IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-8, Aug 2018 E-ISSN: 2347-2693

A Comprehensive Study on Sentiment Analysis Using Deep Forest

Krishna Priya S[1*], Shaksham Kapoor[2], Kavita S Oza[3], R.K. Kamat[4]

[1]Dept. of CSE, Shivaji University, Kolhapur, India
[2]Imarticus Learning, Pune, India
[3,4]Dept. of CSE, Shivaji University, Kolhapur, India

[*]*Corresponding Author: krishnapriyapradeep01@gmail.com*

Available online at: www.ijcseonline.org

Accepted: 12/Aug/2018, Published: 31/Aug/2018

Abstract— In this paper, we study the problem of binary sentiment classification on a set of polar movie reviews. There are many models which have achieved state of the art performance, but one has to deal with the problem of tuning a large number of hyper-parameters. With the addition of the deep forest model as proposed by Zhi-Hua and Ji Feng, the number of hyper-parameters to be tuned is less and the architecture is still able to perform well. The goal of this paper is to use Word2Vec, FastText and Doc2Vec for creating word vector representation of the reviews which are then trained on a deep forest model. In order to further enhance the performance, the trained model is further trained on a different set of classifiers and as a result, a significant improvement in performance was noticed.

Keywords—DeepForest, WordEmbeddings, Word2Vec, FastText, Doc2Vec, SVM

I. INTRODUCTION

Sentiment analysis is the process of extracting opinions, emotions, attitude or sentiment from a document or a text using natural language processing, computational linguistics, and text analysis. Sentiment analysis allows organizations to track how acceptable is the brand, likes and dislikes of people, Company reputation, etc [1]. But now mostly this is restricted to just basic sentiment analysis and metrics that are based on counts. Because of this many time, we are missing out on the deep insights of the data. There are chances when different users have a different opinion about the same product or a movie, but finding such relevant connections is not easy [2]. Much research is going on in improving the natural language understanding by machines. The advancements in deep learning techniques where we try to mimic human brain help in improving the sentiment mining task considerably. Creative use of AI techniques will bring a remarkable improvement in analyzing text.

In order to tackle complicated learning tasks, it is likely that learning models have to go deep. Current deep models are mostly based on neural networks. The deep neural network models deliver remarkable performance based on three features- layer-by-layer processing, feature transformation, and complexity of model [3]. The paper makes use of deep forest which has all these features and adds the number of hyperparameters to be tuned are remarkably less, which works well with small-scale data, parallelize well and better theoretical analysis possible compared to deep neural networks [4].

This paper proposes a sentiment analysis model by making use of the power of deep representation of data, the word embedding, and the classical machine learning classifiers. The organization of the rest of the paper is as follows: Section I states the importance of sentiment analysis and how it is being used nowadays by almost every organization, Section II discusses the literature survey. Section III explains the methods used in the experiments followed by Section IV where details of experiments, methodology, datasets used and results are explained. Finally, in Section V states the conclusion and future scope followed by references.

II. RELATED WORK

Akhtar, M. S. et al trained pre-trained, autoencoder-based, financial word embeddings and lexicon features on various deep learning models which were developed based on Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). A classical supervised machine learning model based on Support Vector Regression (SVR) is combined with the developed deep learning models to create an ensemble which was then evaluated on a benchmark dataset of SemEval-2017 shared task on financial sentiment analysis [5].

Akhtar, M. S et al developed sentiment analysis model considering resource-poor languages. They proposed a hybrid deep learning architecture using CNN and

SVM. CNN is used leaned Sentiment embedded vectors and then augmented to a set of optimized features and passed to SVM. It is then evaluated on two benchmark English datasets [6].

Abdul-Mageed, M., & Ungar, L build a very large dataset for fine-grained emotions and develop deep learning models on it using twitter data. They achieve a new state-of-the-art performance on 24 fine-grained types of emotions. They also extended the classification to eight primary emotion dimensions situated in psychological theory of emotion [7].

Cho et al proposed a novel neural network model called RNN Encoder–Decoder that consists of two recurrent neural networks (RNN). One RNN encodes a sequence of symbols into a fixed length vector representation, and the other decodes the representation into another sequence of symbols. The encoder and decoder of the proposed model are jointly trained to maximize the conditional probability of a target sequence given a source sequence. The performance of a statistical machine translation system is empirically found to improve by using the conditional probabilities of phrase pairs computed by the RNN Encoder–Decoder as an additional feature in the existing log-linear model. Qualitatively, they showed that the proposed model learns a semantically and syntactically meaningful representation of linguistic phrases [8].

Collobert, R et al proposed a unified neural network architecture and learning algorithm that can be applied to various natural language processing tasks including part-of-speech tagging, chunking, named entity recognition, and semantic role labeling. This versatility is achieved by trying to avoid task-specific engineering and therefore disregarding a lot of prior knowledge. Instead of exploiting man-made input features carefully optimized for each task, our system learns internal representations on the basis of vast amounts of mostly unlabelled training data. This work is then used as a basis for building a freely available tagging system with good performance and minimal computational requirements [9].

Guan, Z et al proposed a novel deep learning framework for review sentiment classification which employs prevalently available ratings as weak supervision signals. The framework consists of two steps: (1) learn a high-level representation (embedding space) which captures the general sentiment distribution of sentences through rating information; (2) add a classification layer on top of the embedding layer and use labeled sentences for supervised fine-tuning. Experiments on review data obtained from Amazon show the efficacy of our method and its superiority over baseline methods [10].

Dou, Z. Y et al worked-on document-level sentiment classification problem by conducting experiments on a deep memory network which could capture the user and product information at the same time. They conducted experiments on IMDB and Yelp datasets [11].

A. Hassan and A. Mahmood use CNN and LSTM on pre-trained vectors. They used LSTM to capture long-term dependencies and to reduce the loss of detailed local information in place of pooling. The model is validated on datasets IMDB and Stanford Sentiment Treebank [12].

N. Zainuddin and A. Selamat describe experimental results that applied Support Vector Machine (SVM) on benchmark datasets to train a sentiment classifier. N-grams and different weighting scheme were used to extract the most classical features. It also explores Chi-Square weight features to select informative features for the classification. The experimental analysis reveals that using Chi-Square feature selection may provide a significant improvement on classification accuracy [13].

III. METHODOLOGY

In our work, we have used different word vector models to generate the vectors for words as the input. The task is to classify reviews either positive or negative, but many machine learning algorithms and deep learning architecture are not able to process text directly. In order to deal with this problem, we need to convert our reviews into word vectors which can then be passed to the deep forest architecture.

Deep Forest

Deep models are always built upon neural networks. Deep Forest is an NN style, deep model. Neural networks use a layer by layer processing that makes use of modules characterized by parameterized differentiable and nonlinear modules. But not all properties in the neural networks are differentiable or can be modeled as differentiable. Deep Forest is an attempt to attain deep learning with non-differentiable modules

In this paper, we use Cascade Forest (gcForest) for constructing deep forest. It can be considered as an enhanced version of ensemble methods which uses a decision tree ensemble as a cascade structure for representation learning. The multi-grained scanning feature enables gcForest to be contextually aware. gcForest can work well on small-scale data and has much fewer hyper-parameters to tune

To process the raw features layer-by-layer, gcForest employs a cascade structure, as illustrated in Figure 1, where each level of cascade receives information processed by its preceding level and outputs its processing result to the next level [14].

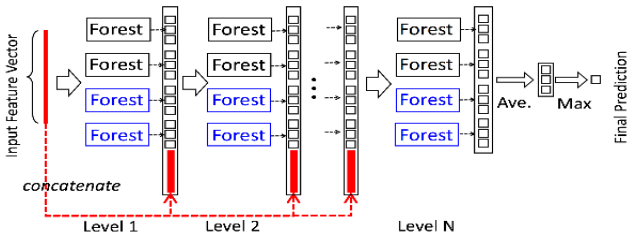

Figure 1: Deep Forest Architecture

Each level is an ensemble of decision tree forests. At the leaf node, the percentage of different classes of training examples are counted where the particular instance falls. Then the average is taken across all trees in the same forest. In this way, each forest will produce the estimate of class distribution as in Figure 2[14].

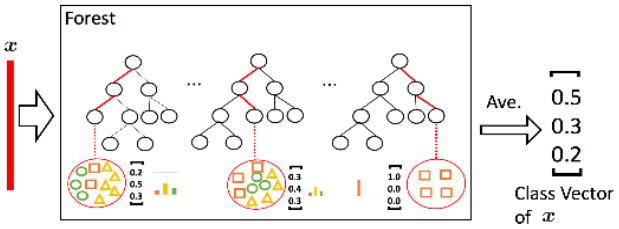

Figure 2: Deep Forest Predicting Outcome

Different marks in leaf nodes imply different classes. These different classes have an estimated class distribution which forms a class vector. The original feature vector and the formed class vector is then concatenated to be given as input to the next level of the cascade. Cascade forest is enhanced with a procedure of multi-grained scanning. Extracted feature vectors are then used to generate class vectors. A completely-random tree forest and a random forest are then trained using the instances extracted from the same size of windows and the class vectors then generated are concatenated as transformed features.

To train the further cascades, the transformed feature vectors are combined with the generated class vector. This procedure is repeated till convergence of validation.

Word Embeddings

It is a parameterized function which map words into n-dimensional vector space where n can be (100, 200, 300, 500 – generally) or in simplistic terms, they are the numeric representation of the given text "embedded" into vector space. For example: -

$$W('dog') = (0.2, 0.3, -0.1, ...)$$

$$W('car') = (0.1, -0.5, -0.7, ...)$$

The learned meaningful vector representations (W) are then used to perform some task. To get an intuition for word embedding, let's visualize them using t-SNE in Figure 3 [15].

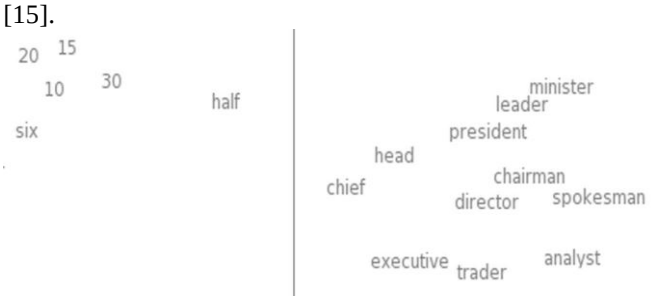

Figure 3: t-SNE Visualization of Word Embeddings

We can see that similar words (20, 15, 10, 30), (leader, minister, president) are close to one another. This kind of "map" seems intuitive where words with similar meaning tend to lie close to one another in vector space. Another way to look at this is to look for words which have closest embeddings in vector space to the given word. See Figure 4 [15],

FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS

Figure 4: Words having word embeddings close to a given word.

This is a remarkable property of word embeddings; words with similar meaning will have similar vectors. They can be classified into two categories: -

a. Frequency-based embeddings: Count Vectors, TF-IDF, Co-occurrence vector
b. Prediction based embeddings: Word2Vectors (Continuous Bag of Words and Skip gram models), Glove etc.

Word2Vec

The word2vec architecture as proposed by Mikolov et al. in 2013, is a feed forward, fully connected neural network which takes as input a corpus of words and outputs a vector space of high dimension (100, 200, 300 or 500 generally) [16]. Each unique word present in the corpus is mapped to its own vector representation in the vector space with words having a similar context are located in close proximity to one another.

Suppose we have a movie review like "Combining conventions Western Gothic horror often directed art movie", let's try to understand the context word by word. Consider the word 'Gothic' which is surrounded by many other words and taking a context window of size say 3, we can see that if we use forward context, then the word 'Gothic' depends on the context 'Combining conventions Western'. If we use backward context, then the word 'Gothic' depends on the context 'horror often directed' and if we use the center context, then the word 'Gothic' depends on the context 'conventions Western horror'. So, the same word can have three different contexts depending upon the position and that's what word2vec takes into consideration.

Using context and a word belonging to that context, it tries to predict the next word from that context (Continuous Bag of Word model) or from the target word, try to predict the context from which it came (Skip-Gram model). In this paper, we have used both the models for creating effective word representations of three hundred dimensions for the

movie reviews and using those to predict the sentiment of the review.

Let's take a look at CBOW model, suppose a window is sliding over the text and 'horror' is the word in focus with four words preceding it (Combining conventions Western Gothic) and four words following it (often directed art movie) making the context. Then, the input to the neural network architecture is the one-hot encoded form of each context word. Given the input context word, the conditional probability of obtaining the focus word is maximized with regards to the weights. See Figure 5 [17],

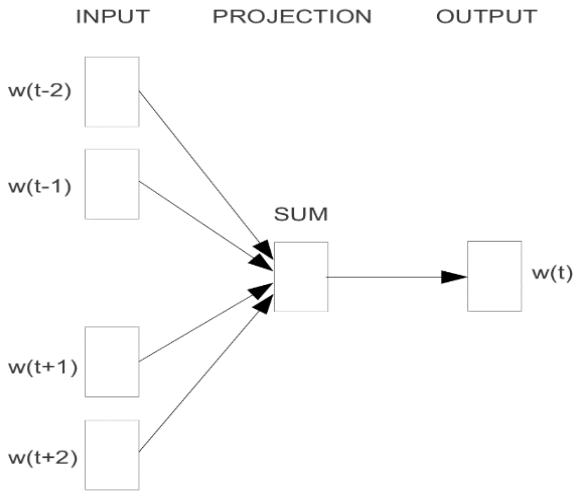

Figure 5: CBOW Model Architecture

In Skip-Gram model, the input is the focus word and the context words are the output. The aim is to maximize the log probability of any context word given the current word in focus. The objective function is defined as

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\,!=\,0}\log\left(p\left(w_{t+j}|w_t\right)\right), \quad \text{..... (1)}$$

where c is the size of context window

Here,

$$p(w_{t+1}|w_t) = \exp\left(v_{w_0}v_{wI}^{T\prime}\right) \Big/ \sum_{k=1}^{V}\exp\left(w_i^T w_k{}'\right) \quad \text{..... (2)}$$

is the softmax function

The problem with this expression is that every time we move over the context, we have to recompute the normalization term in the denominator for every word W which can be of very high order. To deal with this, two strategies were proposed: Hierarchal SoftMax and Negative sampling. In this paper, we have experimented using negative sampling. Figure 6 shows the Skip-Gram architecture [17]

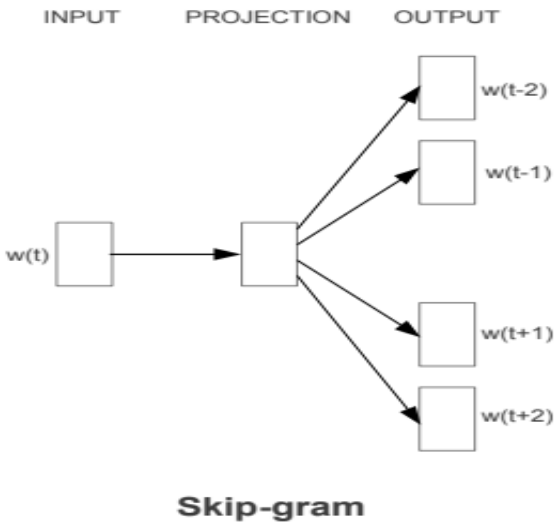

Figure 6: Skip Gram Model Architecture

FastText

FastText created by Facebook is a library for learning word embedding. It treats every single word as the smallest unit whose vector representation is to be found by n-grams of character. Finding the vector representation for rare words can be easily done. Words not present in the dictionary can also be converted to vectors using n-gram character concept [18]. It takes care of the internal structure of words. Extremely useful for morphologically rich languages.

Doc2Vec

The aim of doc2vec or paragraph vectors is to assign documents with labels. It is an extension to word2vec that learns to correlate labels with words rather than words with other words regardless of its length. The concept used is simple: use the word2vec model and add another vector called Paragraph ID, see Figure 7, [19]

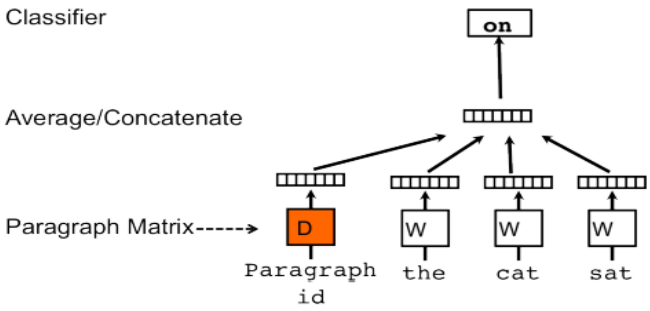

Figure 7: PV-DM Model

The additional feature vector is documented unique, so while training word vectors and document vectors both are trained. Finally, we get a numeric representation of the document. This model is called Paragraph Vectors – Distributed

Memory (PV-DM). The memory here represents what is missing from the current context or what is missing as a topic of the paragraph. The document vector used is intended to represent the concept of the document.

Similar to a continuous bag of word model for word2vec, doc2vec has a distributed bag of word model (DBOW) which is faster than word2vec (CBOW) model and saves memory since word vectors need not be stored. See Figure 8, [19]

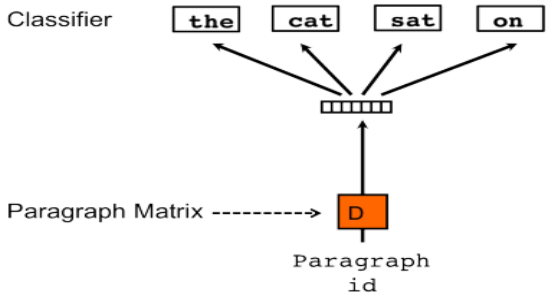

Figure 8: PV-DBOW Model

The movie reviews used in this paper are present in paragraph format where each review is either negative or positive. Using doc2vec, a document vector is generated for every review (for both train and test set). The document vector is concatenated with the word vector to predict the next word in a context. While training a fixed length context window is sampled randomly from a random paragraph which is used to calculate the error gradient in order to update the parameters. Suppose we have N paragraphs each mapped to p dimensions and M words mapped to q dimensions, the total number of parameters (excluding SoftMax parameters) are;

$$N * p + M * q \quad \ldots\ldots (3)$$

The paragraph vectors thus obtained after training are then passed to deep forest architecture. While making predictions on the testing set, an inference step is performed to compute the paragraph vector for the new testing movie reviews.

SVM

Support Vector Machine is a supervised machine learning algorithm which is used for both regression and classification purposes. In classification, the aim of the algorithm is to output an optimal hyperplane which will categorize the inputs into different categories. In 2-D the hyperplane is a line dividing the plane into two parts, where each part represents its own class. See Figure 9, here purple observations represent one set of class and yellow observations represent another set of class and both the classes can be easily separated with a line [20].

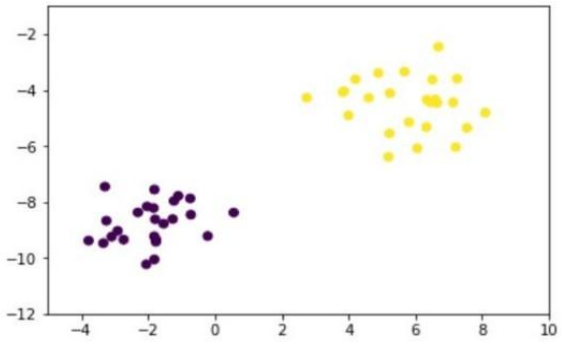

Figure 9: Binary Classification Dataset

An optimal hyperplane has maximum margin, here margin means the distance between the nearest observation from either class and the dividing hyperplane. Let's write down the general equation of a hyperplane in 2-D;

$$w^T.\ x_i + b = 0\ , \forall\ x_i\ , where\ w^T.x_i\ is\ the\ inner\ product. \quad \ldots\ldots (4)$$

Also, (w, b) are weight vector and bias vector respectively. Now, for every input vector x_i either:

$$w^T.\ x_+ + b \geq 1, \qquad for\ x_+\ having\ the\ class\ 1 \quad \ldots\ldots (5)$$

In this case, the input points will be assigned to the positive side of the hyperplane (yellow points) and for the second case;

$$w^T.\ x_- + b \leq -1, for\ x_-\ having\ the\ class - 1 \quad \ldots\ldots (6)$$

The input points will be assigned to the negative class of the hyperplane (purple points). See Figure 10, [20]

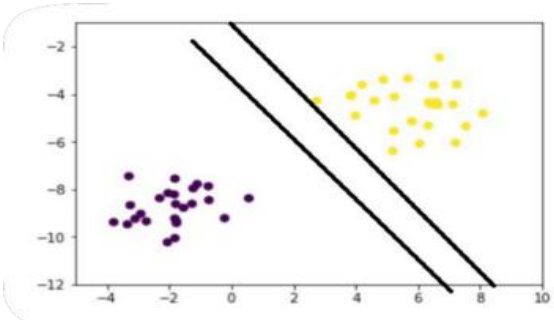

Figure 10: Separating Classes Linearly

Multiplying equation 2 by y_i (which is always -1 for this equation), we get;

$$y_i(w^T.x_- + b) \geq y_i(-1) \quad \ldots\ldots (7)$$

This implies,

$$y_i(w^T.x_- + b) \geq 1, \forall\ x_-\ belonging\ to\ class - 1 \quad \ldots\ldots (8)$$

Combining equations 1 and 3, we get;

$$y_i(w^T.x_i + b) \geq 1, \quad \forall x_i \text{ where } 1 \leq i \leq n \quad \ldots. (9)$$

The equation thus formed is the mathematical equivalent of equations 5 and 6. Normalizing the weight vector with respect to equations 5 and 6 and solving them algebraically, the margin is given by;

$$\frac{w}{||w||}.(x_+ - x_-) = \frac{w^T(x_+ - x_-)}{||w||} = \frac{2}{||w||} = margin \quad \ldots.. (10)$$

Since, w is the only variable in the equation, which is indirectly proportional to margin, thus the optimization problem for maximizing the margin can thus be formulated as;

$$max_w \frac{2}{||w||} \quad \ldots\ldots (11)$$

Subject to,

$$w^T.x_i + b \begin{matrix} \geq 1 \\ \leq -1 \end{matrix} \text{ if } \begin{matrix} y_i = 1 \\ y_i = -1 \end{matrix} \text{ for } 1 \leq i \leq n \quad \ldots\ldots\ldots (12)$$

Or equivalently,

$$min_w ||w||^2 \text{ subject to } y_i(w^T.x_i + b) \geq 1, \quad 1 \leq i \leq n \quad \ldots. (13)$$

Figure 11 summarizes the whole process visually [21]

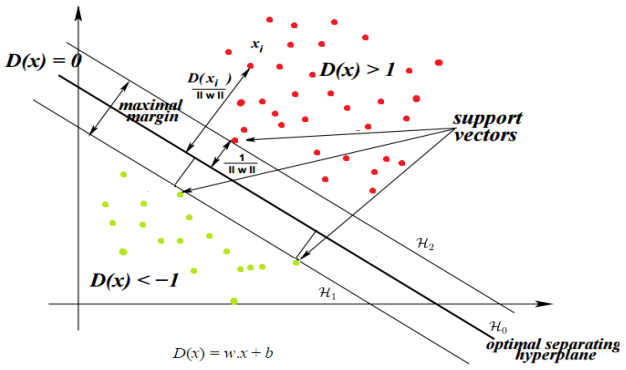

Figure 11: Optimal Hyperplane and Margin

This was a linearly separable case but there are many cases where the data cannot be classified perfectly by a linear boundary. In such cases, the aim of the algorithm is to maximize the margin as well as to minimize the misclassifications; this is achieved by introducing an additional parameter ξ (also called slack variable). The optimization problem thus becomes

$$min_{\substack{w \in R^d \\ \xi_i \in R^+}} ||w||^2 + C\sum_{i=1}^{n} \xi_i \quad \ldots. (14)$$

Subject to,

$$y_i(w^T.x_i + b) \geq 1 - \xi_i, \quad for \ 1 \leq i \leq n \quad \ldots\ldots. (15)$$

Here, C is a regularization parameter. A large value for C allows narrow margin and a small value for C allows a large margin. When the problem becomes multidimensional (4 or more), it is not always possible to separate the data using line (2-D) or a plane (3-D).

In such cases, SVM uses a kernel function to transform the data into the higher dimensional feature space where the data is linearly separable.

IV. EXPERIMENTS

Dataset

This paper has been implemented using a labeled dataset containing 50000 polar IMDB movie reviews provided by Stanford University [22] [23]. It is a binary classification sentiment dataset with 0 being assigned to negative reviews and 1 being assigned to positive reviews.

An additional 50000 unlabelled movie reviews have also been provided. In order to create word vectors, we have used 25000 training reviews, 50000 unlabelled reviews, and an additional 2000 movie reviews. Table 1 describes the dataset

Table 1: Dataset Description

	Negative Reviews	**Positive Reviews**	**Features**
Training set	12500	12500	(review, sentiment)
Testing set	12500	12500	(review, sentiment)
Unlabelled set	NA	NA	(review)
Additional set	NA	NA	(review)

Note: sentiment – 0 = negative and 1 = positive, NA = Not Applicable.

System Design

The data obtained is messy and needs to be cleaned before modeling, for this unlabelled review, training reviews and additional review sets have been taken as input and extensive text pre-processing is performed. The steps involved are divided into two parts: -

a. Punkt Tokenizer – split every paragraph review into a set of sentences.

b. Word2Vec Utility Function – the sentences thus obtained are passed for pre-processing into a function which defined in the Word2Vec UtilityFunction class. The steps involved are: -

I) Removing HTML Markup using Beautiful Soup
II) URL Links (‘https:’ pattern) removal
III) URL Links (‘www.’ Pattern) removal
IV) ‘@’ username or other mentions removal
V) Lowercase the review and negation handling
VI) Removing numbers, hashtags and special characters
VII) Stop words removal (optional) because the broader context of the sentence is considered to produce word vectors.
The pre-processed reviews are converted into a string of words which are then used to create word vectors (Word2Vec and FastText) of three hundred dimensions, having a context window of ten words, a minimum count of five and a threshold of 0.001 for downsampling higher frequency words. For Doc2Vec the minimum count is one rest everything is same. Figure 11 and Figure 12 summarizes the pipeline

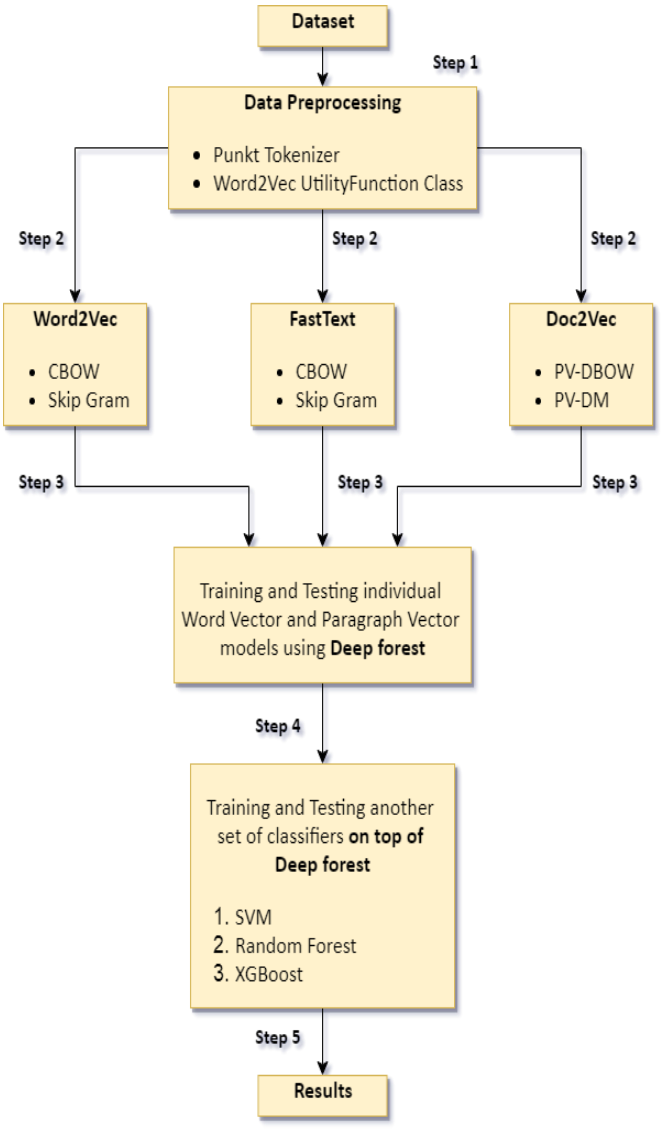

Figure 12: Pipeline

Step 1: Data Cleaning
Step 2: Creating Word vectors and Paragraph Vectors
Step 3: Modeling - I
Step 4: Modeling - II
Step 5: Testing results and Comaprison

Figure 13: Steps Description

V. Results and Discussions

The experiments have been performed in Python using sklearn, gensim, nltk, and gcForest libraries [24].

1. Experiment I

This section shows the results for Continuous-Bag-Of-Words (CBOW) model (Word2Vec, FastText) and Distributed Bag-Of-Words model (Doc2Vec). Firstly, the data vectors for the training set are trained using the deep forest architecture. The results of which are summarized in Table 2. Secondly, the trained deep forest model is trained further on a different set of classifiers (SVM, Random Forest, XGBoost) with SVM showing a slight increase in performance. Table 3 summarizes the result.

Table 2: Testing Set Accuracy

Word 2 Vector - CBOW	87.90 %
Fast text – CBOW	86.99 %
Doc 2 Vector – PV_DBOW	73.98 %

Table 3: Testing set accuracy using Deep Forest + SVM

Word 2 Vector - CBOW	88.02 %
Fast text - CBOW	87.09 %
Doc 2 Vector – PV_DBOW	74.76 %

2. Experiment II

This section shows the results for Skip-Gram (SG) model (Word2Vec, FastText) and Distributed Memory (PV-DM) model (Doc2Vec). Same procedure (as Experiment I) is applied on the training set. Table 4 and Table 5 summarizes the results.

Table 4: Testing set accuracy

Word 2 Vector - SG	88.26 %
Fast text – SG	87.88 %
Doc 2 Vector – PV_DM	88.96 %

Table 5: Testing set accuracy using Deep Forest + SVM

Word 2 Vector – SG	88.49 %
Fast text – SG	88.26 %
Doc 2 Vector – PV_DM	**89.22 %**

The various experiments performed showed that Skip-Gram and Distributed Memory models in general performed better than Continuous-Bag-Of-Words.

VI. CONCLUSION AND FUTURE SCOPE

In this paper, we experimented with different word vector models using the deep forest. We further tried to enhance the accuracy of our model by training the deep forest architecture on a set of classifiers.

In future, a lot of experimentation can be done by passing word vectors of varying dimensionality, improving text pre-processing steps like taking ambiguous tweets into account and using a multi-grained feature of the deep forest. One more aspect which can be explored is hyperparameter tuning.

REFERENCES

[1] Tulsi Jain, Kushagra Agarwal, Ronil Pancholia, "*Sentiment Analysis Based on a Deep Stochastic Network and Active Learning*", International Journal of Computer Sciences and Engineering, Vol.5, Issue.9, pp.1-6, 2017.

[2] M. M. Sutar, T. I. Bagban, "*Applying Sentiment Analysis to Predict Rating and Classification of Text Review*", International Journal of Computer Sciences and Engineering, Vol.6, Issue.7, pp.173-178, 2018.

[3] Static.ijcai.org. (2018). [online] Available at: http://static.ijcai.org/proceedings-2017/0497.pdf [Accessed 13 Jun. 2018].

[4] Arxiv.org. (2018). [online] Available at: https://arxiv.org/pdf/1702.08835.pdf [Accessed 13 Jun. 2018]

[5] Akhtar, M. S., Kumar, A., Ghosal, D., Ekbal, A., & Bhattacharyya, P. (2017). A multilayer perceptron based ensemble technique for fine-grained financial sentiment analysis. In Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP 2017).

[6] Akhtar, M. S., Kumar, A., Ekbal, A., & Bhattacharyya, P. (2016). A hybrid deep learning architecture for sentiment analysis. In Proceedings of the International Conference on Computational Linguistics (COLING 2016).

[7] Abdul-Mageed, M., & Ungar, L. (2017). EmoNet: Fine-grained emotion detection with gated recurrent neural networks. In Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2017)

[8] Cho, K., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2014).

[9] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. Journal of Machine Learning Research

[10] Guan, Z., Chen, L., Zhao, W., Zheng, Y., Tan, S., & Cai, D. (2016). Weakly-supervised deep learning for customer review sentiment classification. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2016)

[11] Dou, Z. Y. (2017). Capturing user and product Information for document level sentiment analysis with deep memory network. In Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP 2017).

[12] A. Hassan and A. Mahmood, "Deep Learning approach for sentiment analysis of short texts", 2017 3rd International Conference on Control, Automation and Robotics (ICCAR), 2017.

[13] N. Zainuddin and A. Selamat, "Sentiment analysis using Support Vector Machine", 2014 International Conference on Computer, Communications, and Control Technology (I4CT), 2014.

[14] Arxiv.org. (2018). [online] Available at: https://arxiv.org/pdf/1702.08835.pdf [Accessed 13 Jun. 2018].

[15] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. and Kuksa, P. (2018). Natural Language Processing (almost) from Scratch. [online] Arxiv.org. Available at: https://arxiv.org/abs/1103.0398 [Accessed 18 Aug. 2018].

[16] Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2018). Efficient Estimation of Word Representations in Vector Space. [online] Arxiv.org. Available at: https://arxiv.org/abs/1301.3781 [Accessed 14 Jun. 2018].

[17] Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2018). Efficient Estimation of Word Representations in Vector Space. [online] Arxiv.org. Available at: https://arxiv.org/abs/1301.3781 [Accessed 18 Aug. 2018].

[18] Joulin, A., Grave, E., Bojanowski, P. and Mikolov, T. (2018). Bag of Tricks for Efficient Text Classification. [online] Arxiv.org. Available at: https://arxiv.org/abs/1607.01759 [Accessed 18 Aug. 2018].

[19] Le, Q. and Mikolov, T. (2018). Distributed Representations of Sentences and Documents. [online] Arxiv.org. Available at: https://arxiv.org/abs/1405.4053 [Accessed 18 Aug. 2018].

[20] Medium. (2018). Chapter 3: Support Vector machine with Math. – Deep Math Machine learning.ai – Medium. [online] Available at: https://medium.com/deep-math-machine-learning-ai/chapter-3-support-vector-machine-with-math-47d6193c82be [Accessed 14 Jun. 2018].

[21] Hackerearth.com. (2018). Simple Tutorial on SVM and Parameter Tuning in Python and R | Machine Learning | HackerEarth Blog. [online] Available at: https://www.hackerearth.com/blog/machine-learning/simple-tutorial-svm-parameter-tuning-python-r/ [Accessed 14 Jun. 2018].

[22] Maas, A. (2018). Sentiment Analysis. [online] Ai.stanford.edu. Available at: http://ai.stanford.edu/~amaas/data/sentiment/ [Accessed 14 Jun. 2018].

[23] Maas, A., Daly, R., Pham, P., Huang, D., Ng, A. and Potts, C. (2018). Learning word vectors for sentiment analysis. [online] Dl.acm.org. Available at: https://dl.acm.org/citation.cfm?id=2002491 [Accessed 14 Jun. 2018].

[24] Zhou, Z. and Feng, J. (2018). Deep Forest. [online] Arxiv.org. Available at: https://arxiv.org/abs/1702.08835 [Accessed 18 Aug. 2018].

Authors Profile

Mrs. Krishna Priya S is pursing her PhD from Shivaji University, Kolhapur. She has 8+ years of teaching experience as an Associate Professor. In addition, she is also a Certified Cloudera Apache Hadoop Developer and IBM Certified Database Associate. At present, she is pursuing her research in sentiment analysis using Deep Learning architectures.

Mr Shaksham Kapoor received his Bachelor's degree in Computer Science from Jaypee University of Engineering and Technology, Guna. Since, his graduation he has been doing research in the field of Data Science with current research being focused on NLP and Deep Learning.

Dr. Kavita S Oza is currently working as Assistant Professor in the Dept. of Computer Science, Shivaji University, Kolhapur. Her thrust areas of research are Data Mining, ICT, Algorithms and Thoery of Languages. She has over 30 research papers in various international and national journals and symposia to her credit. She is a recognized guide for Phd and MPhil progammes in computer science.

Prof.(Dr.) R.K.Kamat is currently HOD of Electronics department at Shivaji University,Kolhapur. He has also worked as a co-ordinator for Computer Science department at Shivaji University . He has successfully executed many research project under Electronics faculty. More than 10 students have completed their Ph.D. under his guidance.

