International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-5, January 2020

Deep Learning Based Opinion Miner and Sentiment Classifier of Social Media Data

K. Jayamalini, M. Ponnavaikko, Jayamalini Kothandan

Abstract*: Social Media opinion Mining and Sentiment analysis is one of the important application of NLP (Natural Language Processing) to analyze online social media users' conversations and discussions , and find out the in depth context of a topic, a brand, a celebrity or a theme. There are various methods and algorithms used in the area of sentiment analysis and opinion mining. The Dictionary Based Approaches only identifies the number of positive words and the negative words in the sentence and takes a decision based on their difference. Thus a phrase "not bad" would became negative phrase. Using doc2vec, a deep learning algorithm which is used in this paper to drive the context from phrases. This algorithm calculates a value of probability of positiveness for each sentence. The range of values 0 -1 are considered as completely negative sentences, and the values from 0.35 to 0.65 are considered as completely positive sentences and the values which are in the middle are considered as neutral.

There are various methods used to analyze and determine the user opinions from the massive amount of online data produced. This paper emphases on usage of deep learning based approach to develop a system to discover public opinions and sentiments. Experimental results prove that the accuracy on both training and testing datasets are high when compared with other machining learning approaches.

Keywords – Social Media Data, Opinion Miner, Sentiment Analyser , Deep Learning, word embedding , word2vec, doc2vec.

I. INTRODUCTION

Social media are computer-mediated [1][3]technologies that facilitate the creation and sharing of information, ideas, career interests and other forms of expression via virtual communities and networks. The variety of stand-alone and built-in social media services currently available introduces challenges of definition; however, there are some common features:

Manuscript published on January 30, 2020.
* Correspondence Author
K. Jayamalini*, Research Scholar, Department of Computer Science Engineering, Bharath University, Selaiyur, Chennai, India.
Email: malini1301@gmail.com.
Dr. M. Ponnavaikko, Provost, Bharath University, Selaiyur, Chennai, India. Email: ponnav@gmail.com.
Jayamalini Kothandan, Assistant Professor, Department of Computer Engineering, Shree L. R. Tiwari College of Engineering, Thane.
Email: jayamalini.k@slrtce.in.

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC-BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

User-generated content, such as text messages, multimedia contents such as digital photos or videos, and data generated through all online interactions, are the lifeblood of social media[1]. Users create service-specific profiles for the website or mobile app that are designed and maintained by the social media organization. Social media facilitate the development of online social networks by connecting a user's with other individuals or groups.

Various Social Medias like twitter, facebook, youtube and instagram are useful in many Businesses to produce successful social drives, distinguish influencers of their product, service & brand, to find strengths and flaws of business competitors, to keep track of the virality of content spread over across the online social media and World Wide

Web and to discover the current trending topics which influences the business.

To determine user opinions and their sentiments about particular event or an individual, in this paper massive amount of Twitter data is used to train the model using word embedding and deep learning algorithms. This paper deals with word2vec a word embedding method to convert textual data into numerical representations and glmnet classifier a deep learning based classifier to classify and find the probability of positiveness of real time tweets.

II. OVERVIEW OF SYSTEM ARCHITECTURE

The following steps are used by the DL Based Opinion Analyzer and Sentiment Classifier [2]. The important steps involved in the construction of a deep learning based model for opinion analyzes and the sentiment classifications are given below:

A. Get the data
B. Create Vocabulary Based Vectorization & Tf-Idf
C. Building Model architecture & Model Parameters
D. Train and test the model
E. Run the model with Real time data

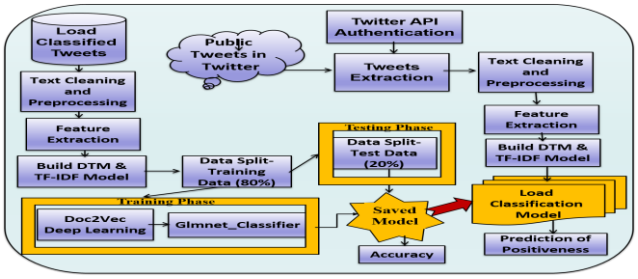

Fig 1: Architecture of DL Based Sentiment Classifier

The Architecture comprises of two main phases:

- Train the Model - to build the model
- Test the Model - to validate the model

Train the Model

The main important steps carried out to train the model are listed below:

- Loading & Pre-Processing A Training Set of Tweets
- Loading Classified Tweets
- Data Splitting On Training and Testing (80% & 20%)
- Pre-Processing & Tokenization
- Creating Vocabulary & Document-Term Matrix
- Construct Term Frequency – Inverse Document Frequency (TF-IDF) Model
- Fit the system using the Training Data
- Transform the data suitable for Fitted system
- Train The System using Glmnet_Classifier
- Save The trained Model for Future Use
- Find Accuracy of the Classifier

Testing Phase:

The important steps carried out in testing phase are listed below:

- Tweets Extraction
- Pre-Processing and Tokenization
- Creating Vocabulary and Document-Term Matrix
- Transforming Data with Tf-Idf
- Loading Classification Model
- Predict Probabilities of Positiveness
- Adding Rates to Initial Dataset (Sentiment Value)
- Visualize The Result

A. Get the data

Sourcing the labelled data for training a deep learning model is one of the most difficult parts of building a model. The data set consists of 1.6 million classified tweets and corresponding polarity value as label, which is used to match each of the sentences to a sentiment score. The value of sentiment score ranges from 0 to 4, 0 being very negative sentiment and 4 being very positive sentiment and the values between 0 and 4 is used to represent the probability of positiveness. The overall data is divided into two parts:

- **Training Data**: 80% of the overall data is used to train system.
- **Test Data:** 20% of the overall data is used as the test data to test the accuracy of the model after training.

B. Create Vocabulary Based Vectorization & Tf-Idf

Before training the model, it is necessary to pre-process, tokenize and create vocabulary for each of the sentences in the corpus and convert them into Tf-Idf model which is used to summarize the text documents. Vocabulary-based Document-Term Matrix (VB - DTM) is created and it is used to fit the system using training data and convert those data suitable for the fitted system. Pre-trained term frequency –inverse document frequency conversion is applied on test data also. The process of transforming words into numerical representation in vector space is called as vectorization, which accumulates the aggregate vocabulary, maps the vocabulary to numerical value, and counts total number of words in the documents.

C. Building Model architecture & Model Parameters

The model is trained using glmnet classifier which is defined in Glmnet package of R. It is used to fit a generalized linear system through penalized maximum likelihood [11]. The computed regularization path for the elasticnet or lasso penalty at a grid of values for lambda , the regularization parameter. cv.glmnet() function finds the optimal value of lambda(λ) and also produces a simpler model.

D. Run the model

Once the model is trained using cv.glmnet() function and saved with .RData extension for future predictions. In real time, the trained model is used on real time data to find out the predictions. The saved model is loaded and the input data is given to the model. It is necessary to preprocess and transform the input data into word embeddings. Then fit the data which is suitable for the model. The saved glmnet classifier takes the preprocessed sentences as input and calculates the sentiment score and predicts the probabilities of positiveness of each sentence.

E. Results and Visuslization

The glmnet classifier calculates the sentiment score of each sentence and predicts the probabilities of positiveness for each sentence. Data visualization techniques are used to visualize the sentiment score and probability of positiveness of each sentence.

III. METHODS AND ALGORITHMS

The important methods used in the implementation of the proposed system are:

- Word Embeddings
- Word2vec and Doc2vec
- Glmnet – (generalized linear model)

A. Word Embeddings

Word embedding [4] is popular way of representing document vocabulary. It captures word's context in a text document, syntactic and semantic similarity and relation with other words. Simply it is a vector representation of a specific word in which texts are converted into numbers.

Word2Vec is one of the most popular technique to learn and produce word embeddings. These are shallow 2-layered artificial neural networks (ANN) used to train and recreate phonological contexts of words. Enormous amount of text data is given to Word2Vec and several hundred dimensions of a vector space is produced as an output, with each single word of the text data is represented as a vector. Word vesctors are positioned in the vector space so that terms of the corpus that shares the common contexts are placed close vicinity to each other in the space. Such representations [10] of word vectors capture different relationships between words, like analogies, synonyms or antonyms, as shown in figure down:

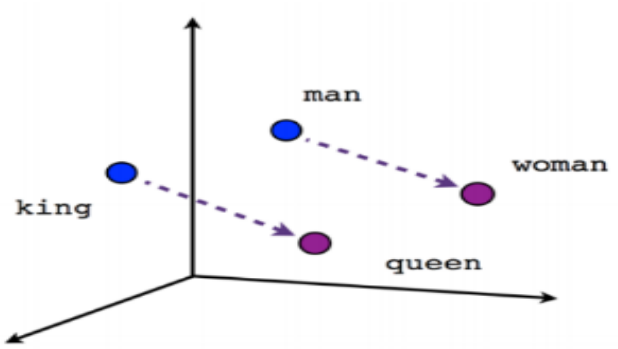

Fig .2: Remarkable analogies of word embedding – ExampleThere are two types of word embeddings broadly used.

a) Frequency based word Embedding
b) Prediction based word Embedding

a) Frequency based word Embedding

There are three different types of methods under this group.

- Count Vector
- TF-IDF
- Co-Occurrence Matrix

b) Prediction based word Embedding

There are two different types of methods [5] under this group.

- **CBOW** , the Continuous Bag-of-Words Model
- **SGM**, the Skip-Gram Model

These methods forms basics for word2vec representations.

B. Word2vec algorithms

There are two ways to create word2vec representation of words corpus.

a) **CBOW** , the Continuous Bag-of-Words Model
b) **SGM**, the Skip-Gram Model

These model architectures were used to learn distributed and numerical representations of words that reduces the computational complexity.

a) *CBOW* Model

CBOW model predicts an output term (center term), when given a bag of surrounding terms (context words). To predict the output word, this model uses a sliding window around input words, i.e the adjoining words. Every word is characterized as a feature vector.

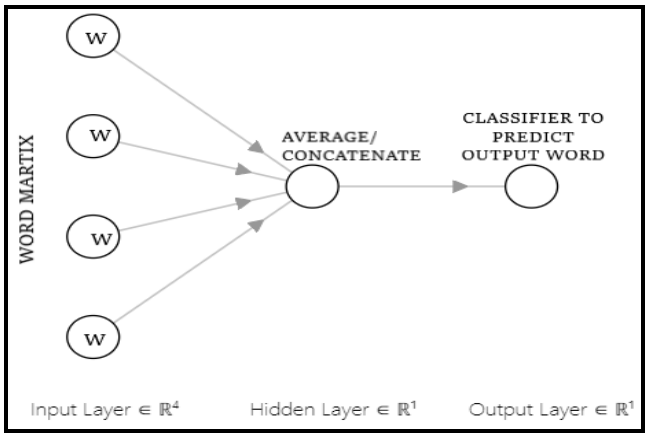

Fig. 3: System to find output word when given context words For example,

{"Actions", "louder", "than", "words"} be the given set of context words and from these set, system should capable to guess the center word "speak".

Parameters & Notations of CBOW Based System:

- $x^{(c)}$ – one hot vector representation of Input
- y or $y^{(c)}$ - Output word
- w_i : i^{th} term of vocabulary V
- $V \in R^{n\times|V|}$: Matrix with Input Terms
- v_i : i-th column of Matrix V
- $U \in R^{n\times|V|}$: Matrix with Output Terms
- ui : i-th row of Matrix U
- n - the random size of embedding space

Steps carried out in CBOW Model:

i. Generate one hot word vectors of size 'm' i.e. ($x^{(c-m)}$, . . . , $x^{(c-1)}$, $x^{(c+1)}$, . . . , $x^{(c+m)}$) , where 'm' is size of input context terms
ii. Resultant embedded word vectors for the context (v_{c-m} = $Vx^{(c-m)}$, v_{c-m+1} = $Vx^{(c-m+1)}$, . . ., v_{c+m} = $Vx^{(c+m)}$)
iii. Find average value of embedded word vectors = $(v_{c-m}+v_{c-m+1}+...+v_{c+m)/}$ 2m
iv. Calculate score vector z using z= U*AV
v. Generated Probabilities of Score (ŷ) = softmax(z)
vi. The Loss function used in this model to find the difference between true probability(y) and predicted probability (ŷ) is given by :

$$H(\hat{y}, y) = -yi \log(\hat{y}_i)$$

vii. The objective function in CBOW model is calculated as negative log likelihood of a word given a set of context words and it is given by following formula:

$$\frac{\exp(v'_{wo} T\ v_{wi})}{\sum_{w=1}^{W} \exp(v'_{w} T\ v_{wi})}$$

wo : output word
wi: context words

viii. Update all related word vectors using Gradient descent algorithm

In CBOW methods, the gradient of error with respect to input-hidden weights and hidden-output weights and were calculated using linear activation methods.

b) Skip gram Model

The algorithm used in word embedding is Skip Gram Model which is straight opposite to CBOW methods. Instead of predicting one word each time in CBOW, here one word is used as input to predict all surrounding words i.e. "context words". As compared to CBOW, SGM is considerably slower than CBOW but produces more precise results with rare, infrequent and common words.

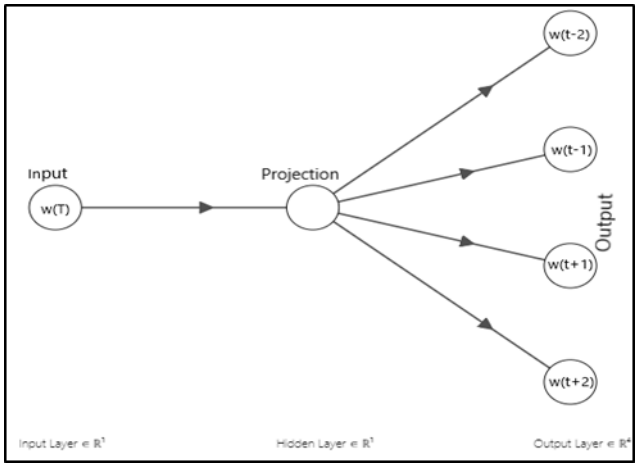

Fig. 4: System to find context words when given a input wordFor example,

Given the center word "jumped", the system will predict the surrounding words "The", "cat", "over", "the", "puddle".

Parameters & Notations of Skip-Gram Model:

x - Input one hot vector (center word)
y (j)- Output vectors
V and U – Input & Output word matrix
w_i - Word i from vocabulary V
$V \in R^{n\times|V|}$ -Matrix of Input word
v_i - i^{th} column of Input Matrix V
$U \in R^{n\times|V|}$ - Matrix of Output words
ui - i^{th} row of Output Matrix U
Steps in Skip-Gram Model:

i. Generate one hot representation of input vector x
ii. Generate embedded term vectors for context words $v_c = V_x$
iii. No average value is calculated like CBOW model, just set $v\hat{} = vc$
iv. Generate score vectors, $u_{c-m}, \ldots, u_{c-1}, u_{c+1}, \ldots, u_{c+m}$ using $u = Uv_c$.
v. Convert these scores into probabilities, y = softmax(u)
vi. Match the probability vector generated to the true probabilities ie the one hot vectors representations of output which are $y^{(c-m)}, \ldots, y^{(c-1)}, y^{(c+1)}, \ldots\ldots, y^{(c+m)}$

The objective function to compute gradients is given by the formula:

$$\text{minimize } J = -\sum_{j=0,j6=m}^{2m} u_{c-m}^{T} Vc + 2m \log \sum_{k=1}^{|V|} \exp\left(u_k^T v_c\right)$$

C. Doc2vec

Doc2vec [6] otherwise called as text2vec which is simple extension of word2vec and it is used to construct an equivalent numerical illustration of a text document, irrespective of its size. The methods used in predicting the documents are:

i) Distributed memory model for Paragraph Vector (PV-DMM)

ii) Distributed bag of words Model for Paragraph Vector (PV-DBOWM)

a) PV-DMM

In CBOW word2vec method, another feature vector called paragraph id is concatenated with the wordvec to get the document-unique text2vec representation, which is used to guess the next term in the document.

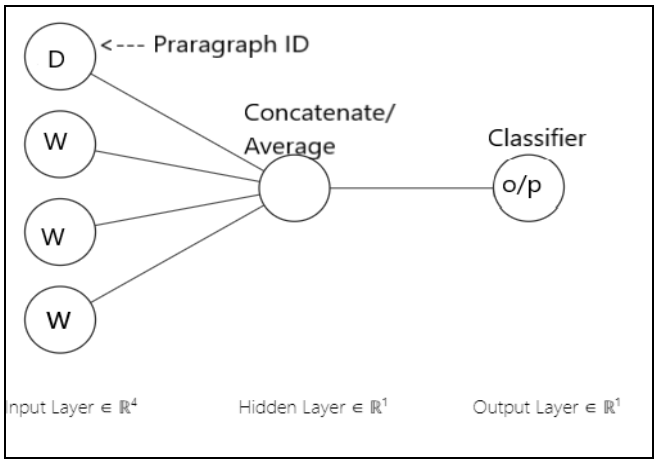

Fig. 5: PV-DMM model

In the training phase along with word vectors 'W', the document vector D also get trained using stochastic gradient descent and the gradient is obtained via back propagation. At the end it holds a numeric representation of the document. The above trained model is called as *of Paragraph Vector Distributed Memory Model* (PV-DMM). It acts as a memory device that stores missing words from the current context or the topic of the paragraph. While the word vectors represent the concept of a word, the document vector intends to represent the concept of a document.

For example,

N – Number of paragraphs in the corpus

M – Number words in the vocabulary,

p- Dimension mapped to paragraph

q- Dimensions mapped to words

Total number of parameters in model = N × p + M × q

When N takes large value, the updates during training phase are normally sparse in nature and thus efficient. After the model is trained, the paragraph vectors contains features of the paragraph. These features can be directly used with any machine learning techniques such as logistic regression, SVM or K-means. At the time of prediction, system need to perform an inference step to compute the paragraph vector using gradient descent for a new paragraph. The main advantage of this model is that it was trained and learned using unlabeled data and thus it can work very well on tasks that do not have an adequate amount of labeled data.

b) PV-DBOWM (Distributed BOW Model)

There is another method which is like skip-gram based word2vec model. It uses PV-DBOWM for prediction. This method ignores the context words in the input, but train the system to guess the output terms arbitrarily from the given passage.

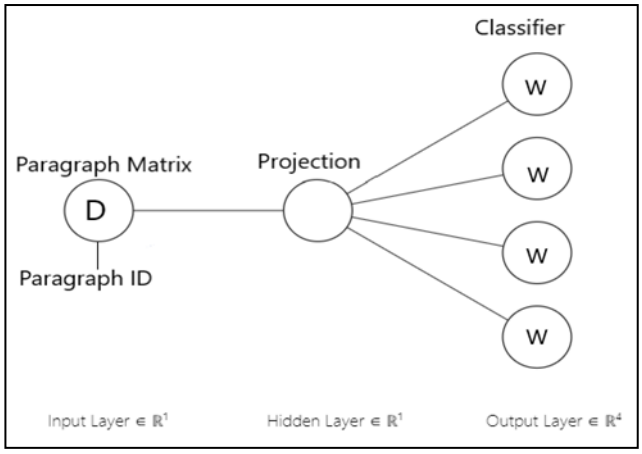

Fig. 6: PV-DBOWM

This method uses stochastic gradient descent for updates. For each iteration, this algorithm, samples a text window, then a arbitrary word is fetched from the text window. It accomplish classification task on the given Paragraph Vector. This is depicted in above figure.
This method is called as PV-DBOWM, and it is opposite to PV-DMM. This method is faster than PV-DMM because there is no need to store and process the word vectors, thus it consumes less memory.

D. glmnet Classifier

Glmnet stands [7] for generalized linear model it fits the model via penalized maximum likelihood. The regularization path is calculated for the elasticnet penalty or lasso penalty at a grid values of lambda (λ). It can deal with all shapes of data, and huge sparse matrices. It fits all models like logistic, linear, multinomial and Cox regression models.
This algorithm is extremely fast. The glmnet classifier used to find solution for the following problem:

$$\min_{\beta 0,\beta 1} \frac{1}{N} \sum_{i=1}^{N} wi\,l(yi, \beta 0 + \beta^{T} xi) + \lambda[(1-\alpha)\frac{||\beta||}{2} + \alpha||\beta||]$$

Here,
λ - Tuning parameter controls complete strength of penalty
l(y,η) – observation's negative log-likelihood contribution
α - Elastic-net penalty controller. Its values lies between 0 - 1. For lasso the default value is (α=1) and ridge the default value is (α=0).
Basic idea of glmnet algorithm:
If the coefficient penalty λ is very large then:
All coefficients = 0.0
A small change in λ minimizes the value of w.
Track incremental changes and find optimum value of w using coordinate descent algorithm.
The function cv.glmnet() in R glmnet package is used to train the model
glmnet(x1,y1,family=c("binomial","gaussian","poisson","cox""multinomial"), weights, offset=NULL, alpha=1, nlambda =100,lambda.min.ratio = ifelse(nobs))
x1- Input matrix
y1- Response variable.
Weights - Observation weights, Default value is 1
Offset - Linear prediction Vector. Values vary based on family. Default value is NULL.
Alpha - Elasticnet mixing parameter, holding values from 0≤α≤1. For lasso penalty α =1 , and α =0 for ridge penalty.

Lambda - user supplied lambda sequence. Computed programmatically based on nlambda and lambda.min.ratio.

In our experiment, glmnet() function is used with following parameters to train the model.
cv.glmnet(1 = dtmtraintfidf, y1 = tweetstrain[['sentiment']], family = 'binomial', alpha = 1, type.measure = "auc", nfolds = 15, thresh = 1e-5, maxit = 1e5)
Where,
dtmtraintfidf – Fitted and transformed model as input using training data.

IV. IMPLEMENTATION DETAILS

This section explains the details of implementation and results. The two phases of implementations are explained in details:

A. Training Phase
B. Testing Phase

A. Training Phase

The model is trained in training phase using 1.6 million preprocessed and labeled tweets. 80% of dataset is used as input to train the system and 20% of total data is used for testing to test and validate the trained system. The algorithm fits the model for the following values of lambda:
lambda.min = 0.0008566753
lambda.1se = 0.0008566753
lambda.1se means largest lambda value, where error is within 1 standard error of minimum.

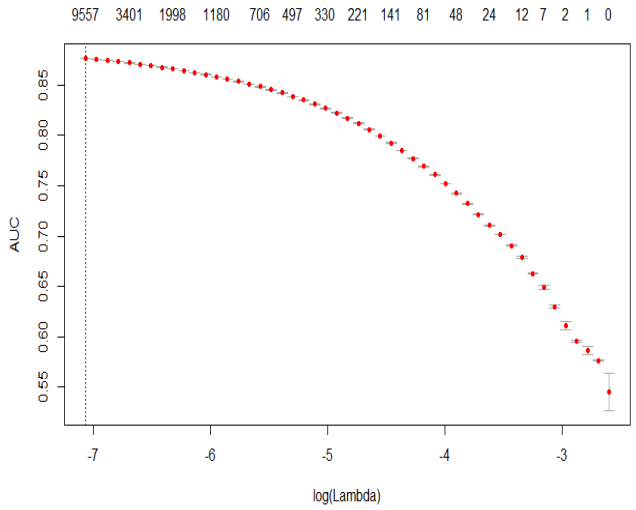

Fig. 7: Glmnet Classifier Plot

While training, the following Glmnet Parameters are learned and shown below:

```
> glmnet_classifier
$`lambda`
 [1] 0.0745092491 0.0678900527 0.0618588876 0.0563635145 0.0513563352 0.0467939799 0.0426369317 0.0388491842 0.0353979298
[11] 0.0293879832 0.0267772358 0.0243984199 0.0222309316 0.0202559969 0.0184565099 0.0168168845 0.0153229188 0.0139616729
[21] 0.0115912261 0.0105614936 0.0096232396 0.0087683375 0.0079893826 0.0072796278 0.0066329258 0.0060436749 0.0055067715
[31] 0.0045718184 0.0041656707 0.0037956040 0.0034584130 0.0031511771 0.0028712352 0.0026161626 0.0023837499 0.0021719842
[41] 0.0018032194 0.0016430264 0.0014970644 0.0013640694 0.0012428892 0.0011324743 0.0010318684 0.0009402001 0.0008566753

$cvm
 [1] 0.5452936 0.5762027 0.5865763 0.5952772 0.6111913 0.6300673 0.6491588 0.6629362 0.6787663 0.6906698 0.7017491 0.71074
[14] 0.7323803 0.7423432 0.7521210 0.7602957 0.7691211 0.7770709 0.7852075 0.7926799 0.7995719 0.8060368 0.8119567 0.81738
[27] 0.8268412 0.8310386 0.8350319 0.8387504 0.8422235 0.8454436 0.8484092 0.8511480 0.8537247 0.8561168 0.8583570 0.86048
[40] 0.8643854 0.8661586 0.8678099 0.8693431 0.8707834 0.8721411 0.8734236 0.8746321 0.8757720 0.8766797

$cvsd
 [1] 0.0186596452 0.0004682367 0.0039981791 0.0011105587 0.0037959529 0.0012874964 0.0024444864 0.0006776788 0.0009433775
[11] 0.0002765356 0.0004012671 0.0006402052 0.0003736889 0.0002576935 0.0002464280 0.0002529537 0.0002628116 0.0003964013
[21] 0.0003629119 0.0003430861 0.0003413208 0.0003597961 0.0003609177 0.0003569135 0.0003561599 0.0003490748 0.0003347161
[31] 0.0003017790 0.0002846371 0.0002775156 0.0002727076 0.0002707998 0.0002563441 0.0002428328 0.0002351275 0.0002280528
[41] 0.0002105556 0.0002009810 0.0001910905 0.0001798277 0.0001682285 0.0001634033 0.0001635529 0.0001679172 0.0001646013
```

Fig. 8: Glmnet Classifier Parameters (lambda, cvm & cvsd)

```
$cvup
 [1] 0.5639533 0.5766709 0.5905745 0.5963878 0.6149873 0.6313548 0.6516033 0.6636139 0.6797096 0.6912984 0.7020256 0.7111487
[14] 0.7327539 0.7426009 0.7523674 0.7611787 0.7693839 0.7774673 0.7855456 0.7930428 0.7999150 0.8063781 0.8123165 0.8177412
[27] 0.8271973 0.8313876 0.8353666 0.8390723 0.8425253 0.8457282 0.8486867 0.8514207 0.8539955 0.8563731 0.8585998 0.8607203
[40] 0.8646057 0.8663692 0.8680109 0.8695342 0.8709632 0.8723093 0.8735870 0.8747956 0.8759399 0.8768443

$cvlo
 [1] 0.5266340 0.5757344 0.5825781 0.5941667 0.6073954 0.6287798 0.6467143 0.6622585 0.6778229 0.6900412 0.7014725 0.7103462
[14] 0.7320066 0.7420855 0.7518745 0.7606727 0.7688583 0.7766745 0.7848694 0.7923170 0.7992288 0.8056954 0.8115969 0.8170193
[27] 0.8264850 0.8306895 0.8346972 0.8384285 0.8419217 0.8451589 0.8481316 0.8508753 0.8534539 0.8558604 0.8581142 0.8602500
[40] 0.8641651 0.8659480 0.8676089 0.8691520 0.8706035 0.8719728 0.8732602 0.8744685 0.8756040 0.8765151

$nzero
  s0  s1  s2  s3  s4  s5  s6  s7  s8  s9 s10 s11 s12 s13 s14 s15 s16 s17 s18 s19 s20 s21 s22 s23 s24
   0   1   1   2   2   6   7  10  12  16  19  24  31  42  48  58  69  81  98 124 141 164 192 221 256
```

Fig. 9: Glmnet Classifier Parameters (cvup, cvlo and nzero)

```
$lambda.min
[1] 0.0008566753

$lambda.1se
[1] 0.0008566753

attr(,"class")
[1] "cv.glmnet"
```

Fig. 10: Glmnet Classifier lambda values

```
$glmnet.fit

Call:  glmnet(x = dtm_train_tfidf, y = tweets_train[["sentiment"]],     family = "binomial", alpha = 1, thresh = 0.001, maxit = 1000)

       Df      %Dev    Lambda
 [1,]   0 -2.051e-11 0.0745100
 [2,]   1  2.720e-03 0.0678900
 [3,]   1  4.982e-03 0.0618600
 [4,]   2  7.118e-03 0.0563600
 [5,]   2  1.024e-02 0.0513600
 [6,]   6  1.577e-02 0.0467900
 [7,]   7  2.240e-02 0.0426400
 [8,]  10  2.997e-02 0.0388500
 [9,]  12  3.772e-02 0.0354000
[10,]  16  4.631e-02 0.0322500
[11,]  19  5.539e-02 0.0293900
[12,]  24  6.426e-02 0.0267800
[13,]  31  7.364e-02 0.0244000
[14,]  42  8.407e-02 0.0222300
[15,]  48  9.482e-02 0.0202600
[16,]  58  1.058e-01 0.0184600
[17,]  69  1.167e-01 0.0168200
[18,]  81  1.275e-01 0.0153200
[19,]  98  1.384e-01 0.0139600
[20,] 124  1.497e-01 0.0127200
```

Fig. 11: Glmnet Classifier for Traing Dataset

The predicted positiveness of the sample test data by the glmnet classifier is given below:

```
> preds <- predict(glmnet_classifier, dtm_test_tfidf, type = 'response')[ ,1]
> preds
  1467811372   1467811592   1467812723   1467812799   1467812964   1467813782   1467813985   1467816665
5.809598e-02 1.037092e-01 1.529464e-01 1.888132e-01 1.626459e-01 5.561425e-01 2.231370e-01 1.092986e-01
  1467818481   1467819712   1467820863   1467822389   1467822687   1467825084   1467834001   1467834227
4.521859e-01 4.219172e-01 4.118685e-01 1.644541e-01 3.275473e-01 5.977620e-01 7.067132e-02 3.229533e-01
  1467834400   1467835305   1467836583   1467837762   1467839007   1467839737   1467839816   1467840386
```

Fig. 12: Glmnet Classifier – Predicted values

Maximum accuracy given by the trained model is 0.8767. The maximum accuracy of the model obtained using test data is 0.8754919. Finally the model is saved with extension .rds (R data file) for future use.

B. Testing Phase

Real time tweets are extracted using 'twitteR' and 'ROAuth' packages. The extracted tweets are preprocessed and tokenized. Then data is converted into document – term matrix (DTM) format for creating equivalent word embedding. Finally the data is transformed into tf-idf format. The saved classifier is loaded and the tf-idf format of preprocessed tweets are given as input to classifier.
Few samples of extracted real time tweets about West Indies tour 2019 are shown in fig. 13 below.

```
1                    RT @CricketNDTV: #WestIndies players have been docked 8
0 per cent of their match fees for maintaining a slow over-rate in the fir
st ODI aga
2                    RT @CricketNDTV: #WestIndies players have been docked 8
0 per cent of their match fees for maintaining a slow over-rate in the fir
st ODI aga
3                     #WestIndies players have been docked 80 per cent of th
eir match fees for maintaining a slow over-rate in the first https://t.co/
QrATnCbmYB
4                                    India vs West Indies | Shreyas Iyer Sh
ows Once Again That Hes The RealDeal https://t.co/7Q9RlFEVmC https://t.co/
xPO87dOBNR
5                          India vs West Indies: @windiescricket fined for s
low over-rate in first ODI\nhttps://t.co/9gfuYRHdLO\n\n@BCCI https://t.co/
9nvSDFn6xM
6                                    India vs West Indies | Shreyas Iyer Sh
ows Once Again That Hes The RealDeal https://t.co/ZgANZty2ma https://t.co/
SuUJrvZ53z
7                    RT @FirstpostSports: The entire West Indies side has be
en imposed with an 80 percent fine for maintaining slow over-rate in the f
irst ODI #
8                    The entire West Indies side has been imposed with an 80
percent fine for maintaining slow over-rate in the first OD https://t.co/r
Xc2FqTW7e
9                     RT @ESPNcricinfo: Which side of the r
un out debate are you on?\n\n#INDvWI https://t.co/jTGJrlZ5CZ https://t.co/
VIrhKMvEy3
10                    West Indies cricket team took 4 hours and 11 minutes a
s against 3 hours and 30 minutes to bowl their full quota of https://t.co/
9GEHJeZRap
```

Fig.13: Sample Tweets Extracted

TF-IDF values of real time tweets i.e. corresponding word embeddings are shown below in fig. 14.

```
1000 x 612386 sparse Matrix of class "dgCMatrix"
   [[ suppressing 62 column names 'ambermarion', 'healthfair', 'claire.only' ... ]]
   [[ suppressing 62 column names 'ambermarion', 'healthfair', 'claire.only' ... ]]


1206554926194708481 . . . . . .
 . ......
1206554356985683971 . . . . . .
 . ......
1206554305626402816 . . . . . .
 . ......
1206553446863777792 . . . . . .
 . ......
1206552581431947264 . . . . . .
 . ......
1206548875986771969 . . . . . .
 . ......
1206548698773295104 . . . . . .
 . ......
1206548386515542017 . . . . . .
```

Fig 14: TF-IDF values of sample tweets

Predicted sentiment values of tweets and its corresponding word embedding generated by the glmnet classifier is shown in fig 15 below.

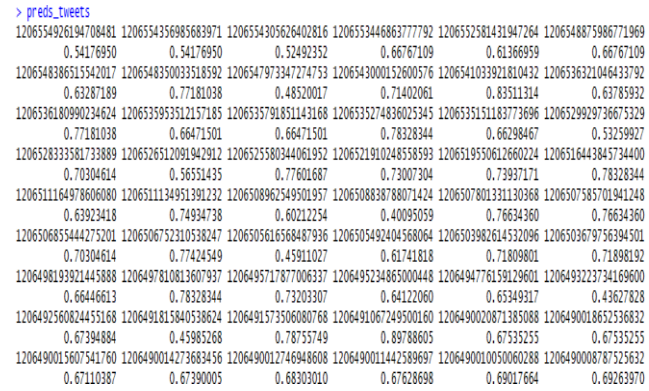

Fig. 15: Glmnet Classifier –predict probabilities of positiveness for Each Tweet

The sentiment values of each tweet calculated by classifier is shown in figure 16 below.

Fig. 16. Sentiment Values calculated by Model for Each Tweet

V. EXAMINATION OF RESULTS

When the glmnet classifier is trained using word embedding, it learns the following parameters:

a. Lambda

It is regularization parameter. When the classifier is trained, it produces a range of values from 'lambda minimum' to 'maximum' and the optimal lambda value is selected by the classifier based on the cross validation. The sequence of lambda values generated are shown in table 1 below:

Table 1: Sequence of lambda values

0.074635 7659	0.068005 3301	0.061963 9240	0.056459 2198	0.051443 5383	0.046873 4362	0.042709 3293	0.038915 1502	0.035458 0355
0.032308 0414	0.029437 8840	0.026822 7035	0.024439 8485	0.022268 6797	0.020290 3916	0.018487 8491	0.016845 4396	0.015348 9371
0.013985 3799	0.012742 9573	0.011610 9081	0.010579 4270	0.009639 5799	0.008783 2262	0.008002 9486	0.007291 9887	0.006644 1885
0.006053 9371	0.005516 1220	0.005026 0849	0.004579 5814	0.004172 7440	0.003802 0489	0.003464 2854	0.003156 5278	0.002876 1106
0.002620 6049	0.002387 7975	0.002175 6722	0.001982 3915	0.001806 2813	0.001645 8162	0.001499 6064	0.001366 3855	0.001244 9996

b. Cross validated error(CVM)

For each value of lambda, the calculated value of cvm is shown in table 2 below:

Table 2: CVM values for sequence of lambda values

0.5457334	0.5757126	0.5864656	0.5941440	0.6104537	0.6290759	0.6489424	0.6625237	0.6786204
0.6907509	0.7013146	0.7104505	0.7215039	0.7319952	0.7420164	0.7517865	0.7607313	0.7689634
0.7768819	0.7848706	0.7923427	0.7991243	0.8054997	0.8114920	0.8169244	0.8219115	0.8263917
0.8306444	0.8346559	0.8383987	0.8419467	0.8451841	0.8481370	0.8508927	0.8534507	0.8558474
0.8581111	0.8602355	0.8622306	0.8640967	0.8658373	0.8674579	0.8689828	0.8704314	0.8718037

c. Estimation of Standard error(cvsd) of CVM

The estimated standard error of cross validation error is shown in table 3 below:

Table 3: Estimated standard error of cross validated error

0.018206 9494	0.000454 2935	0.003970 4570	0.000467 8252	0.004464 6533	0.001186 9329	0.002605 1687	0.000608 4796	0.000743 5910
0.000467 2434	0.000253 1824	0.000343 2082	0.000575 0046	0.000186 6539	0.000188 8544	0.000215 1094	0.000228 8077	0.000361 5265
0.000249 1168	0.000300 8971	0.000296 6572	0.000336 8456	0.000355 0850	0.000335 1444	0.000325 8116	0.000307 4946	0.000286 7022
0.000286 0492	0.000284 9793	0.000301 4704	0.000304 5309	0.000300 1952	0.000285 8155	0.000279 9527	0.000276 7744	0.000269 8865
0.000250 1870	0.000233 6225	0.000218 0422	0.000209 5059	0.000202 8240	0.000198 5767	0.000199 9345	0.000200 0128	0.000203 6871

d. Cross validation Upper Curve(Cvup) & Lower Curve (cvlo)

The upper curve of cross validation is calculated using the formula cvup=cvm+cvsd and the lower curve is calculated using the formula cvlo=cvm=cvsd. The calculated values of above parameters are shown in table 4 below:

Table 4: estimated standard error of cross validated error

$cvup								
0.5639404	0.5761669	0.5904361	0.5946119	0.6149183	0.6302628	0.6515476	0.6631321	0.6793640
0.6912181	0.7015678	0.7107937	0.7220789	0.7321818	0.7422052	0.7520016	0.7609601	0.7693249
0.7771310	0.7851715	0.7926394	0.7994611	0.8058548	0.8118272	0.8172502	0.8222190	0.8266784
0.8309304	0.8349408	0.8387002	0.8422512	0.8454843	0.8484228	0.8511727	0.8537275	0.8561173
0.8583613	0.8604691	0.8624486	0.8643062	0.8660401	0.8676565	0.8691827	0.8706314	0.8720074
$cvlo								
0.5275265	0.5752583	0.5824951	0.5936762	0.6059890	0.6278889	0.6463373	0.6619152	0.6778768
0.6902837	0.7010614	0.7101073	0.7209289	0.7318085	0.7418275	0.7515714	0.7605025	0.7686018
0.7766327	0.7845697	0.7920461	0.7987874	0.8051447	0.8111569	0.8165986	0.8216040	0.8261050
0.8303583	0.8343709	0.8380972	0.8416422	0.8448839	0.8478512	0.8506128	0.8531739	0.8555775
0.8578609	0.8600019	0.8620125	0.8638872	0.8656345	0.8672594	0.8687829	0.8702314	0.8716000

e. Non zero coefficients of ' λ'

Table 5 below shows the nonzero coefficient values of lambda:

Table 5: estimated standard error of cross validated error

s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14
0	1	1	2	2	6	7	10	12	16	19	24	30	42	48
s15	**s16**	**s17**	**s18**	**s19**	**s20**	**s21**	**s22**	**s23**	**s24**	**s25**	**s26**	**s27**	**s28**	**s29**
58	68	82	95	125	139	162	195	224	256	287	329	375	433	500

Finally the type of measure used by the classifier is "auc", and the minimum value of lambda obtained is 0.0009417966. The trained glmnet classifier is fitted for the whole data and it saved with .rds extension for future use.

When the classifier is loaded and used in real time to predict the user sentiment about a particular event, thing or a celebrity. The tweets are extracted and converted to its equivalent word embedding and given as input to the model.

The model predicts the sentiment score of each tweet and shows the probability of positiveness of each. The sentiment score of each tweet is shown below in table 6.

Table 6: Sentiment score calculated by the classifier

Tweet ID	Tweet Text	Equivalent Word Embedding value	Sentiment Score	Classification
1.	Vkohli creates yet another batting record.Im Ro45 now holds the record of hitting most si x.	1210483349124313090	0.6971788	69% of positiveness
2.	AUSvsNZ Smith averaging 84 vs India and 165 vs West Indies,his lowest is actually aga inst Bangladesh.	1210482905610248194	0.6540581	65% of positiveness
3.	India's vice-captain Rohit Sharma, on Sunda y, broke former Sri Lankan captain Sanath J ayasuriya's record	1210473735020351489	0.7459870	74% of positiveness
4.	Virat Kohli trolls Yuzvendra Chahal before I ndia vs West Indies 3rd ODI	1210344671110717440	0.6624723	66% of positiveness
5.	Gill is the youngest Indian to hit a double to n in a first class match	1210342761817411586	0.6747127	67% of positiveness
6.	India vs West Indies second ODI: India bou nce back in spectacular fashionSource	1210274717220782085	0.6914615	69% of positiveness
7.	India vs West Indies second ODI:India boun ce back in spectacular fashion	1210259379896774656	0.6540581	65% of positiveness
8.	Rohit (63) and Rahul added 122 runs for the opening stand to lay the platform for India's chase of 316.	1210249440079966208	0.7044055	70% of positiveness
9.	Lewis and Hope set a strong base forwWest Indies with an opening partnership	1210234036590981120	0.7165891	71% of positive ness
10.	Rohit Sharma reacts to being called 'Borivali Ka Don' by fans during India vs West Indie s match in Mumbai	1210220201075204096	0.5608576	56% of positiveness
11.	Ninth century for Kohli in 35 innings against West Indies equals Tendulkars record for m ost centuries against	1210208325964402689	0.6855791	68% of positiveness
12.	How police pushed the envelope to enhance spectator comfort and experience	1210203855964819457	0.6041249	60% of positiveness

Visualization of the above sentiment values are plotted using ggplot package in R, which is shown below in figure 16. The green line in the plot is the borderline for the positive tweets and the red line in the plot is the borderline for the negative tweets. The sentiment values of each tweets are plotted with red color for negative score, yellow color for neutral tweets and green color for positive tweets. Most of the tweets are neutral comments, that's why the plot depicts those values in yellow color.

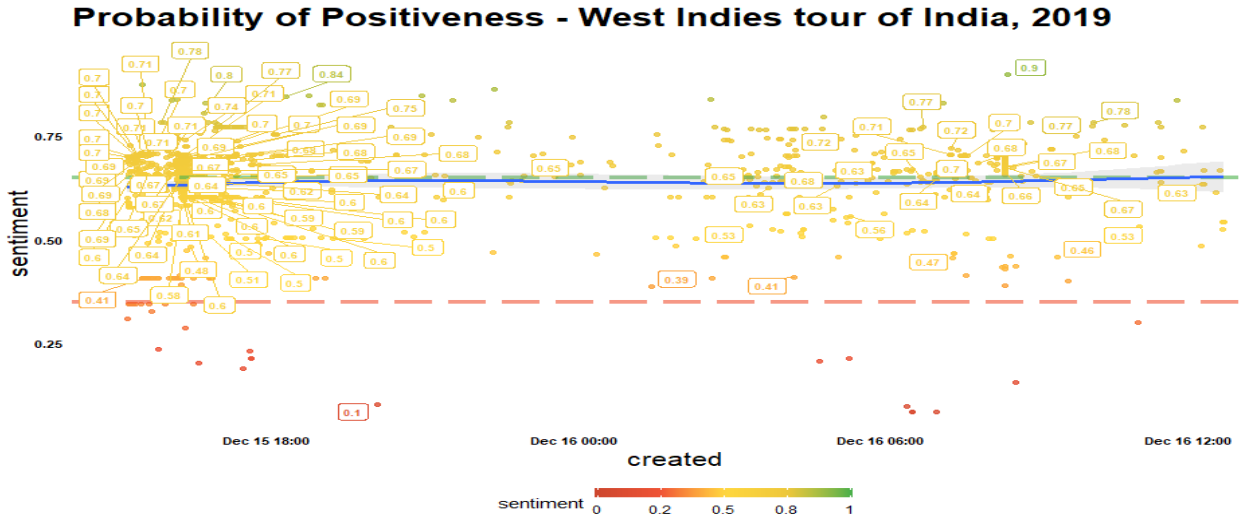

Fig .16: Visualization of Probability of Positiveness calculated by Model

VI. CONCLUSIONS

This paper explained in detailed about deep learning based implementation of sentiment analyzer and opinion miner. This paper deals with different approaches of word embedding, glmnet classifier and how both are used in effectively finding the user opinion and sentiments of social media data. Dictionary based and affective lexicon based approaches give results based on comparison of words with the dictionaries to predict the value. But deep learning based approach gives the probability of positiveness of each sentence in the corpus instead of predicting positive, negative and neutral. This approach gives best results compared to dictionary based approaches and also help [7] the business to create successful social campaigns, recognize brand influencers, compare competitors analysis, discovery of trending topics and customer feedback towards the business.

REFERENCES

1. K. Jayamalini and M. Ponnavaikko, " Social Media Mining: Analysis of Twitter Data to Find user Opinions about GST", Journal of Engineering and Applied Sciences, Year: 2019, Volume: 14, Issue 12 , pp. 4167-4175
2. Min-Yuh Day, Yue-Da Lin," Deep Learning for Sentiment Analysis on Google Play Consumer Review", IEEE International Conference on Information Reuse and Integration, 2017,pp. 382-387.

3. K. Jayamalini and M. Ponnavaikko, " Enhanced Social Media Metrics Analyzer Using Twitter Corpus as an Example", International Journal of Innovative Technology and Exploring Engineering (IJITEE), May 2019, Volume 8, Issue 7 , pp. 822- 828.
4. QuocLe,TomasMikolov," Distributed Representations of Sentences and Documents ", Proceedings of the 31st International Conference on Machine Learning,Beijing,China,2014, pp.1-9.
5. S. Bhoir, T. Ghorpade ,V. Mane, "Comparative analysis of different word embedding models," *International Conference on Advances in Computing,Communication and Control(ICAC3)*, Mumbai, 2017, pp.1-4.
6. H. İ. Çelenli, S. T. Öztürk, G. Şahin, A. Gerek and M. C. Ganiz, "Document Embedding Based Supervised Methods for Turkish Text Classification," *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, Sarajevo, 2018, pp. 477-482.
7. K. Jayamalini, M. Ponnavaikko, "Research on web data mining concepts techniques and applications", *2017 International Conference on Algorithms Methodology Models and Applications in Emerging Technologies (ICAMMAET) Chennai*, pp. 1-5, 2017.
8. https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html accessed on December 2019 .
9. https://www.rdocumentation.org/packages/glmnet/versions/3.0-1/topics/glmnet accessed on December 2019.
10. https://developers.google.com/machine-learning/crash-course/embeddings/translating-to-a-lower-dimensional-space accessed on December 2019.
11. Qike Wang, Jason Q. D. Goodger, Ian E. Woodrow, Le Chang, Mark A. Elgar. "Task-Specific Recognition Signals Are Located on the Legs in a Social Insect", Frontiers in Ecology and Evolution, 2019.

AUTHORS PROFILE

K. Jayamalini, post graduate in Computer Engineering and pursuing Phd. in computer science engineering. Published around 25 papers in various National and International Journals. Worked in various multinational software organizations as a software developer and a team leader and moved to teaching. Currently working as an Assistant Professor in Shree L.R. Tiwari college of Engineering, Mumbai.

Dr. Murugesan Ponnavaikko, graduated in Electrical Engineering from Guindy Engineering College in 1969 and obtained his M.Sc.(Engg.) in Power Systems in 1972 and Ph.D. Optimal Distribution System Planning from I.I.T.(Delhi) in 1983. He started his career from Tamil Nadu State Electricity Board in 1972 and has served in different organizations including Indian Institute of Science (Bangalore), Southern Regional Electricity Board (Bangalore), Bharath Heavy Electricals Limited (New Delhi) and Rural Electrification Corporation (New Delhi) from 1972 till 1984. In 1984, he was deputed to ECCO (Electrical Construction Company), Libya as an Advisor and Secretary to the Board. After serving for 2 years at that capacity at ECCO, he moved to the Higher Institute of Mechanical and Electrical Engineering, Hoon, Libya as a Visiting Professor in 1986 and served there till 1989. The from 1989 till Jan.1995, he was serving as Professor and Head of the Department of Computer Science and Engineering, initially at Mookambigai College of Engineering, Tiruchirappalli, and then at Regional Engineering College, Tiruchirappalli. While at Mookambigai College of Engineering he devised Syllabus for M.Phil. (Comp.Sc.) for the Summer Sequential Programme, instituted at Bharathidasan University and conducted the course for the first time in the University as its Course Director and continued offering the same course while at Regional Engineering College. He later served as Professor and Head of the Department of Computer Science at the Crescent Engineering College, Chennai from 1995 till 2000. From July 2000. He served as the Director, at the level of Vice-Chancellor at the Tamil Virtual University from July 2000 till August 2003. From September 2003 he was serving as the Director, Research and Virtual Education Directorate at the SRM University till June 2007. From July 2007 he was serving as the Vice-Chancellor, Bharathidasan University, Tiruchirappalli, Tamil Nadu till July 2010. From July 2011 he was serving as Vice-Chancellor, SRM University till August 2014 and then he served as the Vice-Chancellor at Bharath University from September 2014 till September 2017. He then served as Provost (Research) at Vinayaka Missions Research Foundation from September 2017 till September 2019 and from October 2019 till date he is serving as Provost at Bharath Institute of Higher Education and Research.