Probabilistic Neural Network and Word Embedding for Sentiment Analysis

Saqib Alam, Nianmin Yao
School of Electrical Information and Electrical Engineering
Dalian University of Technology
Liaoning, Dalian 116024

***Abstract*—In the present days, Artificial Intelligence (AI) is an attractive area of research along with numerous practicable purposes and vigorous subject matters and tasks, such as, understand speech, natural language, diagnose medicine and support basic research. In this study deep learning (DL) techniques, i.e. Probabilistic Neural Network (PNN) and Word Embedding (WE) will be used for sentiment analysis. The entire proposed framework will be divided into three phases: (a) normalization, (b) word vectorization, and (c) execution of proposed model.**

***Keywords*—Deep learning; probabilistic neural network; word embedding; sentiment analysis**

I. Introduction

Artificial Intelligence (AI) become a new trend, leaving behind other areas of research such as Big Data (BD), Internet of Things (IoT), Virtual Reality and many more in the past. In this new revolution of AI, DL turned into a hot zone for analysts, as a subset of Machine Learning (ML). It is a technique that uses many layers of non-leaner information processing for supervised or unsupervised classification, pattern analysis, transformation and feature extraction [1]. DL and its techniques have been used broadly in different areas of research and have an impressive impact on Natural Language Processing (NLP) [2]. In areas such as artificial vision and natural language processing, DL have impressive results because it is hard to use a strict mathematical model to characterize real-world images or languages. For example, it is almost near to impossible to write a powerful algorithm to detect handwriting or objects in an image, it is now a simple implementation of a DL algorithm that learns to perform tasks that exceed human accuracy levels [3]. Previously for sentiment analysis, some DL techniques were used which were enormously effective, such as word2vector [4]. In our this work we will use WE [4] with Probabilistic Neural Network (PNN) to increase the accuracy of sentiment analysis [5]. PNN is extensively adopted by researchers for pattern recognition and classification. We adopted PNN for some of its advantages, such as the training is easy and instantaneous in PNN [6]. PNN has Bayes optimal classification, much faster and accurate than multilayer perception networks [7]. PNN networks generate accurate predicted target probability scores.

This study includes five sections: (I) introduction and background of this study, (II) related work, (III) material and works which elaborate the three phases of this study, (IV) results and discussion of this research work, and (V) the conclusion.

II. Literature Review

Previously PNN was used mostly in image processing, such as [8] proposed in his work a modified PNN novel, the improvements were made by replacing the exponential activation function in the pattern layer of the PNN to the *complex* exponential function. The properties of the classical PNN such as fast training procedure and the convergence to the optimal Bayesian decision were same, but theoretically the recognition performance and space complexity should be approximately (*R* / *C*)2 / 3–times lower. During the experiment, they described a protocol for comparing image recognition methods using well-known data sets, in the situation of small sample problems. The experiments of modern deep neural network model show that due to its high accuracy and low computational complexity, the implementation of the maximum a posteriori (MAP) program is considered very promising in various tasks related to image recognition. Of course, the proposed algorithm is not the most accurate classifier in all cases, especially if the number of occurrences of each class is quite large. However, it has been shown that its method allows the treatment of PNN defects caused by brute force processing of all instances. In addition, unlike the original PNN, its modification allows you to choose a compromise solution between accuracy and computational complexity. Author in [6] reviewed two methods: PNN and the polynomial Adaline for classification based on Bayes strategy and nonparametric estimators for probability density functions. He found the most significant advantage of the PNN network is that the training is straightforward and can be used in real time, as the network can begin to summarize the new model as long as the patterns representing each category are observed. PNN has other advantages. (1) By selecting the appropriate smoothing parameter values, the shape of the decision surface can be made as complex as possible or as simple as possible. (2) The area where the decision area can approach the optimal minimum risk decision (Bayesian criteria). (3) The wrong sample can be tolerated. (4) Rare samples are sufficient to meet the performance of the network. (5) For the statistics that change over time, the old model may be overwritten by the new model. The PNN paradigm has been compared with the popular back propagation neural networks to classify and obtained data as actual measurements of electron emitters. In this particular experiment, PNN trained 200,000 times faster than back propagation. The disadvantage

of PNN is the need to store and use the entire training database to test unknown patterns. In the case of very large databases and mature applications where test time is more important than training time, the Adalin polynomial paradigm has been developed, which does not have these limitations.

In this study [9] proposed a hybrid model was proposed Which includes a PNN and two layer restricted Boltzmann (RBM). The objective of this hybrid model of deep learning is designed to achieve better classification accuracy of the SA, i.e. negative and positive polarity, according to different situation, the model works very well. The experiments were conducted with Panga and Li, and Blitzer et al. datasets, binary classification was implemented in each data set. Accuracy been improved as compared to existing and advanced technology of [10].

In the review study [2] described plenty of studies related to sentiment analysis by using DL models. By analyzing all these studies, it was found that by using the DL method, SA can be analyzed in a more efficient and accurate manner. Since SA is used to predict user views, and the DL model is based on the prediction or imitation of the human mind, the DL model provides higher accuracy than the shallow models. DL networks are superior to SVMs and normal neural networks because DL networks have more hidden layers than normal neural networks with one or two hidden layers. The DL network can provide training in a supervisory/unattended manner. The DL network performs automatic extraction of functions and does not require manual intervention, so they can save time because no functional engineering is required.

In this work [11], proposed a supervised PNN structure determination algorithm. An important feature of this supervised learning algorithm is to directly consider the requirements of network size and classification error rate in the process of determining the network structure. Therefore, the proposed algorithm often leads to a rather small network structure with satisfactory classification accuracy.

In their study [12] proposed an adaptive system based on the learning algorithm Q(0), which selects and calculates the smoothing parameters of the PNN method. It includes all possible PNN models. These models differ in the way they represent smooth parameters. The basis of the new method is a selection algorithm based on Q(0) to IRC adjustment parameters. The proposed method has been tested in six data sets and compared with CRF in conjugate gradient method training, the algorithm SVM classification gene expression programming (GEP), the method k-means, perceptronie neural network and learning vector quantization. In these three classification problems, at least one of the NPSC, PNNV, or PNNVC patterns formed in the proposed process can ensure the highest average accuracy. In four out of six, the PNNS was the second since the last data classification. This means that the representation of the smoothing parameters in terms of vectors and matrices contributes to higher CRF predictions. As can be seen, she trained with the conjugate gradient method CRF gave the best accuracy of data classification for six cases. Therefore, the suggestion of any alternative probabilistic training method in neural networks is justified.

The author in [13] explained in their work how to learn more about what is being used on the tweets to show in the polarity of messages and words. They provide detailed information about their third-party online training program, the key to their success. The result creates a new state-of-the-art on the phrases and is second in part of the messages. All kinds of tests, of their system were the first one on both subtasks. Their network guide includes the use of distance supervised data that focuses on (clearing noisy texts from tweets) to enhance the weight of the network passed from the unsupervised neural network. Therefore, their solution combines two basic aspects of the IR components: unsupervised learning of text representation (WEfrom neural language model) and study of well-managed data (Fig. 1).

Previous work [5] suggested the impact of preprocessing technique on the accuracy of sentiment analysis of different ML algorithms. In that work it was suggested that removing of emoticons and stopwords alongwith stemming and word vectorization can improve the efficiency and accuracy of ML algorithms which are Naive Bayes, Support Vector Machine and Maximum Entropy. Due to the sarcastic nature of tweets removing of the emoticons can affect the accuracy, while stopwords such as *'a', 'is', 'the', 'it'* are the highest in the frequency due to which the desired results cannot be obtained till the removal of it. In some tweets users using a single English character many times for example *@stellargirl: looooooooovvvvvveee my Kindle2*. In this work we will use PNN on the same dataset to improve the accuracy.

III. Material and Methods

Previous study [5] used some ML algorithms to enhance the accuracy of sentiment analysis, and in this study we will use some DL techniques for further incurring the accuracy.

A. Dataset

Twittratr[1] is an internet platform for Twitter's sentiment data, which using a series of negative and positive sentiments. For this work, we used the previous dataset of our research work which contains the Twittrat keyword list. This list includes 174 positive and 185 negative words [5]. For each tweet, we will count the number of negative and positive keywords that appears. The classifier returns a huge amount of polarities.

B. Preprocessing

Text can come in many forms, from single word lists to sentences and many paragraphs with special characters (such as tweets). As with any data science problem, a question can be raised that which steps should be taken to convert words into numerical values which can be understandable by the ML algorithms. Although this is not an exhaustive list, the preparation of the text is a complex art that requires the selection of the best tools, including data and questions. Many libraries and ready services are ready to help, but some may need to manually map terms and vocabulary. After preparing the dataset, supervised or unsupervised machine learning techniques can be used.

[1] http://twitrratr.com/

1) Cleaning data: Twitter is distinguished as a short messages; enclosure of URIs, usernames; special characters and topic markers. It often containing abbreviations and errors, some of these occurrence consist of linguistic noise, which makes make microblog part-of-speech tagging extremely difficult [14]. To avoid such a difficulty we removed emoticons, special characters and hashtages from our dataset.

2) Removal of stopwords: In today's world most of the data are available in textual form. Mostly this textual data congaing some words such as *"a", "the", "it", "as"* which are higher in frequency in every document and effecting the nature and accuracy of that document, if these high frequency words are not remove then they could interrupt the comparison calculation [15].

3) Stemming: In microbloging users often using some words with many alphabets, such as *@I loveooooooooo kindle*. Such kind of words can affect the efficiency of accuracy. To reduce a word to its proper stem is call stemming. The purpose of stemming is to find out the representative indexing form of a word by the purpose of truncation of affixes [16].

C. Word Embedding

WE is a natural language processing (NLP) technique which allows words or phrases to be mapped as vectors of real numbers. This process is important because many ML algorithms as well as deep neural networks require the input that should be vectorized and continues values vectors, as it cannot be done by strings of plain text. In [17], the author presented GloVe, a competitive set of pre-trained embeddings, suggesting that word embeddings was suddenly among the mainstream.

Logically, each feed-forward neural network which acquires words from a term as an input and embeds them as vectors into a lower dimensional space, and it then refine all through back propagation, essentially crop word embeddings as the weights of the first layer, referred as Embedding Layer.

The main dissimilarity between such networks and word2vec is complexity of its computational approach. The modernization and speedy growth in computational approaches improve its importance GloVe.

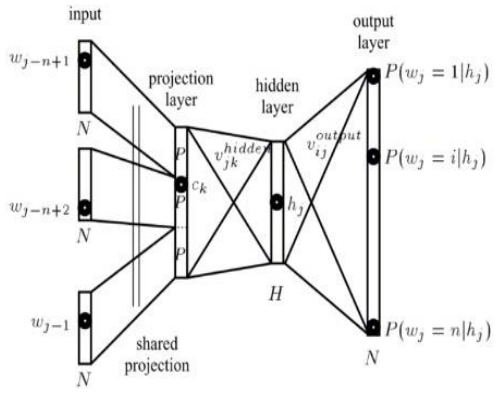

Fig. 1. Neural Language Model [18].

For illustration of words as vector an unsupervised algorithm was introduced by [17]. The training can be performed on combined global word-word co-occurrence statistics from a document. More specifically, the [17] stated that the relationship between the probabilities of the coexistence of two words (rather than their coexistence probabilities) is a factor that contains information, and therefore depends on the encoding of this information as a vector difference.

D. GloVe Algorithm

There was an enormous flow of articles regarding word vector representation after the publishing of Tomas Mikolov [4] work. Following that work, Stanford's Global Vector for Word Representation [17] was one of the best research work, which elucidated that why such algorithms and reformulated word2vec escalate as a particular nature of factorization for word co-occurrence matrices.

Below are the steps of the GloVe algorithm:

1) Collect word co-occurrence statistics in a form of word co-occurrence matrix X . Each element X_{ij} of such matrix represents how often word *i* appears in context of word *j*. Usually we scan our corpus in the following manner: for each term we look for context terms within some area defined by a *window_size* before the term and a *window_size* after the term. Also we give less weight for more distant words, usually using this formula:

$$\left(decay = \frac{1}{offset}\right) \quad (1)$$

2) Define soft constraints for each word pair: $\left(w_i^T w_j + b_i + b_j = \log(X_{ij})\right)$ where w_i - vector for the main word, w_j - vector for the context word, b_i, b_j are scalar biases for the main and context words.

$$J = \sum_{i=1}^{V} \sum_{j=i}^{V} f(X_{ij}) \left(w_i^T + b_i + b_j - (\log X_{ij})^2\right) \quad (2)$$

3) Define a cost function: Here f is a weighting function which helps us to prevent learning only from extremely common word pairs. The GloVe authors choose the following function:

$$f(X_{ij}) = \begin{cases} \left(\frac{X_{ij}}{x_{max}}\right)^2 & if X_{ij} < x_{max} \\ 1 & otherwise \end{cases} \quad (3)$$

E. Probabilistic Neural Network

A probabilistic neural network (PNN) is a supervised network, which can be commonly used in decision making and classification problems [19]. PNN was firstly introduced by [20]. The immediate and easy training makes PNN's main advantage, and can be used for real-time as well [6].

F. Architecture of PNN

A PNN is an completion of a statistical algorithm, called kernel discriminate analysis in which the procedures are structured into a multi-layered feed-forward network with four layers, i.e. input layer, pattern layer, summation layer, and output layer.

- Input layer

This layer distributes the N number of input nodes to the neurons and every neuron symbolizes a predictive variable in this layer. According to the categorical variables, the N-1 neuron can be applicable on N number of categories. It normalizes the series of the values by deducting the medium and dividing by the inter-quartile range. After that the input neurons provides the values to every neurons in the hidden layer.

- Pattern layer

Pattern layer containing the Gaussian functions and for every case of training dataset the layer hold one neuron. Along-with the target values, it also stores the predictive variables values.

- Summation layer

The summation layer performs a sum operation of the outputs from the second layer for each class.

- Output layer

The output layer performs a vote, selecting the largest value. The associated class label is then determined.

G. PNN Algorithm

In Fig. 2, we exemplify the architecture of PNN with hidden layers. The sum of pattern nodes is the same of total of training sample. The synaptic weight $w_{ij}^{(P)}$ in the pattern to input is:

$$w_{ij}^{(P)}=x_i^{(j)} \tag{4}$$

Where, $x_i^{(j)}$ represents the i^{th} input node of the j^{th}sample at the input layer. And for the weight between pattern and summation layers $w_{jk}^{(S)}$ can be represent as:

$$w_{jk}^{(S)} = \begin{cases} 1 & if T_k^{(j)}=1 \\ 0 & otherwise \end{cases} \tag{5}$$

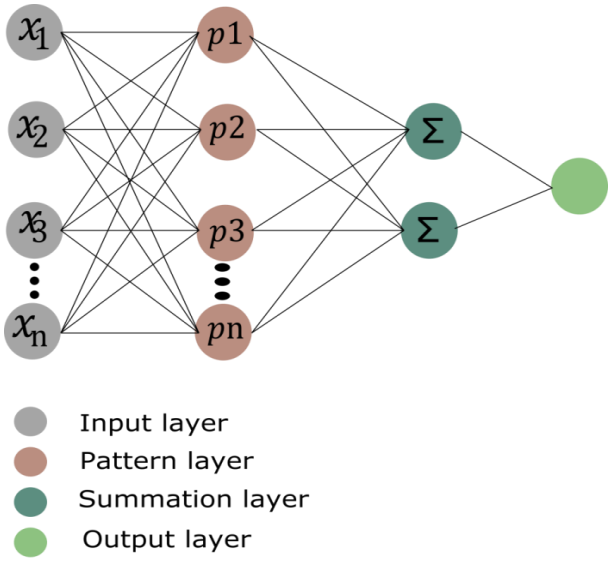

Fig. 2. Architecture of PNN.

Here the T_k^j value is 1 since the association of sample j with class k and otherwise 0s.

After the training procedure as shown in (4) and (5), the input classification pattern can be commenced as under:

$$n_j^P = \sqrt{\sum_i \left(w_{ij}^{(P)} - x_i\right)^2} \tag{6}$$

The pattern out can be calculated as:

$$P_j=\exp\left(-\frac{n_j^{(P)}}{2\sigma^2}\right) \tag{7}$$

Here σ is stander deviation of Gaussian distribution, which is a smoothing parameter corresponding representation.

The summation layer every single node symbolizes an individual class and can be expressed as:

$$S_k=\frac{1}{\sum_j w_{jk}^{(s)}}\sum_j w_{jk}^{(s)} . P_j \tag{8}$$

The input vector classifies into a precise single class by the output layer, if the output value is maximum from the input node at the summation layer:

$$y = \arg max_x S_k \tag{9}$$

H. Our Proposed Model

In our proposed model we used the dataset of 359 documents. The twitter data often containing *urls*, special characters and emoticons, besides this it contains unwanted words such as, *'i'*, *'the'*, *it*, *'a'* which mostly higher in the frequency and can affect the accuracy. As well as it includes words like *'looooooooovvvveeee'*, *'sooooooo'*. Table I shows the raw twitter data. Since we go further, we applied the preprocessing i.e. cleaning data, removal of stopwords and stemming to clean the dataset, which can be seen in Table II.

TABLE I. RAW TWITTER DATA

Stellargirl I loooooooovvvvvveee my Kindle2. Not that the DX is cool, but the 2 is fantastic in its own right.
I hate aig and their non loan given asses. :(
Jquery is my new best friend. $$$
i srsly hate the stupid twitter API timeout thing, soooo annoying!!!!!
How can you not love Obama? He makes jokes about himself.
@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D

TABLE II. TWITTER DATA AFTER CLEANING

Stellargirl love Kindle2. Not DX cool, but 2 fantastic right
hate aig their non loan given asses
Jquery new best friend
srsly hate stupid twitter API timeout thing, so annoying
how not love Obama makes jokes about himself.
A, that's bummer. should got David Carr of Third Day

word	output_vector
pills	[-0.001236052019521594,7.121161906979978E-4,0.0023646559566259384,...
prepare	[-0.0035068343859165907,0.0034635020419955254,1.5873336815275252E-...
half	[-0.003999084234237671,0.004989865235984325,0.001815770985558629,...]
*nice	[-0.003707629395648837,1.896002795547247E-4,-6.314286729320884E-4,...]
waster	[0.0010279074776917696,-0.0012603802606463432,0.002111770678311158...
idol	[0.003927727695554495,-0.0027154632844030857,-3.8873442099429667E-...
clinton	[-0.0015378474490717053,-0.0012981300242245197,0.0019235379295423 6...
your	[0.0014543861616402864,-4.632280324585736E-4,8.334809099324048E-4,...]
childs	[-0.0032335547730326653,1.6314862295985222E-4,-3.8727818173356354E...
without	[7.865516236051917E-4,1.8350420577917248E-4,-0.0017648295033723116...

Fig. 3. Word vectors.

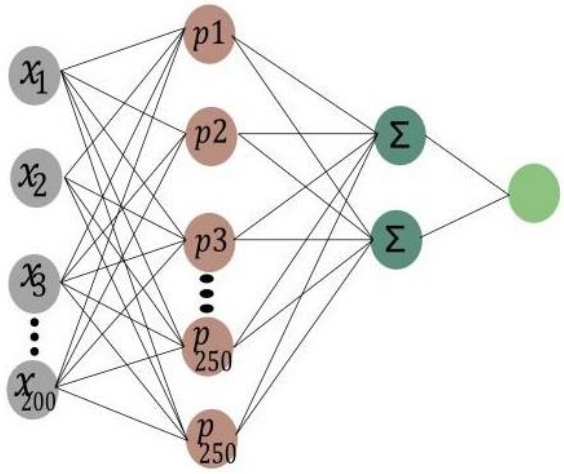

Fig. 4. The input and output nodes of our proposed model.

In the next step, we used WE [4] to convert the strings data into word vector implementing the batch size 1000 and layer size 200. In the following Fig. 3 words and their vectors can be seen.

Later on in the next step, we split the dataset into 70% train and 30% of test datasets, which divided it into 251 and 108 documents simultaneously.

In the last step, we applied PNN [7] on our dataset. As the layers size were 200 in our third step, the input nodes will also be 200 and pattern nodes will be 250 as well as showmen in Fig. 4. We kept the input and pattern layers sizes 200 and 250 simultaneously since the accuracy were at highest at this level.

IV. Results and Discussion

In our previous work [5], we applied preprocessing techniques along-with removing of emoticons on SVM, NB and MaxE algorithms and we got the accuracy results 81.63%, 91.81% and 88.27%, respectively on a dataset of 250 documents which can be seen in Table III.

TABLE III. Classifiers Accuracy after Preprocessing

Algorithms	Accuracy after Preprocessing
SVM	81.63%
NB	91.81%
MaxE	88.27%
PNN	98.00%

TABLE IV. PNN Accuracy Table

Algorithms	Accuracy after Preprocessing
SVM	81.63%
NB	91.81%
MaxE	88.27%
PNN	98.00%

TABLE V. Positive and Negative Predictions

	True positive	False Positive	True Negative	False Negative	F-Measures	Accuracy	Cohen's Kappa
Negative	120	0	127	3	0.987	-	-
Positive	127	3	120	0	0.988	-	-
Overall	-	-	-	-	-	0.988	0.975

To improve the accuracy we used PNN and WE which enhanced the results and can be seen in Table IV.

In the above Table IV, we can see the accuracy improved to 98% as compared to our previous work [5]. We can see in Table V that after applying Word Embedding and PNN on the dataset of 250 twitter documents we get 120 negative and 127 positive prediction document class and have only 3 wrong classified class as well, shown in Fig. 5.

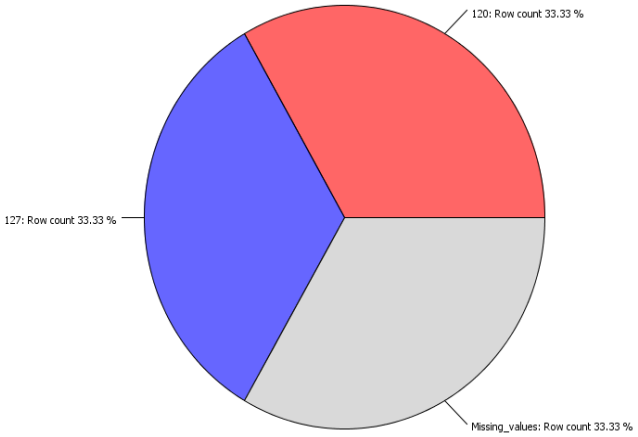

Fig. 5. Shows the row counts of positive, negative and missing values.

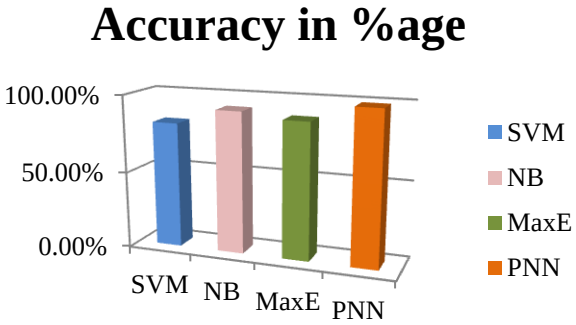

Fig. 6. Accuracy of SVM, NB, MaxE and PNN in per cents.

In Fig. 6, it can be seen that the accuracy of PNN is higher than other 3 algorithms, namely, SVM, NB and MaxE.

V. Conclusion

As in our previous work [5], we applied the preprocessing steps to improve the results. We observed in this work that as compared to Naive Bays, SVM and MaxE, the WE have tremendous effects on PNN. As compared to traditional techniques, PNN has higher accuracy and fast training time. Our investigational results on the basis of hybrid combination of WE and PNN could be a probable solution for enhancing the performance and accuracy of classification and as well decreasing the training time.

References

[1] L. Deng, G. E. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," 2013 IEEE Int Conf Acoust Speech Signal Process, pp. 8599–8603, 2013.

[2] Q. T. Ain et al., "Sentiment Analysis Using Deep Learning Techniques: A Review," Int J Adv Comput Sci Appl, vol. 8, no. 6, pp. 424–433, 2017.

[3] T. J. O'Shea and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," vol. 7731, no. c, pp. 1–13, 2017.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," Arxiv, pp. 1–12, 2013.

[5] S. Alam and N. Yao, "The impact of preprocessing steps on the accuracy of machine learning algorithms in sentiment analysis."

[6] D. F. Specht, "Probabilistic neural networks," Neural Netw., vol. 3, no. 1, pp. 109–118, 1990.

[7] S. G. Wu, F. S. Bao, E. Y. Xu, Y.-X. Wang, Y.-F. Chang, and Q.-L.Xiang, "A Leaf Recognition Algorithm for Plant Classification Using Probabilistic Neural Network," pp. 1–6, 2007.

[8] A. Savchenko, "Probabilistic Neural Network with Complex Exponential Activation Functions in Image Recognition using Deep Learning Framework," no. 14, pp. 1–14, 2017.

[9] R. Ghosh, K. Ravi, and V. Ravi, "A novel deep learning architecture for sentiment classification," 3rd IEEE Int Conf Recent Adv Inf Technol, pp. 511–516, 2016.

[10] Y. Dang, Y. Zhang, and H. Chen, "A lexicon-enhanced method for sentiment classification: An experiment on online product reviews," IEEE Intell Syst, vol. 25, no. 4, pp. 46–53, 2010.

[11] K. Z. Mao, K. C. Tan, and W. Ser, "Probabilistic neural-network structure determination for pattern classification," IEEE Trans Neural Netw., vol. 11, no. 4, pp. 1009–1016, 2000.

[12] M. Kusy and R. Zajdel, "Probabilistic neural network training procedure based on Q(0)-learning algorithm in medical data classification," Appl Intell, vol. 41, no. 3, pp. 837–854, 2014.

[13] A. Severyn and A. Moschitti, "Twitter Sentiment Analysis with Deep Convolutional Neural Networks," Proc 38th Int ACM SIGIR Conf Res Dev Inf Retr - SIGIR 15, pp. 959–962, 2015.

[14] L. Derczynski, A. Ritter, S. Clark, and K. Bontcheva, "Twitter part-of-speech tagging for all: Overcoming sparse and noisy data," Proc Recent Adv Nat Lang Process, no. September, pp. 198–206, 2013.

[15] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of duplicate defect reports using natural language processing," Proc - Int Conf Softw Eng, pp. 499–508, 2007.

[16] Y. Kadri and J. Y. Nie, "Effective stemming for Arabic information retrieval," Proc Chall Arab NLPmt Int Conf Br Comput Soc, pp. 68–74, 2006.

[17] J. Pennington, R. Socher, and C. D. Manning, "GloVe : Global Vectors for Word Representation."

[18] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy Layer-Wise Training of Deep Networks," Adv Neural Inf Process Syst, vol. 19, no. 1, p. 153, 2007.

[19] F. Vicino, "The Probabilistic Neural Network," vol. 33, no. 2, pp. 335–352, 1998.

[20] D. F. Specht, "Probabilistic neural networks for classification, mapping, or associative memory," IEEE Int Conf Neural Netw., pp. 525–532 vol.1, 1988.