CHAPTER 1 INTRODUCTION

As internet users increases, content on the internet also increases. According to Netcraft January 2018 Web Server Survey internet websites approaching over 1.8 billion, Now every website has content that they call it unique. There are search engines like Google, Bing they are caching websites on a daily basis. Roughly estimation of data around 10 billion GB. Now, this data is not 100% unique. For example in figure 1.1 when we query “election 2019” on google, it gives around 1.5 billion(approx..) of search result which means there are 1.5 billion webpages that probably contains the same information.

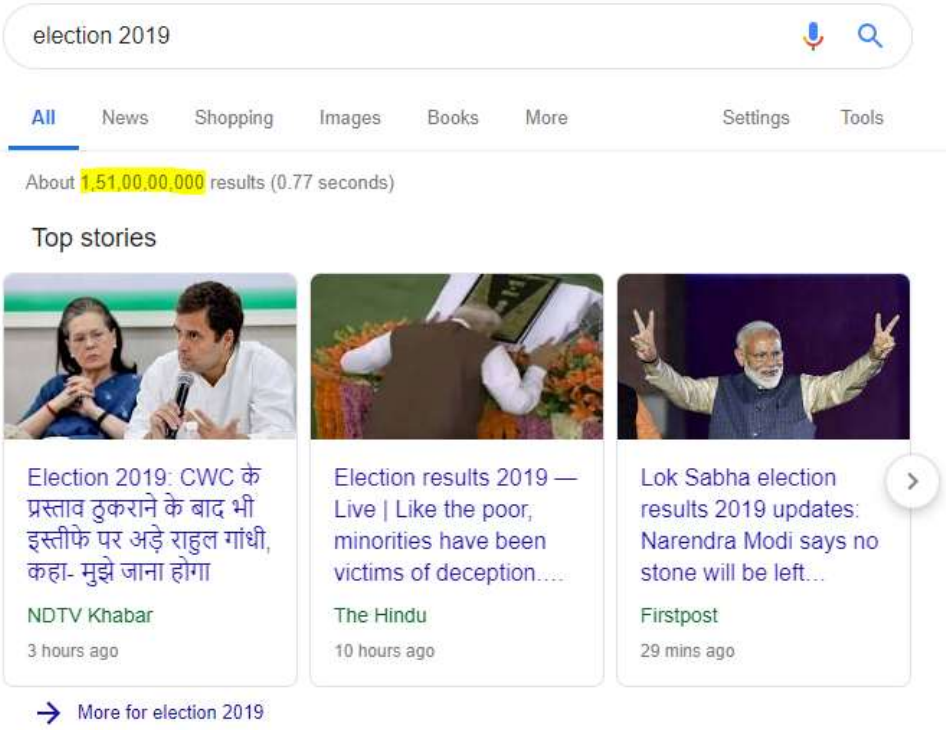

Figure 1.1: Google query results

That means there is billions of redundant data is flowing or the internet. And to store

and maintain the presence of this data investment and data disk are needed. Now, what if we can save this large data, instead of 1 billion textual data only 1 MB is needed to store. For that, we need to compare the data, but data is in the form of languages like English, Hindi, Chinese. In this research, the English language is used, as its most popular language. So to proceed with the research Natural language processing comes into the picture. So we need to build a model that can identify or predict the sentence similarity.

This can prediction model has very potential application like predicting the intention of user, this prediction will help to detect the fake NEWS, for creating more interactive artificial intelligent bot that will interact better respond better act better, for giving the better recommendations such as recommending music [1], prevent suicides [2], helps to build better A.I.(Artificial Intelligent) bot [3]. To achieve this it needs to be computational means if we subtract "some word" from "another word" we must get "some word" that contextually right [4]. For that, it needs to be stored in number so that we can easily perform mathematical operations for example 10 – 5 = 5. This is achieved by giving the weights to each and every words. This is again causing some discrepancy because it won't find the relation between two words. Let sentence 1 "Ram is drinking orange juice" and sentence 2 "Sham is drinking apple ???" corpus of this be ["Ram", "drinking", "orange", "juice", "Sham", "apple"]. Now to predict ??? we need to identify the context of the sentence. For that, we need vectorize on some features and present them into a multidimensional space.

Table 1.1: Dummy Featured representation through word embedding

	Ram	**Sham**	**Orange**	**Apple**	**Jobs**
Gender	-1	-1	0.01	0.01	0.00

Colour	0.05	0.10	0.25	0.50	0.01
Fruit	0.01	0.00	0.98	0.95	0.02
Employee	0.60	0.80	0.00	0.00	0.90
…	…	…	…	…	…
N features	…	…	…	…	…

Then corpus needs to be embedded then the prediction is done on the basis of some weights assigned. So according to table 1.1 there are two possibilities C1: [“Sham”, “Apple”, “Jobs”] and C2: [“Sham”, “Apple”, “Juice”], to predict which set is suitable we need to check with this C3:[“Ram”, “orange”, “Juice”]. Let C3 weight is 100, C1 weight is 30 and C2 weight is 80. With the logistic equation, we can say C2 has a higher probability than C1. So “Juice” is the most preferable option. We use this concept to predict the sentence similarity. The proposed model is based consist of four things word vectorization, represent the feature into multidimensional space, feature extraction through convolution1d and prediction.

1.1 THESIS ORGANIZATION

Chapter two gives the detail of the previously done work in this field and also the description of the enhancement techniques.

Chapter three covers the working of different models and their advantage are discussed.

Chapter four presents the proposed work.

Chapter fifth presents the results of our current approach and validates the results against ground truth.

Chapter Sixth concludes the thesis and further ideas for future work have been presented.

CHAPTER 2 LITERATURE SURVEY

2.1. Research and improvement of feature words weight based on TFIDF algorithm [5]

According to Aizhang Guo, Tao Yang among the various weight measuring algorithm in a document such as entropy function frequency function, Boolean function, The Term Frequency-Inverse Document Frequency(TFIDF) [6] gives better results. The main goal of TIFDF [7] model is to detect the important word for example title of the document, by taking the total count of word frequency divided by the total count of word frequency in the whole document this will give the value between 0 and 1, higher the value more important the word in the document and also it restricts the stopping words like 'is', 'the' due to Inverse Document frequency [5]. Let p be the term on which TfIdf is applied then $D_c(p)$ be the occurrence of p in the document, T_c be the total of all the term in the document, N_D be the total number of documents where $N_{D(p)}$ be the total number of documents with term p. Then term frequency of p will be,

$$\boldsymbol{t_f(p) = \frac{D_c(p)}{T_c}} \tag{2.1}$$

And inverse document frequency,

$$\boldsymbol{I_{df}(p) = log_e \frac{N_D}{N_{D(p)}}} \tag{2.2}$$

Then, TFIDF for p will be, using equation (2.1) and (2.2),

$$\mathbf{t_f(p) \times I_{df}(p) = \frac{D_c(p)}{T_c} \times log_e \frac{N_D}{N_{D(p)}}} \tag{2.3}$$

Although the major drawback of TFIDF is the inability to account the context of the word. Means it can't be used to predict the corresponding word from the corpus. For

that, we need word embedding word vectorization.

2.2. Inferring Affective Meanings of Words from Word Embedding [4]

Minglei Li, Qin Lu proposed a regression model to infer the effective meaning on the basis of word embedding. They used a set of seed words on which they vectorized the corpus and then represent it in multidimensional space. Each word consists set of weight [8] on which they can apply mathematical operations to detect the probability in similar words, after this they applied existing models on which they conclude Ridge, the Bayesian Ridge and Support Vector Regression model with linear kernel give better results.

Table 2.1: Comparison of TFIDF and Word Embedding

TFIDF	Word Embedding
Creates one number per word	Creates one vector per word
Good for classification of documents as a whole	Good for identifying contextual content

They also test these models on various criteria like performance, accuracy, feasibility. On which they found Ridge regression model is best suited for their proposed framework.

Let, s be the seed words its corresponding word embedding $\overrightarrow{w^s}$,

$$\overrightarrow{\boldsymbol{w^s}} = [\boldsymbol{e_1^s} + \boldsymbol{e_2^s} + \cdots + \boldsymbol{e_n^s}] \tag{2.4}$$

And m_i be the mapping function for the i^{th} affective dimension,

$$\boldsymbol{m_i}(\overrightarrow{\boldsymbol{w^s}}) = \boldsymbol{g_i}(\boldsymbol{b_1^i e_1^s} + \boldsymbol{b_2^i e_2^s} + \cdots + \boldsymbol{b_n^i e_n^s}) \tag{2.5}$$

Where b_j^i is the weight of feature j and g_i mapping function.

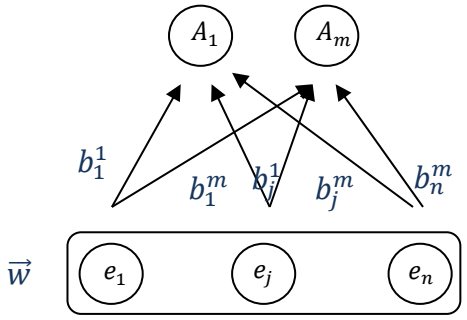

Figure 2.1: Regression method for affective representation

The affective meaning is defined by, using s seed words as a training sample in m dimensional space. With word embedding, this is of n size vector. This model can predict through word embedding the affective value of a new word.

2.3. Two Improved Continuous Bag-of-Word Models [9]

Representing word relatively is an important and fundamental task in natural language processing [10]. Continuous Bag of words(CBOW) is a technique by which one can predict the particular word in the corpus by looking the adjacent words in a sentence. Continuous Bag of words comes under prediction based word embedding and TFIDF comes under frequency based that's why for determining the relation between two words continuous bag of words is used instead of TFIDF because it just tells the frequency. To predict the "context" or "relation" between the words it creates the sliding window on the word. It is used to predict one word from the entire corpus. For example: consider this sentence "The rat sat on the table", context word in the example will be "the", "rat", "on" and "table", to predict the "sat". Continuous Bag of words is probabilistic in nature, it is supposed to perform better to deterministic methods [11].

CBOW makes the prediction on the target word t_w as the h hidden layer obtained,

$$\boldsymbol{p(t_w|C) = exp(e'(t_w)^T h) \Big/ \sum_{t'_w \in V} exp(e'(t'_w)^T h)} \quad (2.6)$$

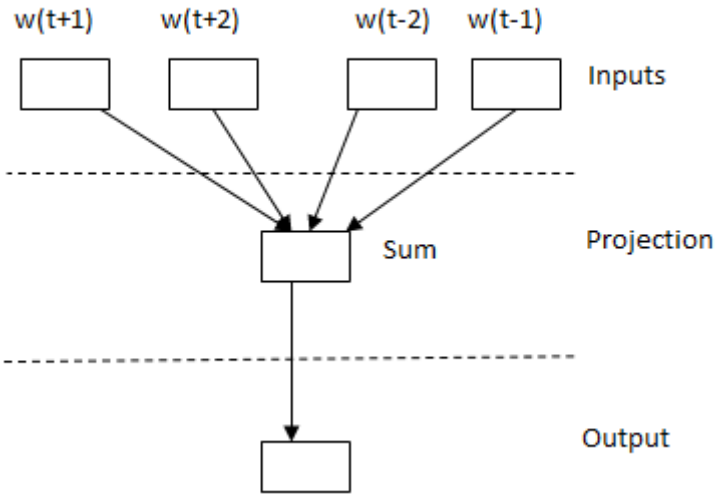

Figure 2.2: Continuous Bag of the word network

Where e' be the embedding and C is the context of t_w. They improved the CBOW by introducing position encoding which provides the order of missing word. It uses the co-occurrence matrix to create a vector for each word.

2.4. Weighted Word2vec Based On The Distance Of Words [12]

Mikolov et al proposed word2vec an application of natural language processing and released by Google in 2013 which consider being very important in natural language processing [13] [14] [15]. word2vec is built from two things Skip gram model and Continuous Bag of Words(CBOW). CBOW perform better in Training Speed as compare to Skip gram, but in case of accuracy is skip gram performance is higher. They majorly worked on CBOW by introducing the fuzzy weight to influence the distance between words in CBOW architecture. Also, they used different membership functions like Gaussian, trapezoidal and triangle to show the slope and also they show it depends on the window size.

2.5. ConceptVector: Text Visual Analytics via Interactive Lexicon Building using Word Embedding [16]

As TFIDF fails to relate the corresponding words, word embedding is introduced that gives the probability of the related words in the corpus by representing the words in the high dimensional space through vectorization [17]. which can also deduce the relativity of the words for example "queen - woman + man = king" this can be achieved from word embedding [18].

2.6. Vector Representation of Words for Sentiment Analysis Using GloVe [19]

Standford proposed the Global vectors(GloVe) which work on the two algorithms Continuous Bag Of Word(CBOW) and skip gram model, which able to generate word vector in a fixed dimension. By which they assume two words are considered as same meaning if they share the same context.

CHAPTER 3 TECHNIQUES & METHODS

3.1. Data Analysis

Data analysis is the most important task in natural language processing. The main motive of data analysis to determine the complexity of data, how the data is organized, is the data is authentic, is data contains a required field to evaluate the results, finding the length of sentence or data contains more missing values that we cannot use all these things lay under the data analysis. This research data analysis results are in the fourth chapter.

3.2. Data Pre-Processing

3.2.1. Handling Missing Fields

As per the data analysis result, there were no missing data field found, So there's no need to tackle missing values or field, but not all data sets are same like this research dataset so to tackle missing field problem there's need to delete the entire row/sentence because it will give false results.

3.2.2. Data Case Conversion

To make the data words more common, there's a need to convert the entire data into the upper case or lower case in this research data is converted into the lower case. For example: ["Why this is so hot here?", "I am Feeling to HOT!! why"], in this example problem is that machine will treat "hot" and "HOT" both words differently because of ASCII convention. To make it even it's needed to convert it into lowercase or uppercase.

3.2.3. Removing Irrelevant Words

After data conversion there's need of removing the common irrelevant words that we called it stopping words for example: "the", "a", "an", "in", these words contains doesn't affect the end result, also if we keep these stopping words the end result will give poor results.

3.2.4. Keeping AlphaNumeric

After removing irrelevant words from the data, there's a need to remove the special characters only keeping the alphanumeric. This will decrease the complexity of the data set and improve the prediction.

3.2.5. Building Corpus

Now each sentence after above data pre-processing steps corpus is created by splitting the sentence against the white spaces and stored it into the array like: ["why", "this", "is", "hot", "here"]. Which is further divided into S_{train} and S_{test} , training and testing dataset respectively.

3.3. Corpus Representation

3.3.1. TFIDF Matrix

For simplification in computation, corpus needs to be converted into the matrix form. TFIDF [20] is the fastest way to convert a document into a matrix. The whole process of Term Frequency-Inverse Document Frequency is written in chapter 2. Also, it supports longer sentences or documents, but it can be resolved through normalization. But this matrix doesn't perverse the semantic relationship between words, it can only deduce the most relevant word in the corpus it's like one hot encoding. For example, consider two

similar words w1 and w2 but from one hot encoding representation placements of '1' is indifferent position, that means their dot product will give '0' which is false.

3.3.2. Embedding Matrix

Embedding in natural language processing for discrete variables referred to as low dimensional continuous vector representations. We can generate Embedding matrix from one-hot encoding but in that case there dot product will give zero that doesn't provide any meaning. So we need to generate meaningful embedding that can be used as input to the prediction model. For instance let suppose ["Actor", "Actress", "Girl", "Boy"]. Now if we subtracts "Actress" from "Actor" then add "Boy", which results in "Girl" corresponding vectors.

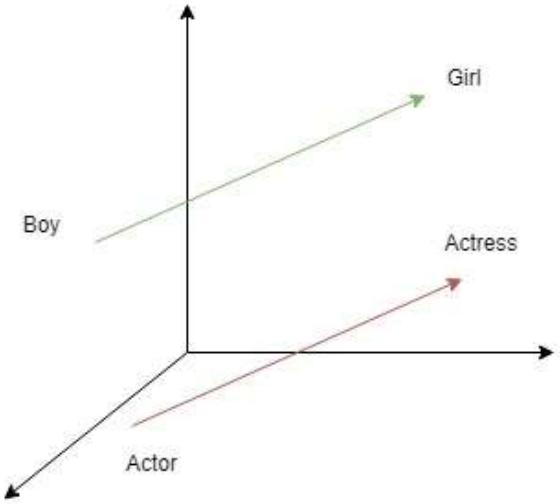

Figure 3.1: Word vector representation

In that case we need context of each and every word in the corpus, consider m and n be the two words that generates some meaning in particular interest and E be the matrix consist of word – word occurrence, E_{mn} be the occurrence of n in context of m. Total number of times arbitrary word from the corpus appears in the context of m will be E_m.

$$\boldsymbol{E_m = \sum_k E_{mk}} \tag{3.1}$$

Let p_{mn} be the probability of n in context of m.

$$\boldsymbol{p_{mn} = p(n \mid m) = \frac{E_{mn}}{E_m}} \tag{3.2}$$

Embedding matrix consist of words of corpus with their related context measure in terms of probability. Let $w \in R^d$ are word vectors and $\breve{w} \in R^d$ are separated context word vectors.

$$\boldsymbol{f(w_m^T \breve{w}_k) = p_{mk} = \frac{E_{mk}}{E_i}} \tag{3.3}$$

So, $f = exp$ then

$$\boldsymbol{w_m^T \breve{w}_k = log(p_{mk}) = log(E_{mk}) - log(E_i)} \tag{3.4}$$

In order to get the vectors with minimum loss weighting function $f(E_{mn})$ is introduced into the cost function J [21]. Let V be the size of corpus and $\log(E_m)$ is considered as b_i bais, so loss function will become,

$$\boldsymbol{J = \sum_{m,n=1}^{V} f(E_{mn}) \, (w_m^T \, \breve{w}_n + b_i + \breve{b}_n - log(E_{mn}))^2} \tag{3.5}$$

Where,

$$\boldsymbol{f(x) = \begin{cases} (x \mid x\_\max)^{\alpha} \; if \; x < x\_\max \\ 1 \, , otherwise \end{cases}} \tag{3.6}$$

This will generate the Gobal vectors. We are using predefined GloVe [19] vector to generate the context associated with each word in the sentence. GloVe vector contains 840 billion tokens of 300 dimension, So to generate embedding matrix, we need to tokenize our sample space called it as a corpus in figure 3.1.

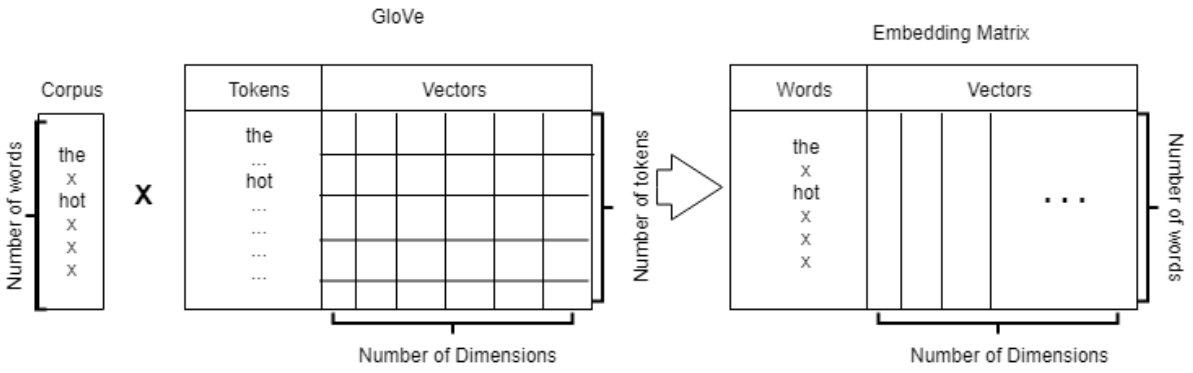

Figure 3.2 Embedding Matrix Generation

After that for each word in the corpus will be matched in GloVe vector and assign those values in the embedding matrix, which contains a total number of words as rows and a total number of dimensions as columns. It uses Eigen which helps to denote the semantic relationship between two words.

$$\boldsymbol{E} \times \vec{\boldsymbol{v}} = \boldsymbol{\mu} \times \boldsymbol{v} \tag{3.7}$$

Where E be the matrix, $\vec{v}$ be the vector, μ scalar co-efficient. Eigenvector corresponds to the matrix as though that matrix were a scalar coefficient. If there is a square matrix of $n \times n$ then there will be n eigenvectors.

To get the mean of the values, simply take the average value of all the v and divide it with the sum of all data points i.e., n

$$\bar{\boldsymbol{v}} = \frac{\sum_{j=1}^{n} v_j}{\boldsymbol{n}} \tag{3.8}$$

Then the square root of average square distances between data points to its mean will produce the average deviation of the results

$$\boldsymbol{s} = \sqrt{\frac{\sum_{j=1}^{n}(v_j - \bar{v})^2}{\boldsymbol{n-1}}} \tag{3.9}$$

Then the measure of data spread is

$$\boldsymbol{\sigma} = \boldsymbol{s^2} \tag{3.10}$$

This matrix consists of the probability of similarity which can be build from pre-trained vector like GloVe [19].

3.4 Models

3.4.1 Multi Linear Regression(LR)

Linear Regression with multiple variables in the prediction model [24], which is used to deduce the relationship between features to predict. In this method, features are scattered in two-dimensional plane two generate hypothesis for new values.

$$\boldsymbol{H = b_0 + b_1 \times X_1 + b_2 \times X_2 + \cdots + b_n \times X_n} \quad (3.11)$$

Here, H is the hypothesis by multilinear regression b_0 is bais, $X_1, X_2 \ldots X_n$ are the features word vector. A two reduce the difference between hypothesis and the actual output y, let L will be the loss function in order to minimize it we used gradient descent.

$$\boldsymbol{L(b_0, b_1, \ldots, b_n) = \frac{1}{2m}\sum_{i=1}^{m}(H_b(X^i - y^i))^2} \quad (3.12)$$

We used gradient descent to optimize parameters of our cost function so that we can find more suitable values of parameters for our hypothesis.

$$\boldsymbol{b_j = b_j - \alpha \frac{\partial}{\partial b_j} L(b_0, b_1)} \quad (3.13)$$

Before applying the multilinear regression we need to assume linearity, multivariate normality, homoscedasticity, lack of multicollinearity, independence of error. Multi-linear regression will work on any size of data set. Multilinear regression model induces the relation between two or more features.

3.4.2 Support Vector Machine Regression(SVR)

SVR is a non-parametric method because it relies on kernel function [25]. Where in multi-linear regression is focusing on reducing error, while SVR is to fit the error within a certain threshold, SVR is capable of working on data set of any size, linear and non-linear. SVR creates a hyperplane that divides the dataset into two.

$$\boldsymbol{P_0 + (P_1 * X_1) + (P_2 * X_2) = 0} \quad (3.14)$$

Where (P_1 and P_2) are coefficients which tell the slope of the line and the intercept (P_0) are found by the learning algorithm, and X_1 and X_2 are the two input variables.

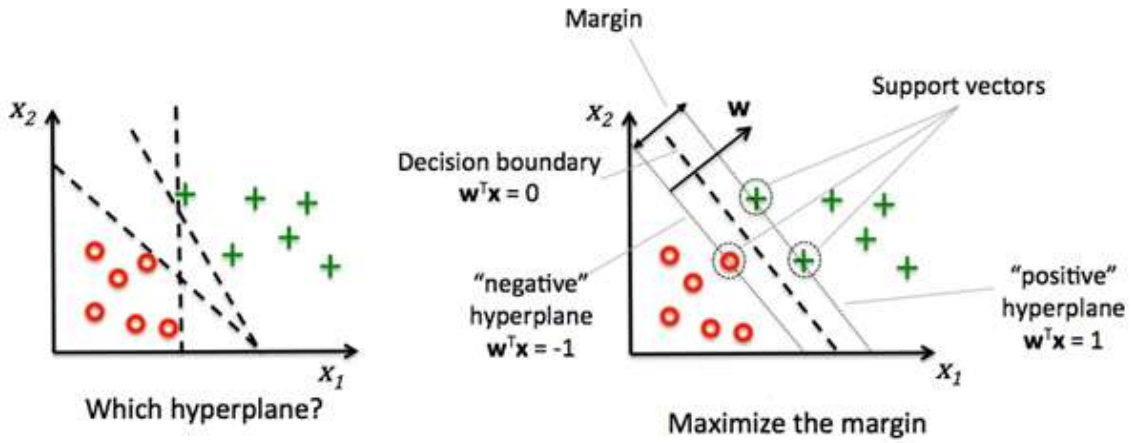

Figure 3.3: SVR Hyperplane

3.4.3 Random Forest(RF)

Random forest algorithm is "Ensemble Learning algorithm", in which decision is made by taking multiple decision trees to create the forest and then combine [26]. Depending on the intensity level decision tree splits the node.

$$\boldsymbol{Info(D) = \sum_{i=1}^{c} -p_i \, log_2 \, p_i} \tag{3.15}$$

Entropy is the measurement of homogeneity in the data. Pi is the probability of arbitrary tuple in D belongs to Class C_i.

$$\boldsymbol{Gain(A) = Info(D) - \sum_{j=0}^{v} \frac{|D_i|}{|D|} | * Info(D)} \tag{3.16}$$

Random forest prediction model, take k data points for the given training data, then build there associated decision tree. While choosing the N number of tree to build the tree and repeat this procedure until it commences. For new data point, make each one of your N trees predict the value of y to for the data point in sentence which assigns the new data points average across all the predicted y values.

3.4.3 Match-LSTM

LSTM(Long Short-Term Memory Networks), is a famous neural network for remembering long term dependency. LSTM is a powerful recurrent neural network (RNN), and a RNN is built to handle sequence dependency. Now, what makes RNN as powerful as tuning machine is it has feedback connection back to the network, which makes it self-adjusting neural network. Hence LSTM being powerful most valuable AI achievement used in classification, decision making to music composition as well.

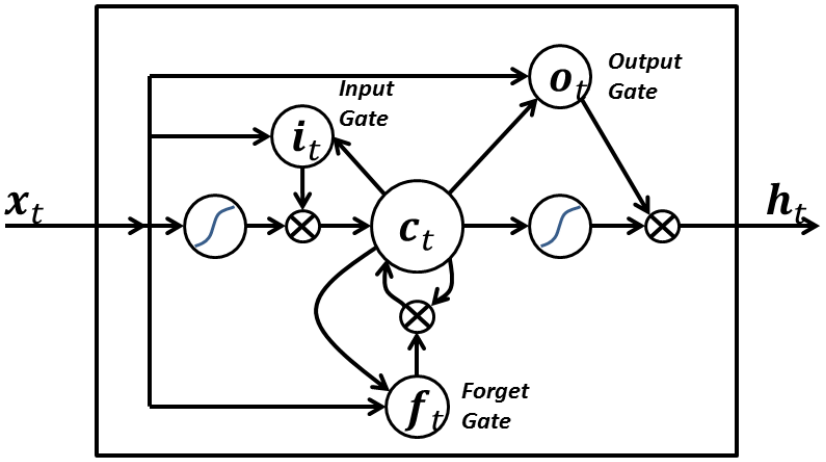

Figure 3.4: LSTM architecture

In LSTM, the smallest unit is composed of a cell, input-output gates and a forget gate. A cell is memory unit in it, and flow of information is regulated via these gates in and out.

$$\boldsymbol{O^t = f(h^t;\theta)} \tag{3.17}$$

$$\boldsymbol{h^t = g(h^{t-1}, x^t;\theta)} \tag{3.18}$$

To the evolution of RNN, let O^t be the RNN at time t, h^t be the hidden layers at time t with input x^t. In Match-LSTM(mLSTM) [24] model match word vector of two sentences i.e., the premise p_i and hypothesis h_i. mLSTM architecture is base on LSTM. In the model instead of directly matching two sentence embedding at once they matched word by word.

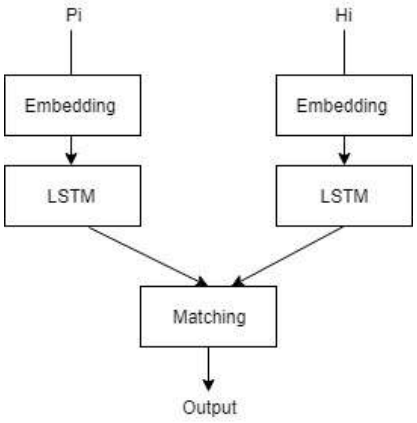

Figure 3.5 Match-LSTM architecture

This architecture one by one process the premise p_i and hypothesis h_i. The mLSTM sequentially went through all the word vectors of the h_i and this representational vector of the current token of h_i will be feed into mLSTM. Then mLSTM will aggregate the matching of the weighting p_i to each token of the h_i. After aggregated using LSTM into a vector so that it will give the final result.

3.4.4 Convolution Neural Network(CNN)

As the data is in textual format, for faster retrieval and better results convolution 1D neural network is suitable for this purpose [27]. For convolution neural network works on the mechanism of sliding window, it not only extracts the relevant feature but also extracts the feature. Here data is input in the form of embedding matrix.

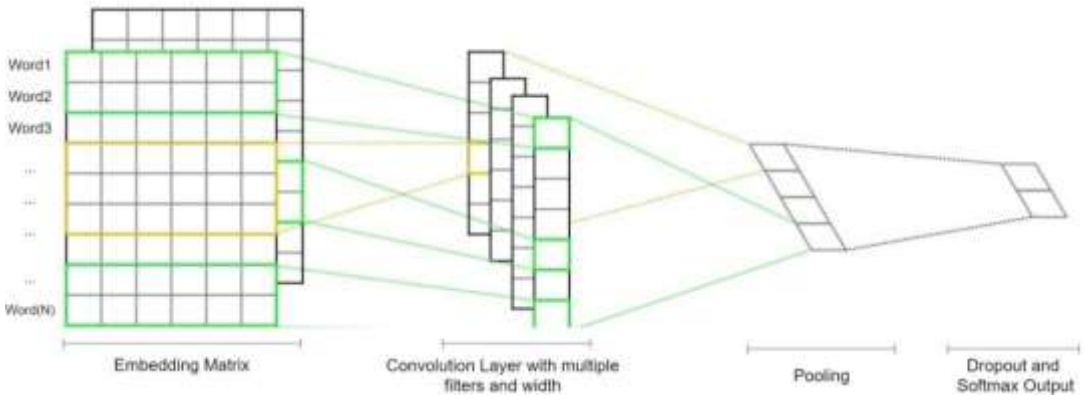

Figure 3.6: Convolution Neural Network 1 Dimensional

Convolution Neural Network consists of two things generally convolution and pooling. Consider convolution as a sliding window or a kernel by which feature extraction can be done. This convolution filter slides through the sentence and extracts the patch from the corpus of the sentence this will be dot product and saved it in the output feature vector.

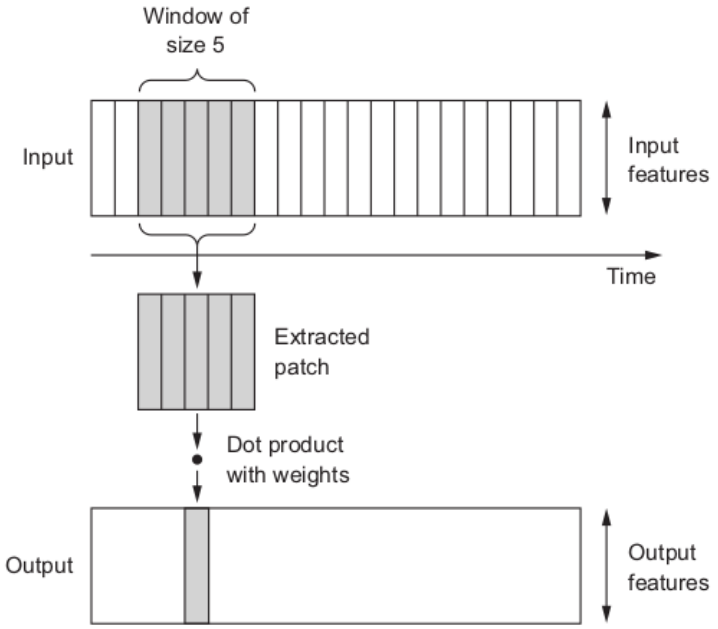

Figure 3.7: Convolution Sliding Window Working

After convolution, pooling is done, there are mainly two types of pooling max pooling and min pooling. For the max pooling, it takes a maximum of among all the corpus words. And the min pooling will take the minimum from the patch.

For the given 1 dimensional array of words in $W_i = w_1, w_2, \dots, w_N$, 1D convolution neural network is used for feature extraction. Where every word in the W_i is associated with an embedding vector of D dimensionality. Let, R be the convolution filter or kernel and K be the sliding window width size which slides over the sentence S_i. Now applying Convolution1D on each window which is dot product of weighted vector u and embedding vector E.

Let δ be the activation function, consider $w_j, w_{j+1}, \ldots, w_{j+k}$ be the window of words. Then z_j be the concatenated vector of j^{th} window.

$$\boldsymbol{z_i} = \left[\boldsymbol{w_j}, .., \boldsymbol{w_{j+3}}, .. \boldsymbol{w_{i+k}}\right] \in \boldsymbol{R^{k \times d}} \tag{3.19}$$

R is then applied on the k width window, then

$$\boldsymbol{r_i} = \boldsymbol{\delta}\left(\boldsymbol{z_j}.\boldsymbol{u}\right) \in \boldsymbol{R} \tag{3.20}$$

Pooling is used to remove the number of feature map coefficient and to process it well to induce the spatial filter. This defines the pooling and the convolution as a feature extractor.

3.4.5 Batch Normalization

Vignesh Thakkar, Suman Tewary [28] shows the importance of batch normalization in terms of performance and in terms of capability to train the model. With the help of Batch Normalization neural network become for reliable, fast and scalable. Its major advantage is to increase the performance of the model and to make the training process very fast. It is applied in between the convolution neural network and the activation function.

$$\widehat{\boldsymbol{v}_{\boldsymbol{l}}} = \frac{\boldsymbol{v_i} - \overline{\boldsymbol{v}}}{\boldsymbol{s}} \tag{3.21}$$

Where s be the standard deviation and l be the dimensionality. The performance of the model increased due to a reduction in the internal covariate shift i.e, change in the distribution of inputs.

In addition, the parameters or scale from the original value to less are dependent on gradients and the higher discrepancies and data acquisition rates allow for saturation.

3.4.6 Rectified Linear Unit(Relu)

ReLu is the activation unit that is applied after the batch normalization [29], as our data

output contains only positive value we need output 0 when x is smaller than 0 and x when x is bigger than zero.

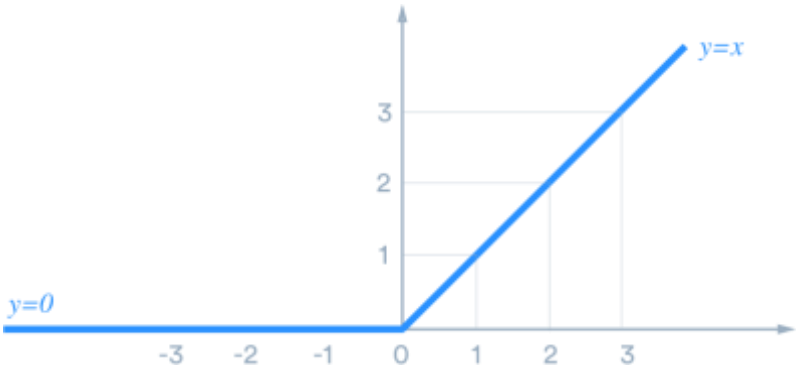

Figure 3.8: Working of Rectified Linear Unit

So Rectified Linear Unit is defined as y,

$$\boldsymbol{y = max(0, x)} \tag{3.22}$$

The major advantage of using ReLu is that it converges very fast and also simple to use.

3.5.7 Dropout

To prevent the overfitting we are using dropout of 25% that will be used for regularization and prevent the overfitting. In such a case, the model is not considering particular neurons and the training speed will drastically increase. We are using dropout after the activation function in the processing unit.

3.4.8 Flattening

Flattening is used after the dropout so that it breaks the spatial structure of the data and transforms your tridimensional tensor into a mono-dimensional tensor. In our case, it's converting two column array into one column array.

CHAPTER 4 PROPOSED APPROACH

In this research, we make a good use word embedding, to get the weights of particular words in vector to get Embedding Matrix. This Embedding matrix will further used to predict the sentence similarity through different machine learning techniques. In this section, we modeled our own approach i.e., Convolution over Word Embedding CoWe.

4.1 Data Preprocessing

The first step in this research is Data Preprocessing, which consist of two things Data Analysis and Data Cleansing. In order to perform Data Cleansing process we need to see how data looks like, is the data contains meaning or it's data, number of rows, how data is distributed, how many sentences are duplicate in training set, after considering all these things we need to clean the data which contains removing empty rows, stemming the data, keeping only alphanumeric data, converting the data into lower case.

4.2 Training And Prediction

In this section, we use embedding matrix which consists of all the words that contain in corpus with there corresponding probability from 0 to 1 that determine context between the words. These word vectors are used as an input two our model. This model is taking four input parameters shown in figure 4.1 i.e., word vectors of sent training set sentence 1, training set of sentence 2, training set of sentence 1 and training set of sentence 2. These embedding is passed through respective input units, we initialize

weights of node i.e, mean 0.0, standard deviation 0.05 and seed 2. In each layer, we are using a number of filters 32, filter length 3 and dropout of 25%.

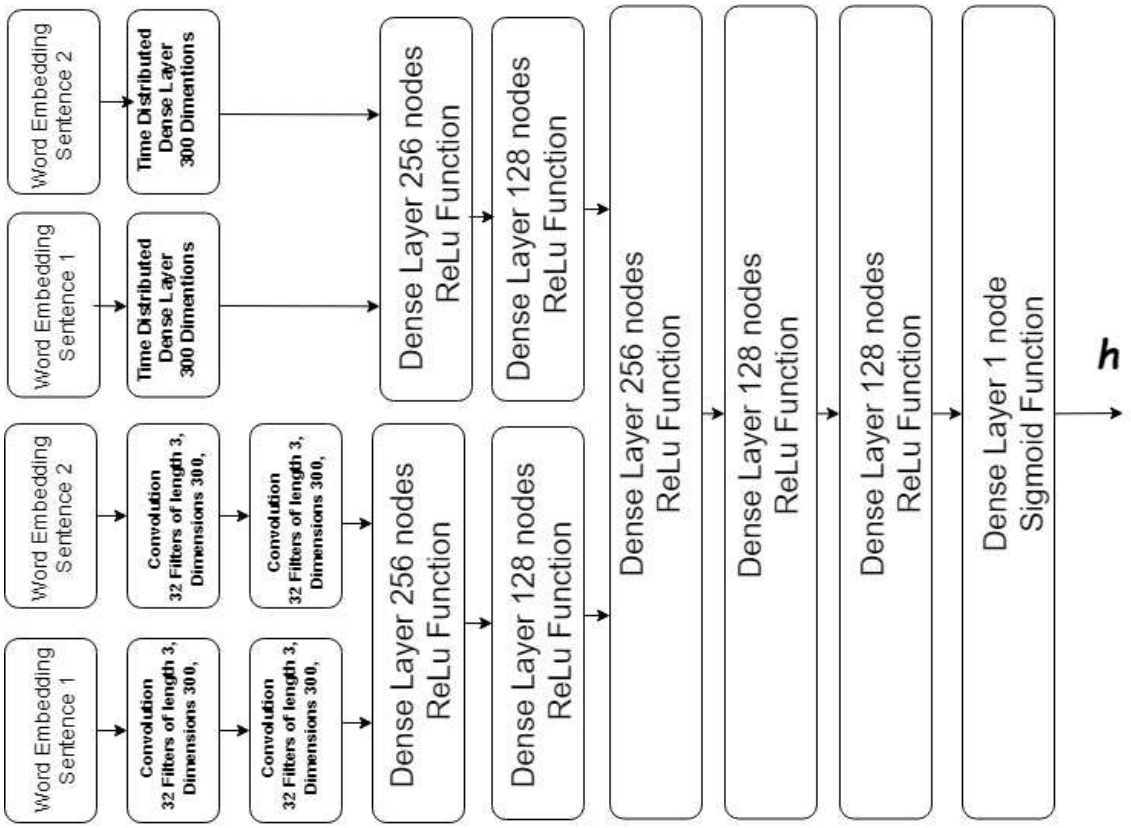

Figure 4.1: Proposed Prediction model (CoWe)

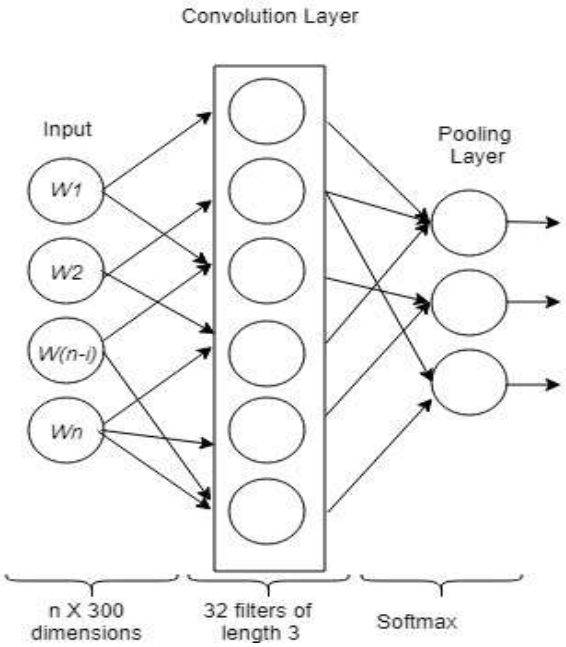

Figure 4.2: Convolution layer

We are wrapping CNN layer with time distributed dense layer of 300 dimensions that will apply a dense layer on every timestamp which gives required intersecting values between its own timestamp. For the given 1 dimensional array of words vectors in $W_i = w_1, w_2, \dots, w_N$, 1D convolution neural network is used for feature extraction. Where every word in the W_i is associated with an embedding vector of D dimensionality. Let, R be the convolution filter or kernel and K be the sliding window width size which slides over the sentence S_i. Now applying the Convolution1D on each window which is dot product of weighted vector u and embedding vector E.

Let δ be the activation function, consider $w_j, w_{j+1}, \dots, w_{j+k}$ be the window of words. Then z_j be the concatenated vector of j^{th} window.

$$\boldsymbol{z_i} = \left[\boldsymbol{w_j}, .., \boldsymbol{w_{j+3}}, .. \boldsymbol{w_{i+k}}\right] \in \boldsymbol{R^{k \times d}} \tag{4.5}$$

R is then applied on the k width window, then

$$\boldsymbol{r_i} = \boldsymbol{\delta}\left(\boldsymbol{z_j}.\boldsymbol{u}\right) \in \boldsymbol{R} \tag{4.6}$$

Each unit of convolution is using ReLu activation function, in which output contains only positive value we need output 0 when x is smaller than 0 and x when x is bigger than zero.

$$\boldsymbol{y} = \boldsymbol{max}(\boldsymbol{0}, \boldsymbol{x}) \tag{4.7}$$

To regularize the output of each layer dropout of 25% is used to minimize the overfitting. As output consist of two arrays after combining the two layers, thus we used dense layer of 128 x 2 nodes after then we applied dense layer of 128 nodes. At last to deduce the similarity we need 1 output node which gives 0 for not similar and 1 for similar. For that purpose we used Sigmoid function,

$$\boldsymbol{S} = \boldsymbol{1}/(\boldsymbol{1} + \boldsymbol{e^{-h}}) \tag{4.8}$$

This will give either 0 or 1. Then we will check against our testing set.

CHAPTER 5 EXPERIMENTAL RESULTS

The following system configuration has been used while conducting the experiments:

- Processor: Intel Core i5 4th Generation
- Clock Speed: 1.60 GHz
- Main Memory: 8 GB
- Hard Disk Capacity: 1024 GB
- Software Used: Jupyter Notebook

In this research, we are using a dataset that consists of 50,000 rows consist of 5 columns in which 2 columns contain 2 sentences and 1 tells the two sentences is duplicate or not. In this section, we performed an experiment on to evaluate our proposed model i.e., CoWe with Multi-linear regression(LR), Random Forest(RF), Support Vector Machine Regression(SVR) and MaLSTM.

5.1 Data Analysis

Data consist of 31350 unique sentences and 18649 duplicate sentences. For the below given graph 0 denotes unique sentences and 1 represents duplicate sentences.

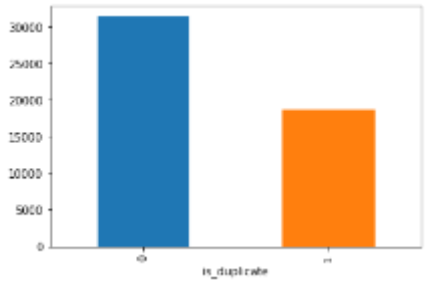

Figure 5.1: Duplicate graph visualization

Data set contains, 49999 total number of sentence pairs for training, 37.3% of duplicate pairs.

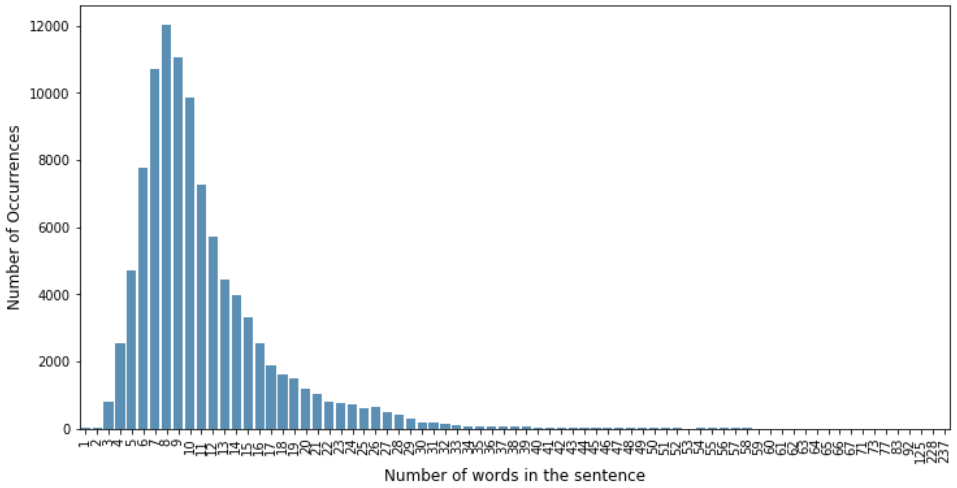

Figure 5.2: Number of occurrence of words

This graph shows the number of occurrences of particular words in a sentence. Figure 5.2 shows mostly there are 8 – 9 words in each sentence. This analysis is very important when we are making a matrix of each sentence to make each one of the same lengths.

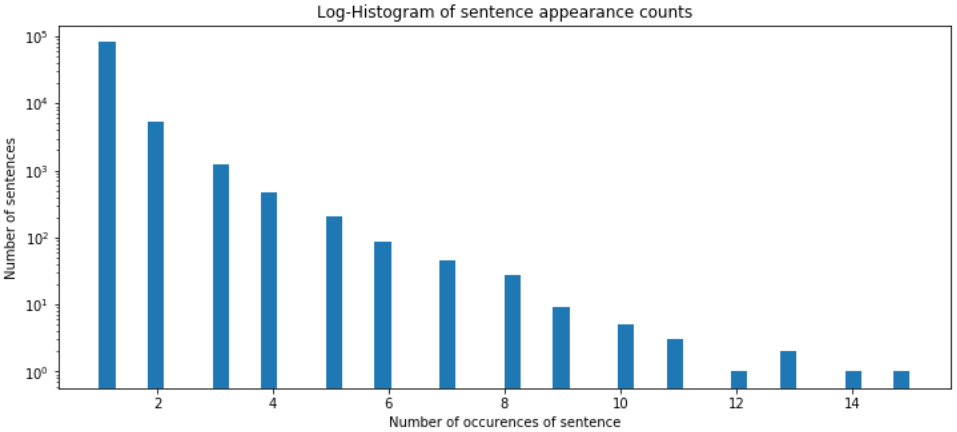

Figure 5.3: Log-Histogram of sentence appearance counts

Figure 5.3, shows the Log-Histogram of sentences appearance counts, which define a number of sentences versus a number of occurrences of the sentence in this data set.

Figure 5.4, shows the normalized histogram of character count in each sentence, this character counts will tell the data distribution in the data set, this will help in

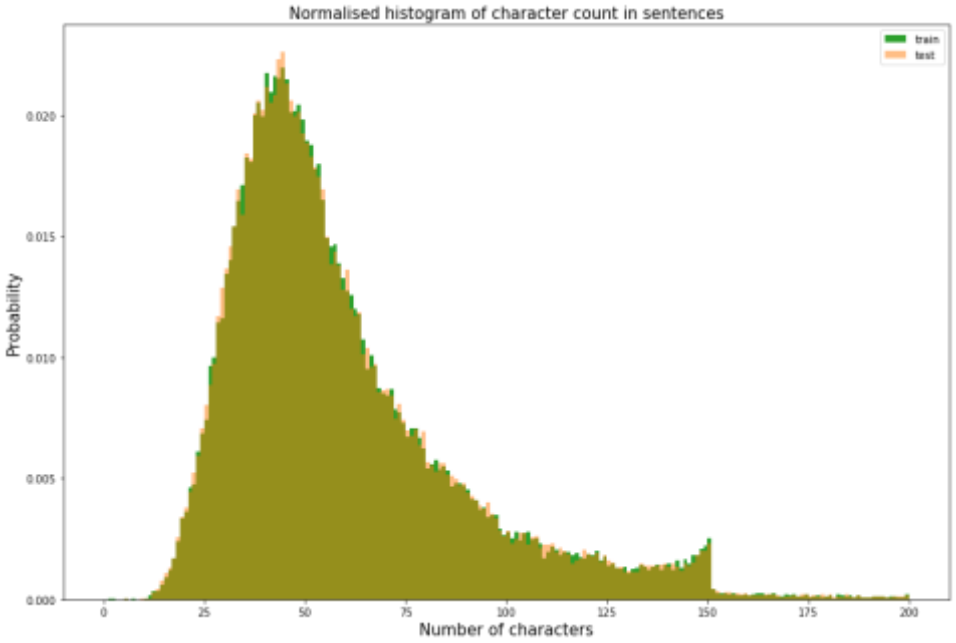

Figure 5.4: Normalized Histogram of character count in sentences

determining the illogical sentences. This figure shows the probability versus a number of characters.

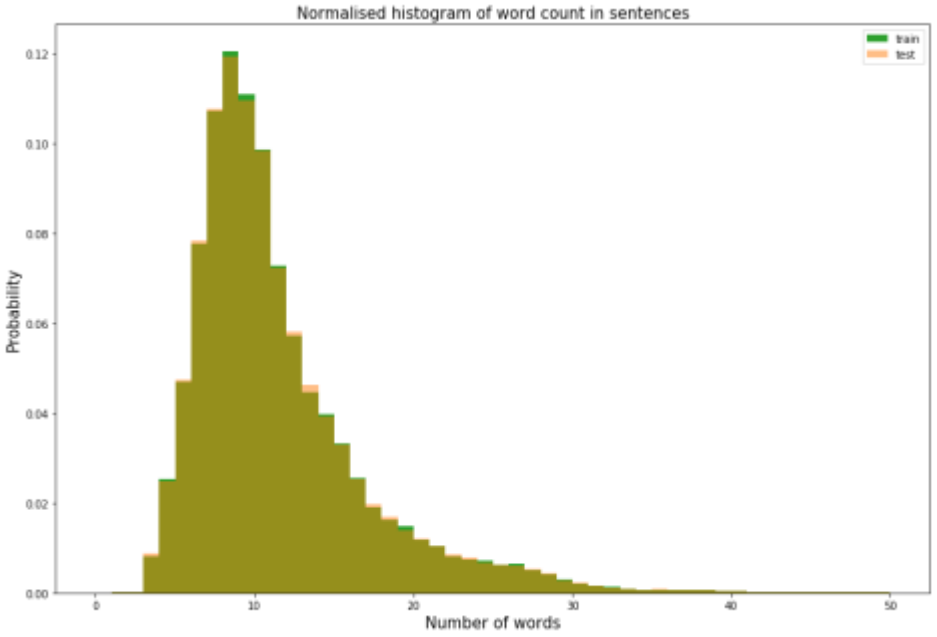

Figure 5.5: Normalized Histogram of character count in sentences

Similarly, figure 5.5 shows the normalized histogram on the character count in the sentences, this will be again used with the character counts both figure 5.4 and figure

5.5 is used to deduce the validity of data set by check its total characters divide by the total words.

5.2 Evaluation

For the evaluation, we will first create the embedding matrix using pre-defined GloVe vectors, GloVe vectors consist of pre-defined vectors of 300 dimension and 840 billion tokens. To generate the word embedding, first, we transform each sentence into fixed sized sentences and make it as a corpus in the array. For example: consider two sentences S1: “What a good weather” and S2 “this is good”, Now we will generate this in corpus of sentences: [[“what”, “a”, “good”, “Weather”],[“this”, “is”, “good”, NULL]]. Now, this corpus is converted into the vectors by assigning the weights through GloVe vectors.

5.2.1. Word Embedding Evaluation

Word Embedding graphical representation of generated word vectors are represented in figure 5.6 which is the 2-dimensional representation.

Which can inference meaning of word “universe” in terms of probability, This shows ‘created’ is having 94.69% probability in terms of associativity rule.

```
[('created', 0.9469516277313232),
('dark', 0.9097416400909424),
('space', 0.9094343185424805),
('infinite', 0.9081777334213257),
('matter', 0.8620973825454712),
('point', 0.8504949808120728),
('energy', 0.8401374220848083),
('bang', 0.830245852470398),
('evidence', 0.7979810237884521),
('limit', 0.7881182432174683)]
```

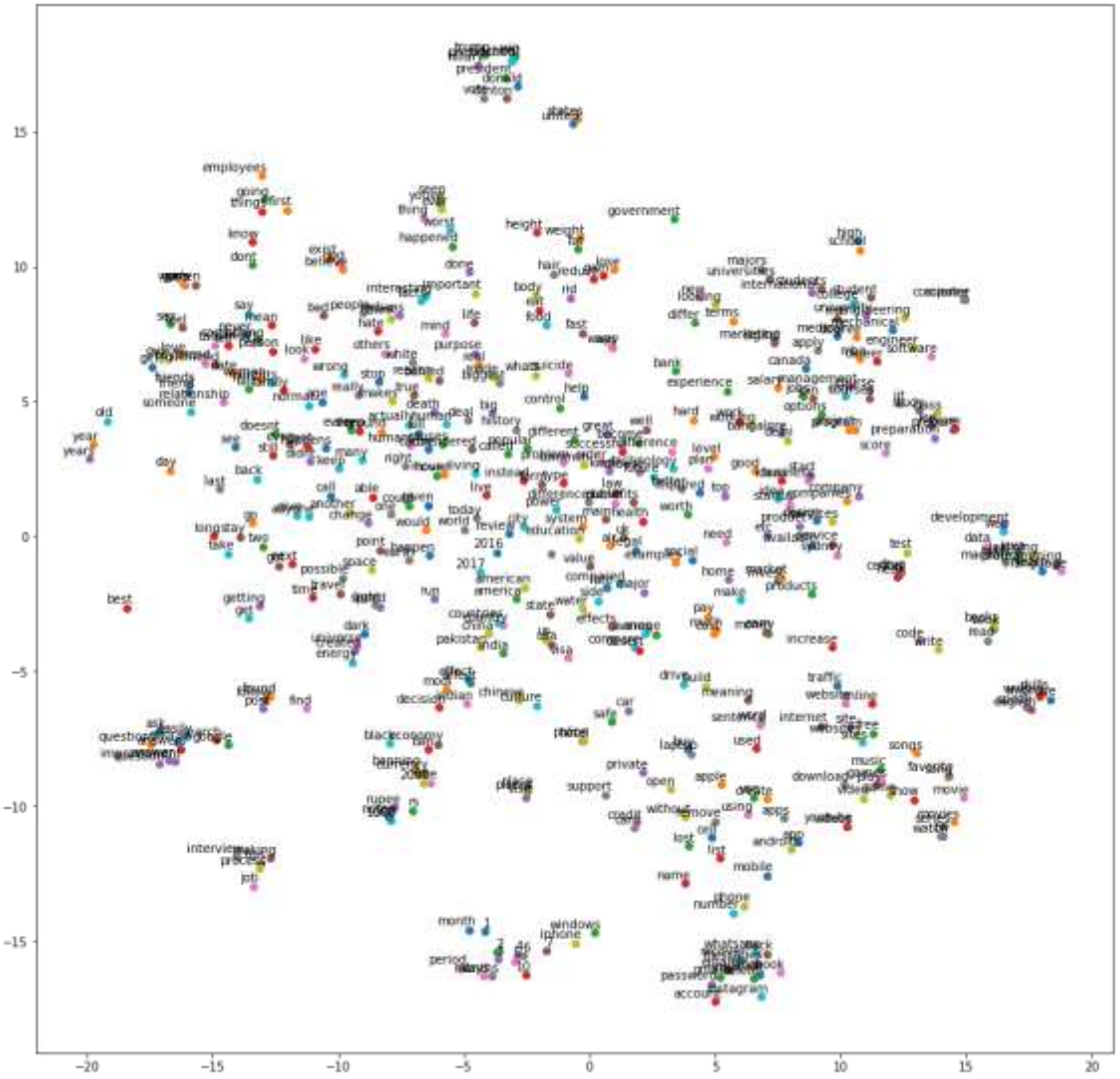

Figure 5.6: Plotting word vectors in the 2-dimensional space

From this, we can actually relate words with others in terms of direction and also deduce the meanings i.e, probability of similarity between two words. You can see in the bottom-middle of Figure 5.6 the word “windows”, “iPhone” comes together which means in the given corpus of words these words are used with each other more often that’s why the distance between these two is very less.

5.2.2. Model Evaluation

For prediction evaluation we are using the following measurements matrices:

1) Accuracy

$$\boldsymbol{AUC} = \frac{\boldsymbol{C_n}}{\boldsymbol{N}} \tag{5.1}$$

2) Root Mean Squared Error(RM)

$$\boldsymbol{RM} = \sqrt{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \frac{(\boldsymbol{O_i - p_i})^2}{\boldsymbol{n}}} \tag{5.2}$$

3) Mean Absolute Error(MA)

$$\boldsymbol{MA} = \frac{\boldsymbol{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} |\boldsymbol{O_i - p_i}| \tag{5.3}$$

4) Mean Absolute Percentage Error(MAPE)

$$\boldsymbol{MAPE} = \frac{\boldsymbol{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} |\boldsymbol{O_i - p_i}| \times \boldsymbol{100} \tag{5.4}$$

Where C_n be the total number of correct predictions, N be the total predictions made, $O_i$be the original value and p_i be the predicted value.

Table 5.1: Evaluation of different techniques

S.No.	Model	AUC	RM	MA	MAPE
1	LR	63.26	60.60	0.367	36.73
2	RF	62.70	61.06	0.372	37.29
3	SVR	62.81	60.98	0.371	37.18
4	MaLSTM	77.67	13.95	0.280	28.09
5	CoWe50	74.13	15.04	0.309	30.90
6	CoWe100	74.81	15.46	0.321	32.13
7	CoWe200	75.12	13.03	0.276	27.63
8	CoWe300	83.72	12.50	0.272	27.26

Above, Table 5.1 represents the evaluation of different models, where LR stands for multi-Linear regression, RF for Random forest, SVR for Support Vector Machine Regression model, and CoWe for our proposed model for Sentence similarity with 50 dimensional, 100 dimensional, 200 dimensional, 300 dimensional. For this table 5.1, we conclude our model CoWe with 300 dimensionalities outperform by the accuracy of 83.72%.

In this experimental results, we found all three regression model i.e., LR, RF, and SVR are giving similar results but are very less in comparison to our approach CoWe and MaLSTM. That is because these regression model will work great on inferencing word meaning as researched by Minglei [4] but not on collective knowledge and long term dependency. That's why MaLSTM and our model performed well as they preserve the contextual meaning of the whole sentence, not a particular word.

MaLSTM is using two LSTM that takes the premise and hypothesis to each input of LSTM. First hidden layer consist of 50 nodes and second hidden layer consist of 100 noes, with taking 25% dropout for preventing regularization and overfitting. Both LSTM units consist of rectifier linear unit with batch normalization then output layer consists of sigmoid function which gives an accuracy of 77.67%.

In our model, it's taking four input instead of two inputs in MaLSTM i.e., sentence 1 of the training set, sentence 2 of training set, sentence 1 of training set and sentence 2 of training set which passed through embedding layer to generate embedding vectors through predefined GloVe [19] which generates Embedding Matrix. This embedding is passed through respective input units, we initialize weights of node i.e, mean 0.0, standard deviation 0.05 and seed 2. In each layer, we are using a number of filters 32, filter length 3 and dropout of 25%. We are wrapping CNN layer with TimeDistributed layer their output will flatten and merged to be input into Dense Layer which consists 128 x 2 nodes with activation function relu and this will be input into another Dense Layer of 128. Other two inputs consist of TimeDistribution layer which consists of Dense Neural Net their output will be merged through concatenation and their output will be taken as input into Dense Layer which consists 128 x 2 nodes with activation function relu and this will be input into another Dense Layer of 128 nodes. The output coming from CNN and TimeDistribution is then merged through concatenation the

their output will be taken as input for the into Dense Layer which consists 128 x 2 nodes with activation function relu and this will be input into another Dense Layer of 128 nodes, but as it needs more processing we again pushed the output in the Dense layer of 128 nodes and their output is then passed to 1 node Dense layer with Sigmoid activation function. To this wrapping CNN layer with TimeDistribute dense layer and increasing the hidden layers for better preprocessing and remembering dependency achieved accuracy of 83.72% which is better then MaLSTM.

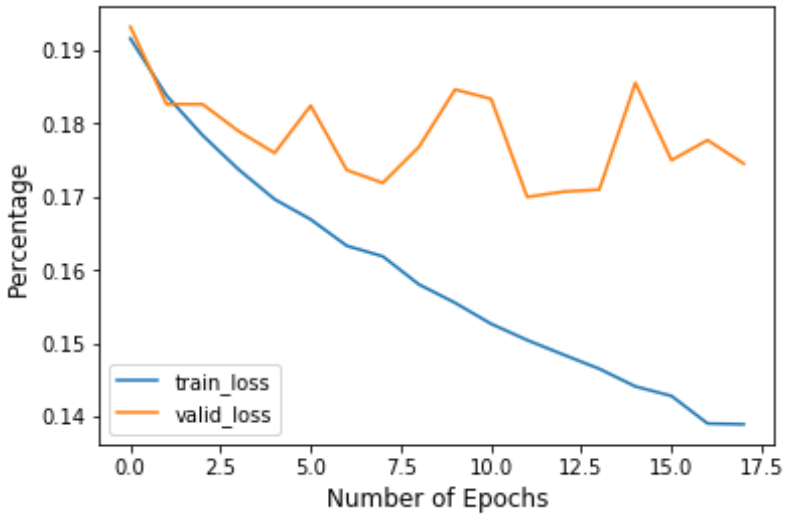

Figure 5.7: 50 Dimensional train loss vs valid loss CoWe

In figure 5.6, it shows the training loss and the valid loss with increasing the number of epochs while using 50 Dimensional GloVe vector, the x-axis represents a number of epochs and y-axis represents the percentage of train loss and the valid loss. According to this, we achieved maximum accuracy at 12^{th} epochs i.e, 74.13%.

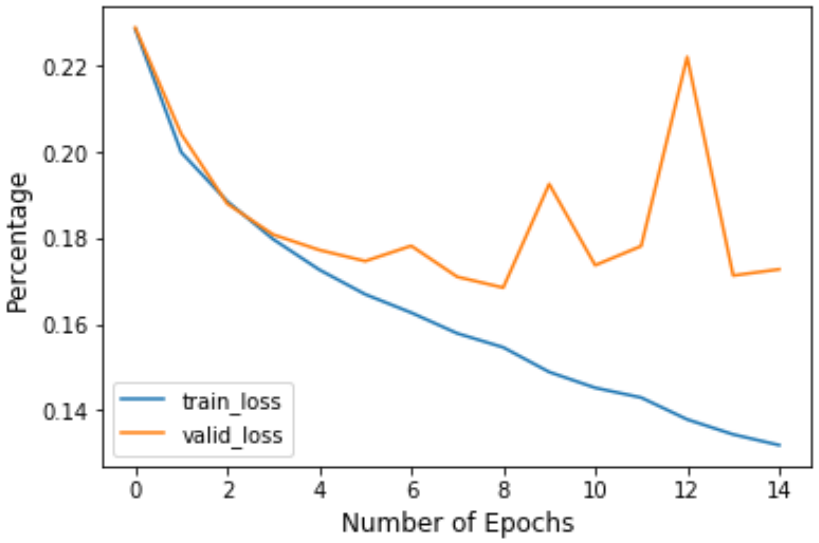

Figure 5.8: 100 Dimensions train loss vs valid loss CoWe

In figure 5.7, it shows the training loss and the valid loss with increasing the number of epochs while using 100 Dimensional GloVe vector, the x-axis represents a number of epochs and y-axis represents the percentage of train loss and the valid loss. According to this, we achieved maximum accuracy at 12th epochs i.e, 73.16%.

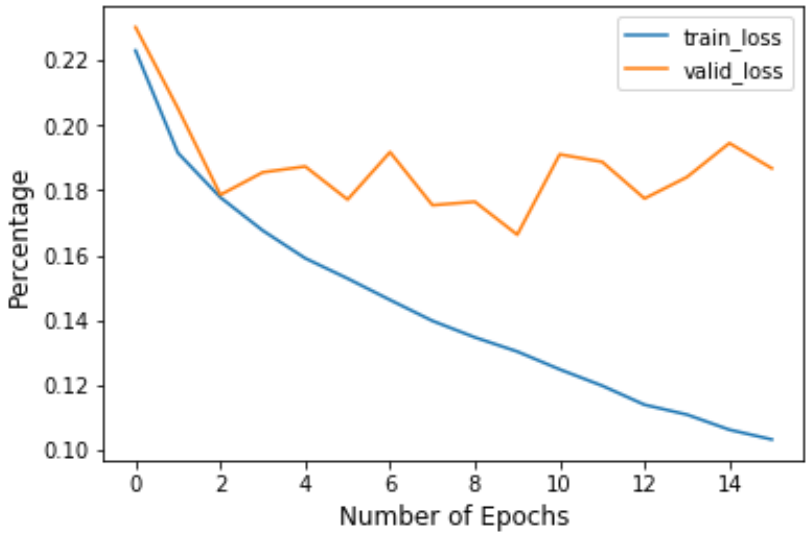

Figure 5.9: 200 Dimensions train loss vs valid loss CoWe

In figure 5.8, it shows the training loss and the valid loss with increasing the number of epochs while using 200 Dimensional GloVe vector, the x-axis represents a number

of epochs and y-axis represents the percentage of train loss and the valid loss. According to this, we achieved maximum accuracy at 8th epochs i.e, 75.12%.

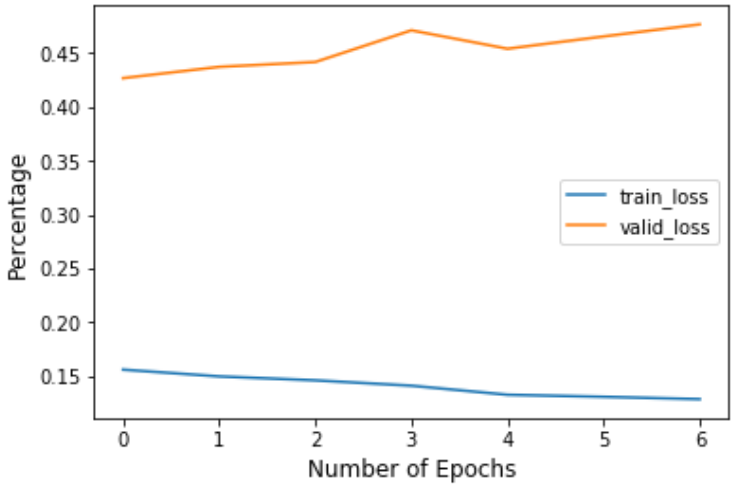

Figure 5.10: 300 Dimensions train loss vs valid loss CoWe

In figure 5.9, it shows the training loss and the valid loss with increasing the number of epochs while using 300 Dimensional GloVe vector, the x-axis represents a number of epochs and y-axis represents the percentage of train loss and the valid loss. According to this, we achieved maximum accuracy at 2nd epochs i.e, 83.72%.

Table 5.2: Evaluation of Proposed Approach CoWe

S.No.	Embedding dimension	Token in GloVe (billion)	Number of epoch	Training Accuracy	Accuracy	Training Loss	Valid Loss
1	50	6	12	78.10	74.13	15.03	16.99
2	100	6	12	79.54	73.16	14.29	17.81
3	200	6	8	81.79	75.12	13.03	16.63
4	300	840	2	80.79	83.72	14.98	43.72

We achieved the embedding dimension maximum accuracy of 83.72% with 300 dimensional and 840 billion tokens in 2 epoch. Which concludes as the number of dimensionality increase and also the token counts directly proportional to the accuracy achieved.

CHAPTER 6 CONCLUSION AND FUTURE WORK

In this research, it proves the power of word embedding with our proposed CoWe. Which outperform from the rest of the regression models. We consider our approach as a general purpose to predict the similarity between the sentences through CoWe which achieved accuracy of 83.72%. In future work, we will be more focus on data preprocessing and try with other neural networks.

References

[1] Picard, R. W., "Affective Computing," *Tech. Rep. 321, MIT Media Lab, 20 Ames St., Cambridge, MA 02139,* 1995.

[2] Hoste, B. Desmet and V., "Emotion detection in suicide notes," *Expert Systems with Applications,* vol. 40, p. 6351–6358, 2013.

[3] Lee, B. Pang and L., "Opinion mining and sentiment analysis,," *Foundations and trends in information retrieval,* vol. 2, pp. 1 - 135, 2008.

[4] Minglei Li, Qin Lu, Yunfei Long, and Lin Gui, "Inferring Affective Meanings of Words from Word Embedding," *IEEE Transactions on Affective Computing,* 2017.

[5] T. Y. Aizhang Guo, "Research and improvement of feature words weight based on TFIDF algorithm," *IEEE Information Technology, Networking, Electronic and Automation Control Conference,* pp. Pages: 415 - 419, 2016.

[6] SALTON G, FOX E A, WUH., "Extended Boolean information retrieval," *Communications of the ACM,* vol. 26 (11), pp. 1022 - 106, 1983.

[7] S., JONES K, "A statistical interpretation of term specificity and itsap-plication in retrieval," *Journal of Documentation,* pp. 11-21, 1972.

[8] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Proceedings of 27th Annual Conference on Neural Information Processing Systems (NIPS), (Nevada, United States),* p. 3111–3119, 2013.

[9] Qi Wang, Jungang Xu, Hong Chen, Ben He, "Two Improved Continuous Bag-of-Word Models," *International Joint Conference on Neural Networks (IJCNN),* pp. pages 2851 - 2856, 2017.

[10] Salakhutdinov, G. E. Hinton and R. R., "Reducing the dimensionality of data with neural networks," *Science,* vol. 5786, pp. 504 - 507, 2015.

[11] Y. Bengio, H. Schwenk, J. S. Senecal, F. MOrin, Gauvain and J. L., "A neural probabilistic language model," *Journal of Machine Learning Research,* vol. 3, pp. 1137 - 115, 2003.

[12] Chia-Yang Chang, Shie Jue Lee , Chih Chin Lai, "Weighted Word2vec Based on the Distance of Words," *International Conference on Machine Learning and Cybernetics (ICMLC),* 2017.

[13] T. Brants, A. C. Popat, P. Xu, F. J. Och and J. Dean, "Large language models in machine translation," *In Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Language Learning,* pp. 858 - 867, 2007.

[14] J.Weston, R. Collobert and, "A unified architecture for natural language processing: deep neural networks with multitask learning," *In Proceedings of International Conference on Machine Learning,* pp. 160-167, 2008.

[15] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa, "Natural language processing (Almost) from scratch," *Journal of Machine Learning Research,* vol. 12, pp. 2493 - 2537, 201.

[16] S. K. J. L. J. C. N. D. a. N. E. Deokgun Park, "ConceptVector: Text Visual Analytics via Interactive Lexicon Building using Word Embedding," *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS,* Vols. VOL. 24, NO. 1, 2018.

[17] C. D. A. B. a. I. T. W. Ling, "Two/too simple adaptations of Word2vec for syntax problems," *In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* p. 1299–1304, 2015.

[18] W. Ling, Y. Tsvetkov, S. Amir, R. Fermandez, C. Dyer, A. W. Black, I. Trancoso, and C.-C. Lin, "Not all contexts are created equal: Better word representations with variable attention," *In Proceedings of the Conference on Empirical Methods in Natural Language Processing,* pp. 1367 - 1372, 2015.

[19] G. A. P. J. Yash Sharma, "Vector Representation of Words for Sentiment Analysis Using GloVe," *International Conference on Intelligent Communication and Computational Techniques (ICCT),* pp. 279 - 284, 2017.

[20] Z. X. CongyingShi, "Review TFIDF algorithm," *Computer Applications,* pp. 167 - 170, 2009.

[21] Jeffrey Pennington, Richard Socher, Christopher D. Manning, "GloVe: Global Vectors forWord Representation," *conference on empirical methods in natural language processing (EMNLP),* p. 1532–1543, 2014.

[22] R. Torkzadeh, A. Mirzaei, M. M. Mirjalili, A. S. Anaraki, M. R. Sehhati and F. Behdad, "Medium term load forecasting in distribution systems based on multi linear regression & principal component analysis: A novel approach," *19th Conference on Electrical Power Distribution Networks (EPDC),* pp. 66 - 70, 2014.

[23] Fu, Kun; Wang, You-Hua; Dong, Yong-Feng; Hou, Xiang-Dan; Shen, Xue-Qin; Yan, Wei-Li, "Support vector regression method for boundary value problems," *International Conference on Machine Learning and Cybernetics,* vol. 7, pp. 4295 - 4298, 2005.

[24] Torizuka, K.; Oi, H.; Saitoh, F.; Ishizu, S., "Research of Text Categorization Model based on Random Forests," *IEEE International Conference on Industrial Engineering and Engineering Management (IEEM),* pp. 487 - 491, 2018.

[25] ShuohangWang, Jing Jiang, "Learning Natural Language Inference with LSTM," *arXiv:1512.08849v2,* 2016.

[26] Kiranyaz, Serkan; Ince, Turker; Abdeljaber, Osama; Avci, Onur; Gabbouj, Moncef, "1-D Convolutional Neural Networks for Signal Processing Applications," *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* pp. 8360 - 8364, 2019.

[27] Vignesh Thakkar, Suman Tewary, "Batch Normalization in Convolutional Neural Networks – A comparative study with CIFAR-10 data," *Fifth International Conference on Emerging Applications of Information Technology (EAIT),* pp. 1 - 5, 2018.

[28] Fujii, Shohei; Hayashi, Hitoshi, "Comparing of performane by activation functions on deep Image Prior," *International Conference on Artificial Intelligence in Information and Communication (ICAIIC),* pp. 255 - 258, 2019.

[29] V. R. K. R. Bhaskar Dhariyal, "Sentiment analysis via Doc2Vec and Convolutional Neural Network hybrids," *IEEE Symposium Series on Computational Intelligence (SSCI),* pp. 666 - 671, 2018.

[30] Mukherjee, Sourabrata, "t-SNE based feature extraction technique for multi-layer perceptron neural network classifier," *International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT),* pp. 660 - 664, 2017.

[31] Severoğlu, Nagihan, “Mammogram images classification using Gray Level Co-occurence Matrices,” *24th Signal Processing and Communication Application Conference (SIU),* 2016.

[32] Yu, Hongchuan; Bennamoun, M., “1D-PCA, 2D-PCA to nD-PCA,” *18th International Conference on Pattern Recognition (ICPR'06),* vol. 4, pp. 181 - 184, 2006.