most successful since it does not change the TF*IDF weights of features, whereas other activation functions remove the effect of the term weighting process. Consequently, we conclude that there is only a slight difference between the accuracies of the two classifiers. The basic ELM is quite successful in Twitter sentiment analysis, and it has high generalization performance even though its random parts. It is also efficient in terms of training time when compared to the SVM.

In previous works, researchers made different extensions on the architecture of the basic ELM to improve its performance. By making such extensions they can produce better results when compared to the basic ELM. Therefore, in our future work, we are planning to implement a kernel-based ELM, to improve the generalization performance of the basic ELM. We hope that, we may increase the accuracy of the ELM in Twitter sentiment analysis.

References

[1] S. Sommer et al., “Analyzing customer sentiments in microblogs–A topic-model-based approach for Twitter datasets,” In Proceedings of the Americas Conference on Information Systems, Detroit, USA, 2011.

[2] M. Michelson and S. A. Macskassy, “Discovering users’ topics of interest on twitter: a first look,” In *Proceedings of the fourth workshop on Analytics for noisy unstructured text data*, Toronto, Canada, 2010, pp. 73-80.

[3] *A Survey of Opinion Mining and Sentiment Analysis*, B. Liu and L. Zhang, In Mining Text Data, C. Aggarwal, C. Zhai, eds, Boston, MA, Springer, Boston, 2012.

[4] Twitter, 2016; Available from: https://en.wikipedia.org/wiki/Twitter

[5] A. Giachanou and F. Crestani, “Like it or not: A survey of twitter sentiment analysis methods,” *ACM Computing Surveys.*, vol. 49, no. 2, 2016.

[6] G. Huang *et al.*, “Trends in extreme learning machines: a review,”

Neural Networks, vol. 61, pp. 32-48, 2015.

[7] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489-501, 2006.

[8] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," In *10th European Conference on Machine Learning*, Chemnitz, Germany, 1998, pp. 137-142.

[9] G. B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107-122, 2011.

[10] G. Huang *et al.*, "Semi-supervised and unsupervised extreme learning machines," *IEEE transactions on cybernetics*, vol. 4, no. 12, pp. 2405-2417, 2014.

[11] F. Benoít *et al.*, "Feature selection for nonlinear models with extreme learning machines," *Neurocomputing*, vol. 102, pp. 111-124, 2013.

[12] S. Poria *et al.*, "Sentic patterns: Dependency-based rules for concept-level sentiment analysis," *Knowledge-Based Systems*, vol. 69, pp. 45-63, 2014.

[13] E. Cambria *et al.*, "An ELM-based model for affective analogical reasoning," *Neurocomputing*, vol. 149, pp. 443-455, 2015.

[14] W. Zheng, Y. Qian, and H. Lu, "Text categorization based on regularization extreme learning machine," *Neural Computing and Applications*, vol. 22, no. 3-4, pp. 447-456, 2013.

[15] X. G. Zhao *et al.*, "XML document classification based on ELM," *Neurocomputing*, vol. 74, no. 16, pp. 2444–2451, 2011.

[16] X. Wang *et al.*, "A depression detection model based on sentiment analysis in micro-blog social network," In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Berlin, Heidelberg, 2013, pp. 201-213.

[17] M. Çetin and M. F. Amasyali, "Supervised and traditional term weighting methods for sentiment analysis," In *Signal Processing and Communications Applications Conference*, Haspolat, Turkey, 2013, pp. 1-4.

[18] A. Hayran and M. Sert, "Sentiment analysis on microblog data based on word embedding and fusion techniques," In *Signal Processing and Communications Applications Conference*, Antalya, Turkey, 2017, pp. 1-4.

[19] O. Coban, B. Ozyer, and G. T. Ozyer, "A Comparison of Similarity Metrics for Sentiment Analysis on Turkish Twitter Feeds," In *IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, Chengdu, China, 2015, pp. 333-338.

[20] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," Stanford, CA, USA, CS224N Project Report, 2009.

[21] Apache. Lucene. Available from: https://lucene.apache.org/core/

[22] A. A. Akın and M. D. Akın, "Zemberek, an open source nlp framework for turkic languages," *Structure*, vol. 10, 2007.

[23] C. Whitelaw, N. Garg, and S. Argamon, "Using appraisal groups for sentiment analysis," In *Proceedings of the 14th ACM international conference on Information and knowledge management*, Bremen, Germany, 2005, pp. 625-631.

[24] I. Kanaris *et al.*, "Words vs. character n-grams for anti-spam filtering," *International Journal on Artificial Intelligence Tools*, pp. 1-20, 2006.

[25] H. Lodhi *et al.*, "Text classification using string kernels," *Journal of Machine Learning Research*, vol. 2, pp. 419-444, 2002.

[26] Ö. Çoban, and G. T. Özyer, "Twitter duygu analizinde terim ağırlıklandırma yönteminin etkisi," *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, vol. 24, no. 2, pp. 283-291, 2018.

[27] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513-523, 1988.

[28] G. B. Huang and H.A. Babri, "Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions," *IEEE Trans. Neural Networks*., vol. no. 1, pp. 224–229, 1998.

[29] Y. W. Huang and D. H. Lai, "Hidden node optimization for extreme learning machine," *Aasri Procedia.*, vol. 3, pp. 375–380, 2012.

[30] S. Xu, J. Wang, "A fast incremental extreme learning machine algorithm for data streams classification," *Expert Systems with Applications*, vol. 65, pp. 332-344, 2016.

[31] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," In *Proceedings of the IEEE International Joint Conference on Neural Networks*, Budapest, Hungary, 2004, pp. 985-990.

[32] A. Gosso, and M. A. Gosso, "Package 'elmNN'," ELM Package Version 1.0, July. 17, 2012. [Online]. Available: https://cran.r-project.org/web/packages/elmNN/index.html

[33] Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. Data mining and knowledge discovery, 2(2), 121-167.

[34] D. Fradkin and I. Muchnik, "Support vector machines for classification," *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 70, 13-20, 2006.

[35] The four outcomes of a classifier. Available from: https://classeval.wordpress.com/introduction/basic-evaluation-measures/

[36] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," In *International Joint Conference on Artificial Intelligence*, Montreal, Canada, 1995, pp. 1137-1145.

[37] L.J. Sheela, "A Review of Sentiment Analysis in Twitter Data Using Hadoop," *International Journal of Database Theory and Application*, vol. 9, no. 1, pp. 77-86, 2016.