A Comparative Study of Deep Learning Algorithms for Text Classification

DISSERTATION

Submitted in partial fulfillment of the requirements of the

MTech Data Science and Engineering Degree Programme

By

Preetam Kumar
2018AB04517

Under the supervision of

Prashant Sugur
Principal Project Manager

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
Pilani (Rajasthan) INDIA

August 2021

DSE CL ZG628T DISSERTATION

A Comparative Study of Deep Learning Algorithms for Text Classification

Submitted in partial fulfillment of the requirements of the

M. Tech. Data Science and Engineering Degree Programme

By

Preetam Kumar
2018AB04517

Under the supervision of

Prashant Sugur
Principal Project Manager

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN)

August 2021

ACKNOWLEGDEMENTS

"*Imagination is more important than knowledge. As knowledge is limited to all we know and understand, while imagination encircles the entire Universe. – Albert Einstein*"

I would like to take this opportunity to express my gratitude to my Mentor cum Supervisor **Prashant Sugur**, who gave me the opportunity to do this project, lending me his experience and counsel. He was a tremendous help in the guiding, mentoring, motivating, and shaping my imagination to achieve this goal.

Many thanks to **Prof. Sarvamangala, BITS Pilani** for giving timely feedback on the project and encourage for continuous improvement through her guidance during the review sessions and igniting the mind to imagine and go beyond the obvious.

Last but not the least I would also like to thank my family and friends who supported me in my endeavours in ways both big and small, but always significant during this project.

Preetam Kumar

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the Dissertation entitled ___**A Comparative Study of Deep Learning Algorithms for Text Classification**

and submitted by Mr./~~Ms~~.__**Preetam Kumar**_____ IDNo.___**2018AB04517**_____

in partial fulfillment of the requirements of DSE CL ZG628T Dissertation, embodies the work done by him/~~her~~ under my supervision.

Signature of the Supervisor

Name : Prashant Sugur

Place: __Bangalore_____

Designation: Principal Project Manager

Date:__10th August'2021_____

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
Work Integrated Learning Programmes Division
II SEMESTER 2020-21

DSE CL ZG628T **DISSERTATION**

(Final Evaluation Sheet)

NAME OF THE STUDENT: Preetam Kumar

ID NO. : 2018AB04517

Email Address : preetamkumar@gmail.com

NAME OF THE SUPERVISOR: Prashant Sugur

PROJECT TITLE : A Comparative Study of Deep Learning Algorithms for Text Classification

(Please put a tick (□) mark in the appropriate box)

S.No.	Criteria	Excellent	Good	Fair	Poor
1	Work Progress and Achievements	√			
2	Technical/Professional Competence		√		
3	Documentation and expression	√			
4	Initiative and originality	√			
5	Punctuality	√			
6	Reliability		√		
	Recommended Final Grade		√		

EVALUATION DETAILS

EC No.	Component	Weightage	Marks Awarded
1	Dissertation Outline	10%	
2	Mid-Sem Progress Seminar Viva Work Progress	 10% 5% 15%	
3	Final Seminar/Viva	20%	
4	Final Report	40%	
	Total out of	100%	

	Supervisor	Additional Examiner
Name	**Prashant Sugur**	
Qualification	B.Tech	
Designation & Address	Principal Project Manager SAP India, RMZ Ecoworld, Bangalore	
Email Address	**prashant.sugur@sap.com**	
Signature		
Date	10th August 2021	

NB : Kindly ensure that recommended final grade is duly indicated in the above evaluation sheet.
POSTAL ADDRESS FOR ALL FUTURE CORRESPONDENCE. FILL IT UP NEATLY IN CAPITAL LETTER WITH PIN CODE ETC.

Address:
1073_PRESTIGE IVY TERRACES, KAVERAPPA LAYOUT, PANATHUR, BANGALORE, KARNATAKA
Pin Code 560103

Abstract

Text classification is one of the fundamental problems of Natural Language Processing which is playing a vital role in enterprise applications e.g. conversational AI based chat bots, document classification, sentiment/opinion analysis, incident/ticket classification, information classification, experience analysis etc. since past few years. In general, the classical algorithms like decision tree, rule base classification, logistic regression (for binary classification e.g. spam filtering) are used for this purpose. The major challenge that one faced in text classification is word data are sparse, high dimensional and low frequency. Traditional text classification like bag-of-words, n-gram heavily dependent on feature engineering.

In contrast the text extract from an enterprise system (e.g. SAP or Oracle) the key words (e.g. Transaction Code, Purchase Order, Billing, Sales Order, Vendor, Customer, Quotation, Authorization etc.) are dense and relatively of high frequency. With the development of deep learning technique over the last decade and reduced cost of computing power, many deep learning models have been proposed for the text classification. It is found that the results of various deep learning models have been encouraging over the traditional models. Any improvement over the accuracy of text classification model will tremendously reduce the human effort and time in revalidating the classification. Apart from the traditional token/keywords-based classification the sentiments behind the text plays a key role in text classification. An amalgamation of token/keywords and sentiment analysis-based algorithm will be effective in understanding human written text and classification of the text by assigning appropriate label which could not be achieved through the conventional text classification algorithms. This project is an attempt to do a comparative study of the deep learning methods like CNN and Text Level Graph Neural Network applied for sentiment classification.

DISCLAIMER: *This project is an attempt to conduct a comparative study of deep learning models for text classification from the academic perspective that may help the organization while evaluating different model for text classification for any enterprise level uses. It should be noted that no organizational resources like code snippet, data or any other resources has been used in performing and executing any part of this project. The data source used is available in public domain.*

List of Symbols & Abbreviations Used

r_n – *Lower case bold later represents* vector

R^{mxn} – *Upper case bold later represents Matrix of dimension mxn.*

$Z_v{}^T$- *Denotes the transpose of the Matrix Zv*

GNN – *Graph Neural Network*

GCN - Graph Convolution Network

CNN/ConvNets – *Convolution Neural Network*

RNN – *Recurrent Neural Network*

LSTM – *Long Short Term Memory Network*

NLP – *Natural Language Processing*

TF-IDF – *Term Frequency-Inverse Document Frequency*

List of Tables

Table – 1 – *CNN Model Result Analysis*

Table – 2 – *Text Level GNN Model Result Analysis*

Table – 3 – *Project Plan and Deliverables*

List of Figures

(The list of figures does not include the code snippet and the result of execution of code.)

Figure 1- A *typical Graph 'G' and it's representation*

Figure 2- *Deep Neural Network*

Figure 3- *A typical model of text classification using CNN.*

Figure 4- *Example of Text GCN*

Figure 5- *Example of Text Level Graph Neural Network*

Figure 6- *Node Embedding of a Graph*

Figure 7- *Message Passing in Graph Network*

Figure 8- *High Level Design*

Table of Contents

Abstract 5

List of Symbols & Abbreviations Used 6

List of Tables 6

List of Figures 6

Chapter-1 10

Introduction and Background: 10

Chapter 2 11

Problem Statement: 11

Objective of the Project 11

Uniqueness of the Project 12

Benefit to the Organization 12

Scope of Work 12

Chapter 3 13

Literature Survey on Deep Learning for Text Classification: 13

Word Embedding: 13

Pre-trained Word Embedding (Glove): 14

Chapter 4 15

Graph, Deep Learning & Convolution Neural Network: 15

Graph Convolution Neural Network: 16

Graph Neural Network for Text Classification: 18

Chapter 5 22

Experiment Set-up: 22

Model Set-up, Tunning and Analysis: 27

Chapter 6 33

Observation: 33

Conclusion: 33

Direction of Future Work: 33

Appendix-A 34

Project Plan & Deliverables 34

Illustrative examples of citation of Bibliography / References: 35

Check list of items for the Final report 36

Chapter-1

Introduction and Background:

The classification problem is one of the primitive activities that human has been performing since dawn of human race. The problem of classification has been studied widely in the modern era by computer scientist in the context of database, data mining and information retrieval. The problem of classification can be defined as a records say $D = \{X_1, X_2, \ldots\ldots\ldots, X_n\}$ to be assigned to a class from a set of k class namely $C = \{C_1, C_2, \ldots\ldots\ldots, C_k\}$. In the context of machine learning the data set assigned to the respective classes can be used to train and build a classification model that can learn the association based the underlying feature of the data.

Text classification is a fundamental problem of Natural Language Processing with lot of useful application build on the concept of text classification namely sentiment analysis, fake news classification, SPAM detection, document organization & retrieval etc. Some of the key classical methods that are used for text classification are Decision Tree, Rule (Pattern) based classification. Probabilistic generative model like Bayesian classifier is also used to classify text. With the availability cost effective computing power, a lot of research and development in text classification using Deep Learning are being carried out recently. Neural Network like CNN uses the discriminative classifier to classify the text. One of the recent developments in deep learning method for the text classification by using the representation power of graph neural network or popularly known GNN. One of such GNN algorithm is Text Level Graph Neural Network. This algorithm uses the concept word embedding to represent the feature of a word in terms of Word2Vector in d dimension space and each word that is represented as a node in the text graph pass on the message to its neighbouring word which ultimately decided the class of the text. This project is based on this concept or the algorithm and comparing this with the CNN model of deep learning method that is used for text classification.

Some of the key classical methods that are used for text classifications are

Decision Trees: Decision tree is a hierarchical structure based upon the underlying data space and divides/classify by using difference text features. This basically creates a class partitions which are more skewed in terms on class distribution.

Patter or Rule based classifier: In this type of supervise classification we determine the word pattern that are most likely to be related to the different classes. Based on the underlying data a set of rules are formed that corresponds to a particular pattern and respective labels. These rules are used for the purpose of classification.

Bayesian Classifier: Bayesian classifier is a generative classifier. It determines the posterior probability of a text belonging to a particular class based on the presence of difference words in the text and selects the class that generates the maximum probability.

Chapter 2

Problem Statement:

Text classification is one of the fundamental problems of Natural Language Processing which is playing a vital role in enterprise applications e.g. conversational AI based chat bots, document classification, sentiment/opinion analysis, incident/ticket classification, information classification, sentiment analysis etc. In general, the classical algorithms like decision tree, rule base classification, logistic regression (for binary classification e.g. spam filtering) are used for this purpose. The major challenge that one faced in text classification is word data are sparse, high dimensional and low frequency. Traditional text classification like bag-of-words, n-gram heavily dependent on feature engineering.

In contrast the text extract from an enterprise system (e.g. SAP or Oracle) the key words (e.g. Transaction Code, Purchase Order, Billing, Sales Order, Vendor, Customer, Quotation, Authorization etc.) are dense and relatively of high frequency. With the development of deep learning technique over the last decade and reduced cost of computing power, many deep learning models have been applied for the text classification. It is found that the results of various deep learning models have been encouraging over the traditional models. Any improvement over the accuracy of text classification model will tremendously reduce the human effort and time in revalidating the classification. Apart from the traditional token/keywords-based classification the sentiments behind the text plays a key role in text classification. An amalgamation of token/keywords and sentiment analysis-based algorithm will be effective in understanding human written text and classification of the text by assigning appropriate label which could not be achieved through the conventional text classification algorithms.

Objective of the Project

The main objective is to analyse and conduct a comparative study of the deep learning models on the context of text classification using CNN and a text level graph neural network (GNN).

Uniqueness of the Project

To do a comparative study of Deep Neural algorithms for text classification to understand best suitable model for text classification for an intelligent enterprise software in terms of performance and accuracy on context of twitter tweets. We will be selecting 2 Deep Learning models namely 1. A Deep Learning Models with CNN for Text Classification and 2. Graph Neural Network based Text Classification.

Benefit to the Organization

The comparative analysis of deep learning models can throw light on the performance as well as the accuracy of the model in the context of text classification which will help in selecting and further optimizing in future to be used at an enterprise scale.

Scope of Work

To perform a comparative analysis of Deep Learning model for Text Classification. For this project the scope includes the following

1. To analyse some of the resent research papers related to use of Deep Learning in Text Classification. Select a research paper to build the experimental model for the assessment of that in text classification problem.
2. Identifying Data Set available in public domain to be used in the model evaluation
3. Build an experimental model (To prepare a python notebook) to build, train and evaluate a deep neural network (CNN) on the dataset
4. Build the experimental model (To prepare a python notebook) to build, train and evaluate the model based on the research paper selected on the data set
5. Conduct a comparative study on the performance by tuning the hyper parameters

Chapter 3

Literature Survey on Deep Learning for Text Classification:

The research on deep learning for text classification can be categorized in 2 categories. One group of researchers focusing on the model based on word embedding whereas the 2nd group of researcher's research focused on deep neural networks. The recent studies mainly by researchers like Shen, Joulin and Wang showed that the success of the text classification deep learning model largely depends on the effectiveness of the word embedding. Another group of researchers (Kim in 2014, Zhang & Zhao in 2015) extensively used the deep learning models like CNN and RNN for text/sentence classification. The architecture is the direct application of CNN model that are used computer vision however with only one difference that in case of text classification the input is one-dimensional convolution. Tai, Socher & Manning in 2015 and Liu, Qiu & Huang in 2016 used LSTM a special type of RNN to learn text representation that further fuels the research to come up with the attention mechanism for text classification.

Researcher like Zhang, Zhao and LeCun used the concept of character level convolution networks for text classification. They treated text as a kind of raw one dimensional signal at the character level and used the CNN model of ConvNet for the classification model. They used a model consist of 2 ConvNets- one large and one small. Both the ConvNets are 9 layers deep with 6 convolution layers & 3 fully connected layers. The input was equal to 70 features due to the character quantization method i.e. there are total 70 non-space characters i.e. abcd….z0123….9 + other special character and newline character.

Word Embedding:

A computational model or system can only process numerical values, it cannot process words or any lexical symbol directly. Hence to processing of words and text in computational model we must convert the word or lexicon to a numerical value. This method of converting word to a corresponding numerical value is called word embedding. In other words "Word Embeddings" are the texts or words converted into numbers. It may be the same text represented by different numerical value. The different type of word embeddings can be broadly classified as 1. Frequency based embedding 2. Prediction based embedding. Count vector, TF-IDF vector and co-occurrence vector are some of the examples of frequency-based embedding.

Count Vector: If we consider a corpus C of D documents denoted as $\{d_1, d_2, d_3, \ldots, d_D\}$ and it contains N unique words or tokens then the dictionary will consists of N words and the count vector matrix will be a D X N matrix. Each row of the matrix will contain the frequency of the token in that particular document d_D.

TF-IDF: In the TF-IDF method the term frequency TF is calculated as TF = (Number of times term t appears in a document)/(Number of terms in the document) and IDF as IDF = log(N/n), where, N is the number of documents and n is the number of documents a term t has appeared in (where N is the number of documents and n is the number of documents a term t has appeared in.). The TF-IDF is nothing but the product of these 2 factors.

Co-Occurrence Matrix: The idea behind the co-occurrence is that similar words appears/ occurs together and forms the context in a text e.g. Sachin Tendulkar is a cricketer. Virat Kohli is an Indian cricketer. Hence both Sachin Tendulkar and Virat Kohli tend to have similar context i.e. cricketer. In co-occurrence matrix the concept of context window defines the number of steps and direction. The co-occurrence of 2 words is the number of times they have appeared together in a context window. A co-occurrence matrix of V X N size where N is a subset of V and can be obtained by removing irrelevant words like stopwords etc. where each element of the matrix defines the occurrence of the word identified by the corresponding row and column.

Pre-trained Word Embedding (Glove):

In recent year a group of Stanford NLP researchers namely Jeffrey Pennington, Richard Socher, and Chris Manning released a tool called GloVe (Global Vectors for Word Representation). Since its release this tool has been used extensively for learning continuous-space vector representations of words. In an unsupervised method for learning word representation the statistics of word occurrence in a corpus is the primary source of information that is available to us. The GloVe model tries to generate the meaning of a word from the insight derived from the corpus.

Chapter 4

Graph, Deep Learning & Convolution Neural Network:

Graph is an abstract mathematical structure that has been extensively used in computer science and AI to resolve a number of practical computational problems. In mathematics graph $G = (V, E)$ is define as mathematical structure consists of a nonempty set V of vertices (or nodes) and a set E of edges. Each edge has either one or two vertices associated with it, called its endpoints. An edge is said to connect its endpoints. In computer science graph is a type of data structure that models a set of objects known as vertices and represents the relationship among them as edges of the graph.

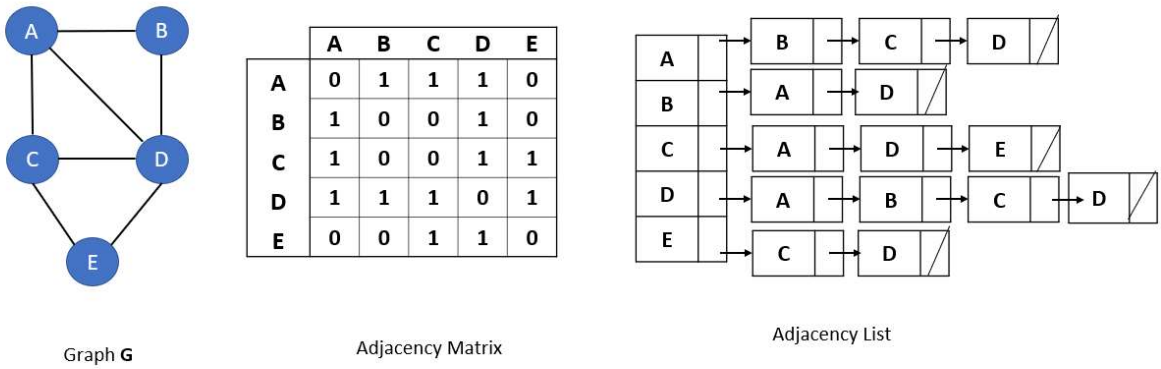


Figure 1- A typical Graph 'G' and it's representation

A neural network is an interconnected group of artificial neurons that mimics the human brain via a series of algorithms to recognize the underlying relationship in a set of data. It is computing systems inspired by the biological neural networks that constitute animal brains. Though the concept of artificial neural network first suggested in 1940s however it became famous in last decade when Deep Learning produced great result based on Artificial Neural Network with representation learning. It has been significantly applied in field of computer vision and imaging, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance. The adjective deep in Deep Learning refers to the multiple layers of network.

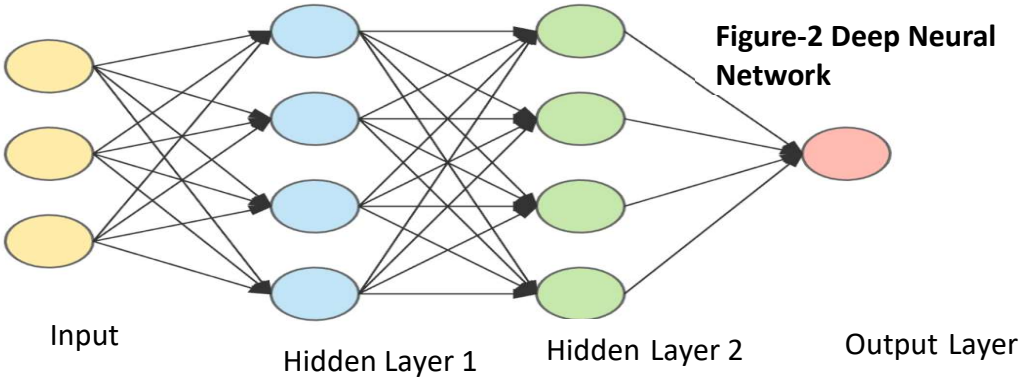


Figure-2 Deep Neural Network

In mathematics convolution is a mathematical operation between 2 functions that generates a third function which express the how the one function is modified by the other function. In deep learning literature Convolution Neural Network (CNN) is deep neural network that converts the more complex structure into a form that is easier to process without losing the feature of original structure. CNN is nothing but a type of neural network that uses the convolution mechanism in place of simple matrix multiplication in at least of the layer. Though CNN concept was first introduced in 1980s but the start of use of CNN for image recognition and computer vision since last one decade. Though the use of CNN started with computer vision and image recognition however recently the use of CNN to NLP has produced promising results.

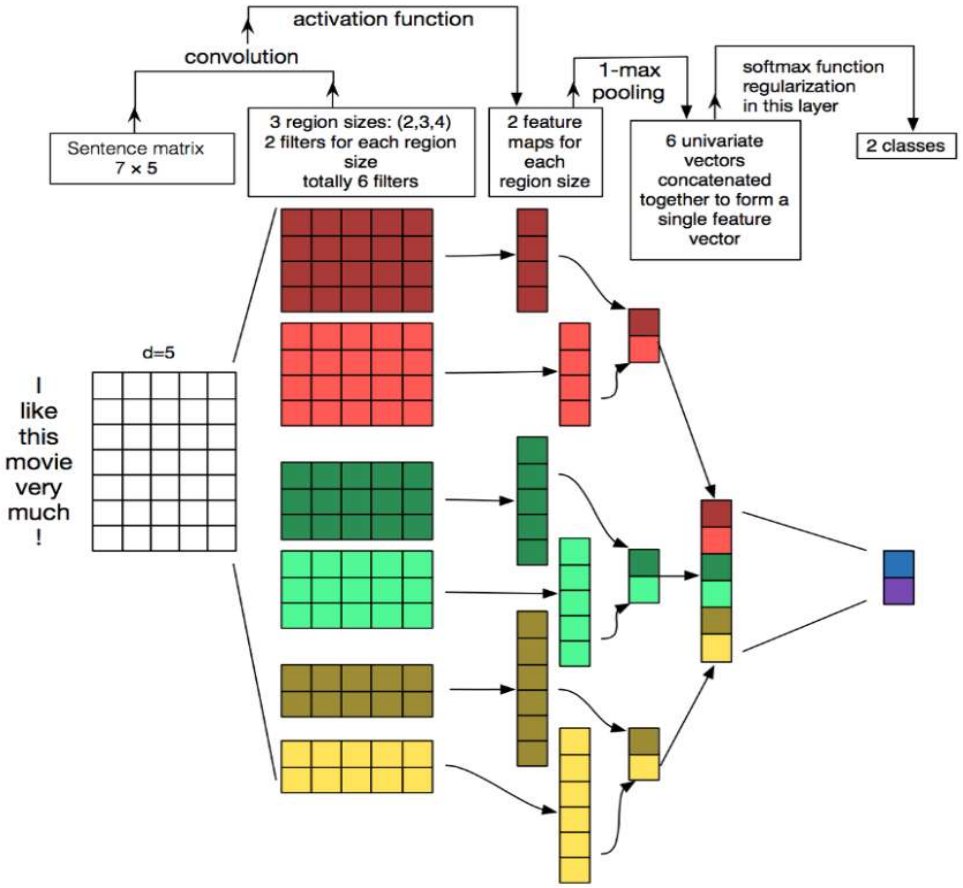


Figure- 3 A typical model of text classification using CNN.

Image Reference : http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

Graph Convolution Neural Network:

The concept of Graph Convolution Network (GCN) model was proposed by Kipf[7] & Welling in 2017. A GCN is basically a multiplayer neural network that directly operates on the graph by inducing the embedding vector of nodes based on the properties of their neighbourhood nodes. Let's consider a graph G = (V,E) where V is the set of nodes(|V| = n) & E is the set of edges in the graph G. The assumption is that all the nodes in the graph is connected to

themselves i.e. if |V| = n then there exist at least n self-loops in the graph and (v, v) Є E. Let m be dimension of feature vector or node embedding then $\mathbf{R^{nxm}}$ is the matrix that contain the feature vector of all the nodes i.e. the row X_v Є $\mathbf{R^m}$ is the feature vector vertex v. The adjacency matrix **A** of graph G will have all the diagonal elements as 1 due to presence of self-loop in the graph for all nodes. Another matrix called as degree matrix denoted by **D**, where $D_{ii} = \Sigma_j A_{ij}$. GCN can capture the information about the immediate neighbour with one layer of convolution. Thus multiple stacked GCN layer can extract information and integrate from larger neighbourhood.

For a one-layer GCN, the new k-dimensional node feature matrix is computed as

$$\mathbf{L^{(1)} = \rho\,(A'XW_0)}$$

Where $\mathbf{A' = D^{-1/2}AD^{-1/2}}$ is the normalized symmetric adjacency matrix and $\mathbf{W_0}$ Є $\mathbf{R^{mxk}}$ is a weight matrix and **ρ** is an activation function e.g. ReLU.

The text GCN is a very large and heterogenous text graph which contains word nodes as well as the document/text nodes that helps in explicit modelling of global co-occurrence and will be easy for adaptation of graph convolution.

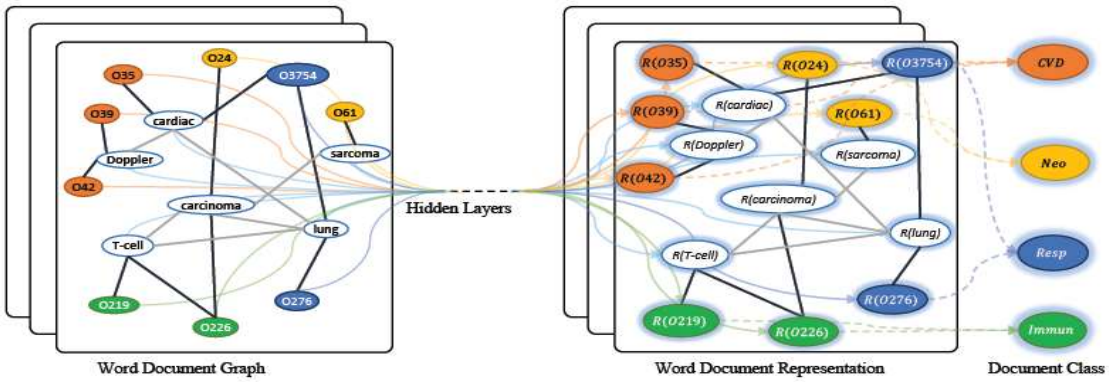


Figure 4 –Example of Text GCN

Schematic of Text GCN. Example taken from Ohsumed corpus. Nodes begin with "O" are document nodes, other sare word nodes. Black bold edges are document-word edges and gray thin edges are word-word edges. R(x) means the representation (embedding) of x. Different colors mean different document classes (only four example classes are shown to avoid clutter). CVD: Cardiovascular Diseases, Neo: Neoplasms, Resp: Respiratory Tract Diseases, Immun: Immunologic Diseases.

*#- The above diagram is taken from the paper **"Graph Convolutional Networks for Text Classification** by Liang Yao, Chengsheng Mao, Yuan Luo"*

For a text GCN like the figure shown above the adjacency matrix can be defined as

$$A_{ij} = \begin{cases} \text{PMI (i,j) i,j are words, PMI(i,j) > 0} \\ \text{TF-IDF}_{ij} \text{ , i document and j is a word} \\ 1 \text{ , i = j} \\ 0 \text{ , otherwise} \end{cases}$$

Where PMI is the point wise mutual information defined as

$$\text{PMI (i,j)} = \log [p(i,j)/p(i)p(j)]$$
$$p(i,j) = \#W(i,j)/\#W$$
$$\text{and } p(i) = \#W(i)/\#W$$

Where #W(i) is the number of sliding windows containing the word i in the corpus, #W(i,j) is the number of sliding windows that contain both word i and j and #W total number of words in the corpus. Thus a positive PMI indicates a high semantic correlation of words and a negative PMI indicates little or no semantic correlations in the corpus. It is customary to take the feature matrix **X** as **I** i.e. identity matrix as the identity matrix indicates a one-hot vector.

After building the input matrix it is fed to a 2-layer GCN where the 2n layer node embedding has the same size as the number of labels and finally fed to a softmax classifier. During the training the objective function cross-entropy-loss is minimized to arrive at the optimum parameter values.

Graph Neural Network for Text Classification:

For text classification using Graph Neural Network (GNN) most of the model builds a single graph for the entire corpus and the edge weight of such graph are assigned a fixed value. Such models use the concept of node similarity to learn & build the classification model for the text classification. This type of GNN model faces mainly 2 challenges, one related to the high memory consumption as the entire graph build over the whole corpus has to be loaded to the memory while training and second the expression ability of the edges reduces considerably as the edges are assigned a fixed weight. To address such problem **Lianzhe Huang, Dehong Ma** proposed a new GNN based model for text classification in their paper ***"Text Level Graph Neural Network for Text Classification[1]"***. They proposed an idea that instead of building a single graph for the entire corpus, individual graph for each text to be build. In other words for text level graph all the words appearing in a text will be considered as nodes/vertices of a graph and they will be connected within a reasonably small windows (e.g. each word connected to 2 other word just preceding and succeeding it in) a text instead of building a fully connected graph of the entire corpus.

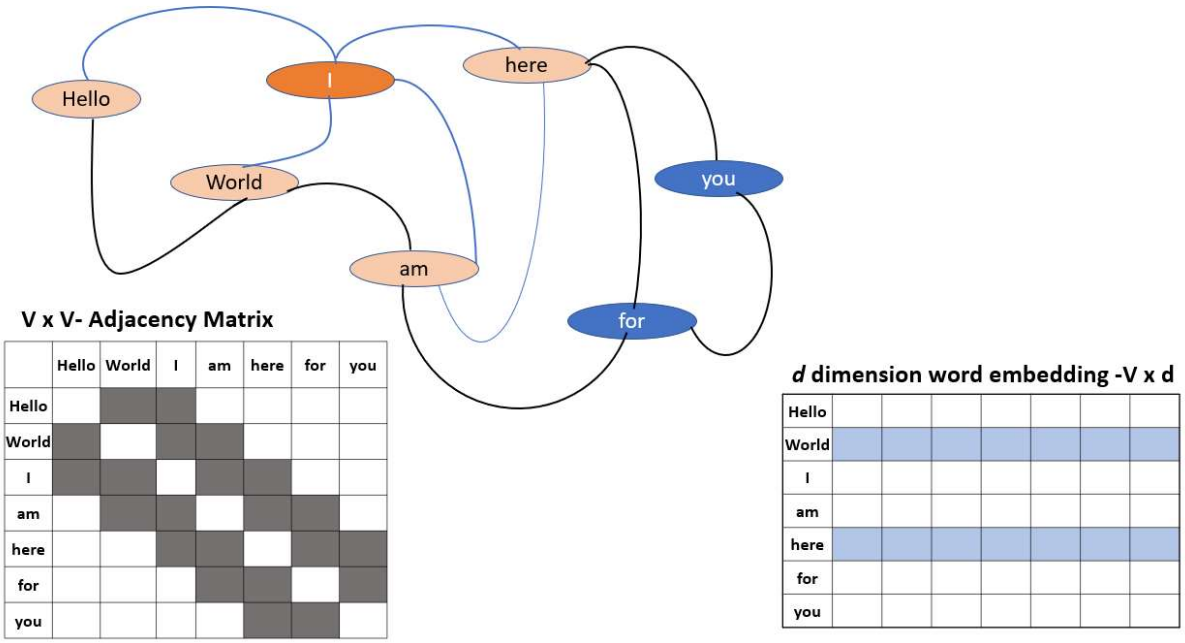


Figure 5 –Example of Text Level Graph Neural Network

Above figure 3 shows an example of a typical text level graph. The text ***"Hello World! I am here for you."*** has been depicted using a text level graph where each word in the text is represented by a node in the graph and the edges in the graph depends on a parameter value p that represents the number words that a word can influence its succeeding words and can be influence by the preceding the words. In other words, the adjacency matrix will have entries non-zero value for such edges and zero for others. Each node can be mapped to a d-dimension node embedding.

Node Embedding: The intuition behind node embedding is to map the nodes of a graph to a d-dimension so that similar type of nodes in the graph are embedded closer. One of such similarity measures is cosine similarity. If one finds the cosine similarity between two similar node it will be close to 1. In figure 6 $\mathbf{Z_u}$ represents some d-dimensional node embedding of node u whereas $\mathbf{Z_v}$ represents the node embedding of node v.

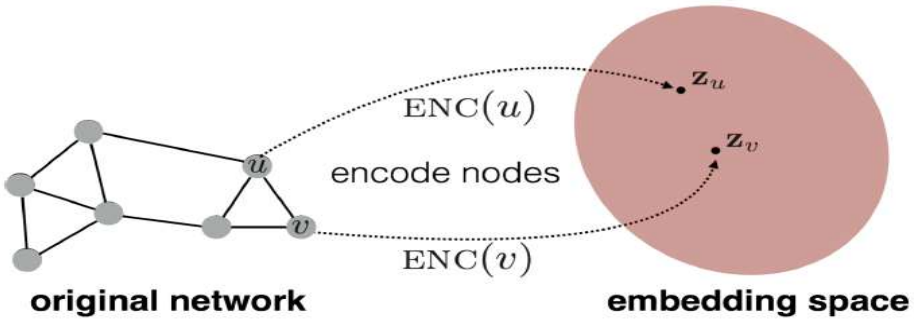


Figure 6– Node Embedding of a Graph

** - The figure was taken from the book "Graph Representation Learning" by William L Hamilton*

In figure 4 the similarity between node u and v is defined as

similarity (u,v) = cosine similarity (u,v) = $(\mathbf{Z_v})^T\mathbf{Z_u}/(\mathbf{Norm}\ |\mathbf{Z_v}|*\mathbf{Norm}\ |\mathbf{Z_u}|)$

Building Text Level Graph: A text level graph built from a text/statement /tweet/ review i.e. from the text data. The graph will have a set nodes representing each word in the text.

i.e. $T = \{ \mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}, \ldots\ldots\ldots, \mathbf{r_l} \}$

where, $\mathbf{r_i} = [d_1, d_1, d_1, \ldots\ldots\ldots\ldots, d_d]$ represents the node embedding of i^{th} node in some vector space of d-dimension word encoding. Each edge starts with the text represented as a node and ends at the p adjacent nodes.

To summarize the text level graph is represented as

Node $N = \{ \mathbf{r_i} \mid i \in [1, l] \}$

Edge $E = \{ e_{ij} \mid \mid i \in [1, l], j \in [i-p, i+p] \}$

Text Level Graph $T = \{ \mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}, \ldots\ldots\ldots, \mathbf{r_l} \}$ where $\mathbf{r_i} = [d_1, d_1, d_1, \ldots\ldots\ldots\ldots, d_d]$.

Message Passing Mechanism: The main intuition behind message passing in a graph is that each node is not standalone, and it is influenced by the neighbouring nodes and also influences the neighbouring node. The key idea is to generate the local embedding for the node and pass the influence using appropriate message passing function to the neighbouring nodes. This message passing function represents the influence of the neighbouring nodes on the nodes i.e. this representation brings the information of the neighbouring nodes. For a text classification problem this information could be the context of the text that is brought though this message passing mechanism. In this method even the meaning for a polysemous word/phrases (Males of the human species (i.e., man vs. woman)) which has multiple meanings in different context can be derived precisely by the influence of the neighbouring words though the message passing mechanism.

Message passing/Propagation variant to collect the information from neighboring nodes.

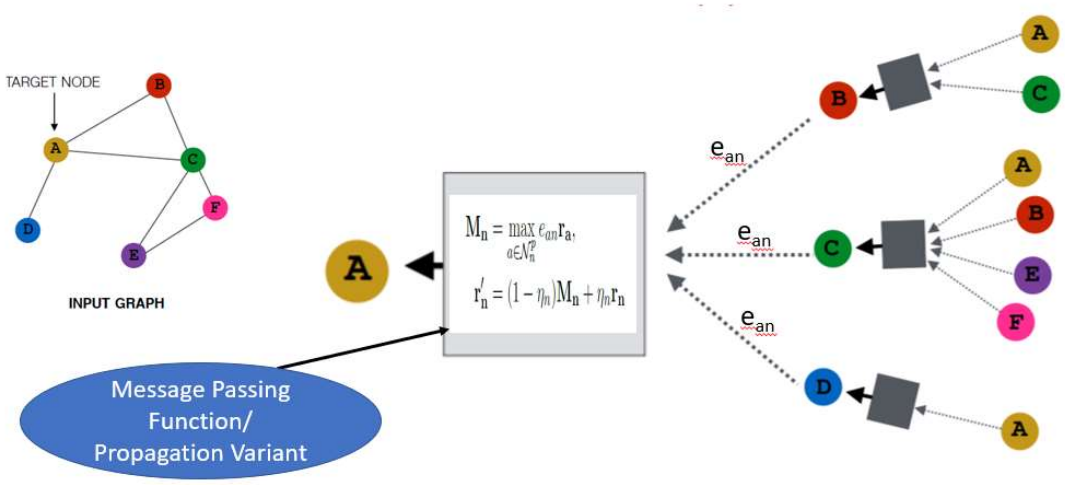


Figure 7 – Message Passing in Graph Network

The message passing mechanism first collects the information from the neighbouring nodes factored with appropriate value and updates it's representation based on it's original representation and the collected information from the adjacent nodes. This is an example of representation learning in graph.
For this project the below message passing mechanism is used

$$M_n = \max_{a \in N_n^p} e_{an} r_a$$

$$r_n' = (1 - \eta_n) M_n + \eta_n r_n$$

where,
r_a – node embedding vector (w2v)
e_{an} - Edge weight of edge connecting to node (nos. of edges depend on the value of p).
η_n – Parameter that determines the amount of massage that should be collected

In the above equation $M_n \in R^d$ is the message that node n receives from it's neighbouring nodes. M_n is nothing but adjacent nodes embedding d-dimension vector factored by edge weight and the final vector is derived by applying the max function for each of the dimension to derive M_n . N_n^p denotes the nodes that represent the nearest p words of n in the original text.

Finally, the representation of all the nodes in a graph (i.e. all the words in the text) are used to predict the label of the text using the below classifier

$$Y_i = \text{SoftMax}(\text{Relu}(W \sum_{n \in N_i} r'_n + b)$$

Where, $W \in R^{dxc}$ is a parameter matrix that maps the vector into an output space, N_i is the node set of text i and $b \in R^c$ is called as bias.

For the above classifier the loss/cost function

$$loss = -g_i \log y_i$$

where g_i is the one hot encoding vector of the true class label.

For a binary class the below loss function is used to avoid the 0 value for the loss function

$$loss\ (g,y) = -\ g_i\ log(y_i)\ -\ (1-\ g_i\ )\ log(1\ -\ y_i)$$

Here, *log* smooths the curves to compute gradient descent easily. The curves are either monotonically increasing or decreasing. To prove the credibility of the cost function, let's take the case where g = 1 and y = 1; log(1) = 0, meaning cost/error is 0. Similarly, when g = 0 and y = 0, log(1-0) = 0 meaning cost/error is 0. Thus, when the predicted outputs are same as expected output, then the cost function would be 0.

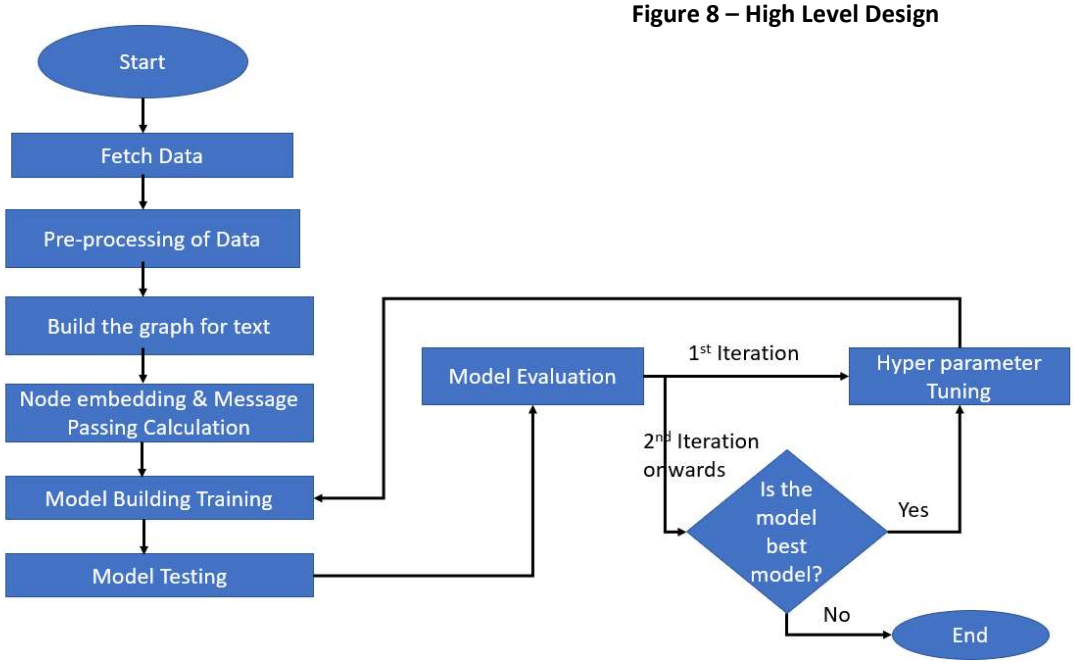
Chapter 5

Experiment Set-up:

For experiment the twitter sentiment dataset[3] was used for conducting the sentiment analysis using the text level graph model. For the word embedding of the work node to a d-dimension vector space I used the GloVe (Global Vector for Word Representation) release by Stanford NLP group researcher Jeffrey Pennington, Richard Socher, and Chris Manning for learning continuous-space vector representations of words. For the purpose of this experiment I used the GloVe vector for sentiment analysis vocabulary with 100 dimensions (twitter.27B, dim = 100).

The GloVe model learns word vectors based on the concept of cooccurrence by examining *word co-occurrences* within a text corpus in this case the twitter word corpus. Before we train the actual model, we need to construct a *co-occurrence matrix* X, where a cell X_{ij} is a "strength" which represents how often the word i appears in the context of the word j.

Figure 8 – High Level Design



1. Fetching of Data & GloVe corpus:

Data can be fetched directly from Kaggle data set but for the convenience during the development testing the data set was downloaded and stored in google drive.

```
# Code to read csv file into Colaboratory:
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[4] df = pd.read_csv(r"/content/drive/My Drive/Dissertation/Kaggle Sentiment140.csv",encoding =DATASET_ENCODING , names=DATASET_COLUMNS)
```

```
glove = torchtext.vocab.GloVe(name="twitter.27B", dim = 100)
```

```
.vector_cache/glove.twitter.27B.zip: 1.52GB [04:46, 5.30MB/s]
100%|█████████| 1192998/1193514 [00:54<00:00, 21662.98it/s]
```

2. Data Pre-processing:

The tweets contain lot of emojis hence the emojis needs to be replace by their respective sentiments so that the emojis cannot be ignored by the model during the training and validation.

```
# Defining dictionary containing all emojis with their meanings.
emojis = {':)': 'smile', ':-)': 'smile', ';d': 'wink', ':-E': 'vampire', ':(': 'sad',
          ':-(': 'sad', ':-<': 'sad', ':P': 'raspberry', ':O': 'surprised',
          ':-@': 'shocked', ':@': 'shocked',':-$': 'confused', ':\\': 'annoyed',
          ':#': 'mute', ':X': 'mute', ':^)': 'smile', ':-&': 'confused', '$_$': 'greedy',
          '@@': 'eyeroll', ':-!': 'confused', ':-D': 'smile', ':-0': 'yell', 'O.o': 'confused',
          '<(-_-)>': 'robot', 'd[-_-]b': 'dj', ":'-)": 'sadsmile', ';)': 'wink',
          ';-)': 'wink', 'O:-)': 'angel','O*-)': 'angel','(:-D': 'gossip', '=^.^=': 'cat'}
```

The tweets also contain number irrelevant words e.g. URL, www, https etc. that do not have any impact on the sentiment that the tweet depicts, hence those words need to be removed from the tweets using the below pre-processing function.

```
#User Defined function to remove patten
#Removing Twitter Handles(@user)
#User Defined function to remove patten
#Removing Emojis

def preprocess(tweet):
    # Defining regex patterns.
    urlPattern        = r"((http://)[^ ]*|(https://)[^ ]*|( www\.)[^ ]*)"
    userPattern       = '@[^\s]+'
    alphaPattern      = "[^a-zA-Z0-9]"
    sequencePattern   = r"(.)\1\1+"


    # Replace all URls with 'URL'
    tweet = re.sub(urlPattern,' URL',tweet)
     # Replace all emojis.
    for emoji in emojis.keys():
        tweet = tweet.replace(emoji, "EMOJI" + emojis[emoji])
    # Replace @USERNAME to 'USER'.
    tweet = re.sub(userPattern,' USER', tweet)
    # Replace all non alphabets.
    tweet = re.sub(alphaPattern, " ", tweet)

    return tweet
```

```
def preprocess_2(tweet):
      sequencePattern   = r"(.)\1\1+"
      tweet = re.sub('www',' ', tweet)
      tweet = re.sub('WWW',' ', tweet)
      tweet = re.sub('USER',' ', tweet)
      tweet = re.sub('user',' ', tweet)
      tweet = re.sub('URL',' ', tweet)
      tweet = re.sub('url',' ', tweet)
       # Replace 3 or more consecutive letters by ' '.
      tweet = re.sub(sequencePattern,' ',  tweet)
      return tweet
```

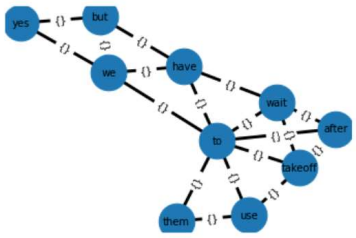
3. Text Level Graph Building

The function “Graph” takes the input nodes as a list and the value of p (an integer value). The value of p defines the edge connectivity between the nodes. For this experiment value of p is taken as 2 i.e. p=2.

```
[36] #Function to create a graph from the node list and p value(defines the maximum nos of edge 1 node can have.)
     def Graph(node,p):
       G = nx.Graph()
       G.add_nodes_from(node)

     #Creating the edges from the node list based on value of p (i.e. if p =2 then each node is connected to previous 2 nodes.).
       edge = []


       for i in range (0,len(node)):
           for j in range (1,(p+1)):
               if((i+j)< len(node)):
                   edge.append([node[i],node[i+j]])
       G.add_edges_from(edge)
       return G
```

```python
# This function prints a given graph

def print_graph(graph):

    pos = nx.spring_layout(graph)  # positions for all nodes

    # nodes
    nx.draw_networkx_nodes(graph, pos, node_size=1200)

    # edges
    nx.draw_networkx_edges(graph, pos, edgelist=graph.edges, width=3)

    # labels
    nx.draw_networkx_labels(graph, pos, font_size=10, font_family="sans-serif")

     # labels
    nx.draw_networkx_edge_labels(graph, pos, font_size=10, font_family="sans-serif")

    plt.axis("off")
    plt.show()
```

```python
#Example of Text Level Graph Creation using the value of p=2.
ex =1
node1 = dataset_text[ex].split()
Graph1 = Graph(node1,2)
print_graph(Graph1)
```



4. Updating the Adjacency Matrix:

The adjacency matrix created in the above step has the value M_{ij} as 1 or 0 depending on whether there exists any edge between the nodes or not. However, this has to be trained and updated with the value of the co-occurrence probability of the 2 words represented by the nodes. In Glove word embedding method the co-occurrence probability between 2 words i and j is given as

$$\text{CosineSimilarity(u, v)} = \frac{u.v}{||u||_2||v||_2} = cos(\theta)$$

Where u and v are the word vectors represented by the word embedding and $||u||_2$ and $||v||_2$ are norm of word vector u and v.

```python
def Update_Prob_Adj_Mat(Graph):
[37]  node = list(Graph.nodes)
      Adj_Matrix = nx.to_numpy_matrix(Graph)
      Adj_Matrix = np.array(Adj_Matrix)
      m = Adj_Matrix.shape[0]
      n = Adj_Matrix.shape[1]
      for i in range(0,m):
        for j in range(0,n):
          if (i != j):                    #For i=j the adjacency matrix element will be always 0 hence this should be excluded.
            vector1 = glove[node[i]]
            vector2 = glove[node[j]]
            normalize_vector1 = tf.nn.l2_normalize(vector1,0)  # Normalize the word vector
            normalize_vector2 = tf.nn.l2_normalize(vector2,0)  # Normalize the word vector
            # Calculate the norm value for both the word vector.
            a = tf.reduce_sum(tf.multiply(normalize_vector1,normalize_vector1))
            b = tf.reduce_sum(tf.multiply(normalize_vector2,normalize_vector2))
            a = tf.cast(a, float, name=None)
            a = a.numpy()
            b = tf.cast(b, float, name=None)
            b = b.numpy()
            if (a!=0 and b !=0):    # If any one of the norm of the word vector is 0 then it should be excluded as the cosine similarlity will be nan.
              prob =  1 - spatial.distance.cosine(vector1, vector2) #cosine_similarity between 2 glove vector
              Adj_Matrix[i][j] = prob
      return  Adj_Matrix
```

5. Node embedding and Message Passing:

The below function transforms the word (represented as node in the graph) to the corresponding GloVe vector.

```
#Node embedding function
def node_embedding(node):
  x= len(node)
  y = 100 #Number of Dimension for Node embedding
  w2v = np.empty((x,y), float)
  for i in range (0, len(node)):
    if (i< len(node)):
      T = glove[node[i]]
      Vec = T.numpy()
      w2v[i] = Vec
  return w2v
```

The below function calculates the below message passing vector update the same to prepare the new word2vector matrix.

$$M_n = \max_{a \in N_n^p} e_{an} r_a$$

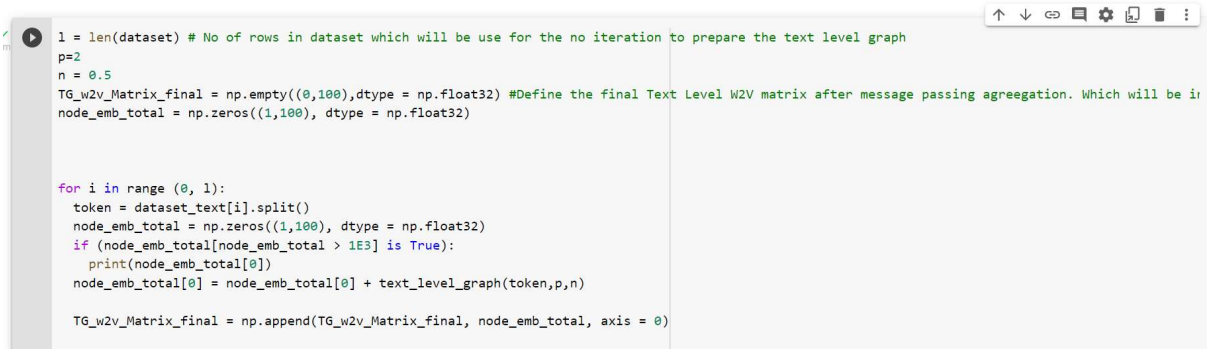
$$r_n' = (1 - \eta_n) M_n + \eta_n r_n$$

```
#Message Passing between the nodes of the text level graph

def msg_pass(node,Adj_Mat,w2v,n):    # n is a trainable variable(value between 0 to 1) for node n that indicates how much information of rn should be
  x= len(node)
  y = 100 #Number of Dimension for Node embedding

  Mn_Temp = np.empty((4,y),float) # Temporary Matrix to hold the calculation of w2v[row] * edge[i][j] weight
  Mn = np.empty((1,y), float) #Matrix to captur the highestest value in each dimension for each node(word) processing
  Temp = np.empty((1,100), float) # Temporary Matrix used for capturing the value of multiplication of the edge weight and w2v of that word.
  w2v_new_Temp = np.empty((x,y), float)#Matrix to capture all Mn for each node(word) in the graph
  w2v_new= np.empty((x,y), float)#Final word embeddd Matrix to capture updated with message passing from linked nodes(words)

  for i in range (0, x):
    for j in range ((i-p), (i+p+1)):
      if ( (j>=0) & (j != i) & (j<len(node))):
        Temp = Adj_Mat.item(i,j)*w2v[i]
        Temp = Temp.reshape(1,100)
        Mn_Temp = np.append(Mn_Temp, Temp , axis = 0)
        Mn_Temp = np.delete(Mn_Temp, 0,0)
    Mn = np.amax(Mn_Temp, axis = 0)
    Mn = Mn.reshape(1,100)
    w2v_new[i]= w2v_new[i]+(1-n)*Mn[0] + n*w2v[i]
  w2v_new = np.nan_to_num(w2v_new)
  return w2v_new
```

6. Preparation of Final embedded Matrix of tweet dataset:

This part of the code executes to transform the entire tweets data set to the new embedded vector ready to be fed to the deep learning models.

Preparation of input data matrix 'X' (New embedding matrix updated with the message passing) for the entire Tweeter dataset.



```
l = len(dataset) # No of rows in dataset which will be use for the no iteration to prepare the text level graph
p=2
n = 0.5
TG_w2v_Matrix_final = np.empty((0,100),dtype = np.float32) #Define the final Text Level W2V matrix after message passing agreegation. Which will be ir
node_emb_total = np.zeros((1,100), dtype = np.float32)



for i in range (0, l):
  token = dataset_text[i].split()
  node_emb_total = np.zeros((1,100), dtype = np.float32)
  if (node_emb_total[node_emb_total > 1E3] is True):
    print(node_emb_total[0])
  node_emb_total[0] = node_emb_total[0] + text_level_graph(token,p,n)

  TG_w2v_Matrix_final = np.append(TG_w2v_Matrix_final, node_emb_total, axis = 0)
```

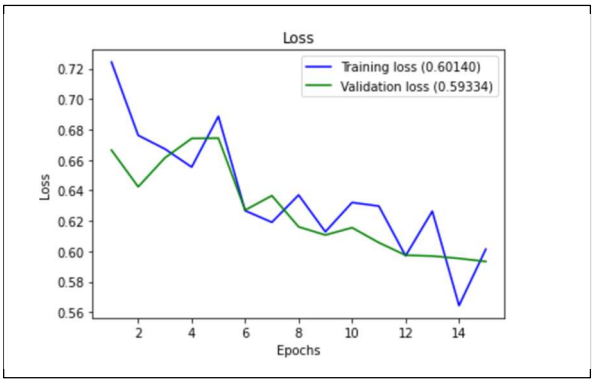
Model Set-up, Tunning and Analysis:

For the experiment we utilized the twitter data set available in public domain (https://www.kaggle.com/kazanova/sentiment140). The initial models were build using the CNN deep learning model. However instead of the entire dataset of 1.5 million records, a sample of 100K records were selected for the experiment purpose to ensure the model can be executed within the available computing resources.

Model 1: *CNN model with 3 layers and dropout of 0.5.*
Optimizer used: Adam Optimizer (learning rate = 0.01)
Loss Function – Binary Cross Entropy
Validation Split – 20%
Batch size = 128
No of Epochs = 15

Model Summary

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer_3 (KerasLayer)   (None, 128)               124642688
_________________________________________________________________
dense_9 (Dense)              (None, 512)               66048
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_10 (Dense)             (None, 256)               131328
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 257
=================================================================
Total params: 124,840,321
Trainable params: 197,633
Non-trainable params: 124,642,688
_________________________________________________________________
```



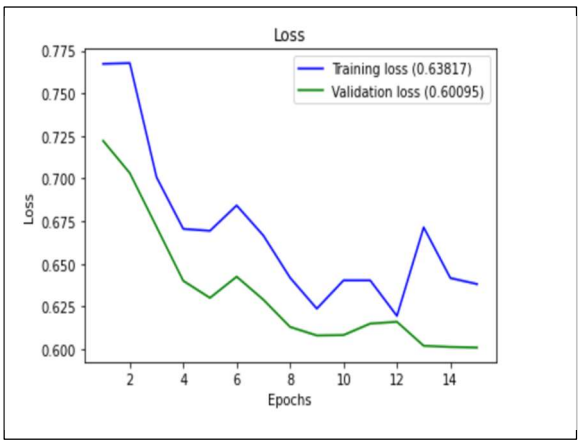
Model Evaluation:

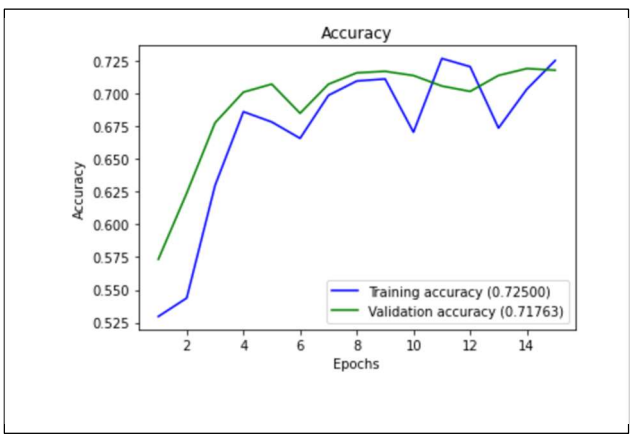
```
625/625 - 2s - loss: 0.5893 - accuracy: 0.7190

Test accuracy: 0.718999981880188
```

Model 2*: Same as Model 1 with the below changes*
Changing the dropout to 0.9 and to layer 4

Model Summary

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer_4 (KerasLayer)   (None, 128)               124642688
_________________________________________________________________
dense_12 (Dense)             (None, 512)               66048
_________________________________________________________________
dense_13 (Dense)             (None, 256)               131328
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_14 (Dense)             (None, 1)                 257
=================================================================
Total params: 124,840,321
Trainable params: 197,633
Non-trainable params: 124,642,688
_________________________________________________________________
```



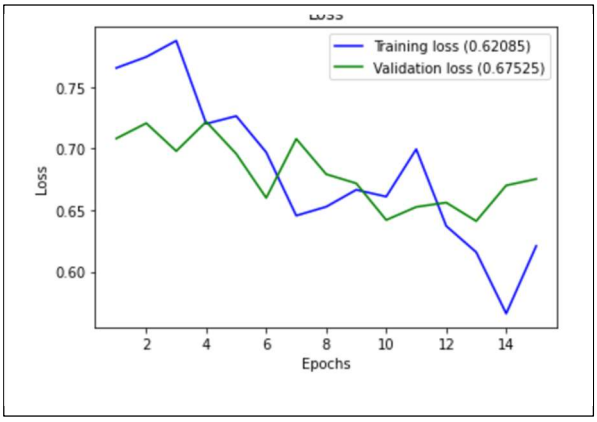


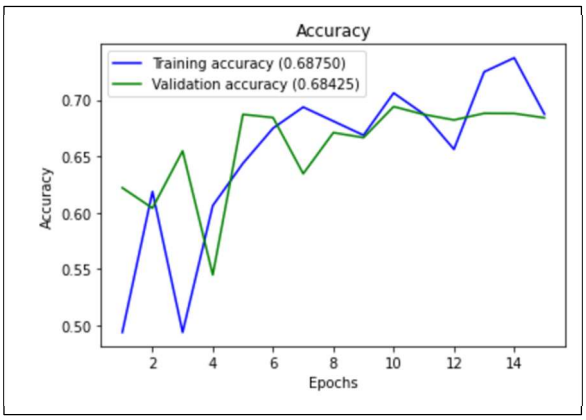
Model Evaluation:

```
625/625 - 2s - loss: 0.6007 - accuracy: 0.7185

Test accuracy: 0.7184500098228455
```

Model 3: *CNN model same as Model 1 with the below change*
Changing the batch size to 32

```
]
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer_5 (KerasLayer)   (None, 128)               124642688
_________________________________________________________________
dense_15 (Dense)             (None, 512)               66048
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_16 (Dense)             (None, 256)               131328
_________________________________________________________________
dense_17 (Dense)             (None, 1)                 257
=================================================================
Total params: 124,840,321
Trainable params: 197,633
Non-trainable params: 124,642,688
_________________________________________________________________
```





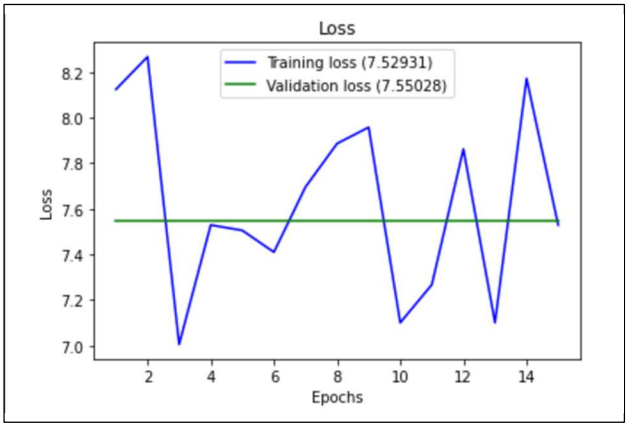
Model Evaluation:

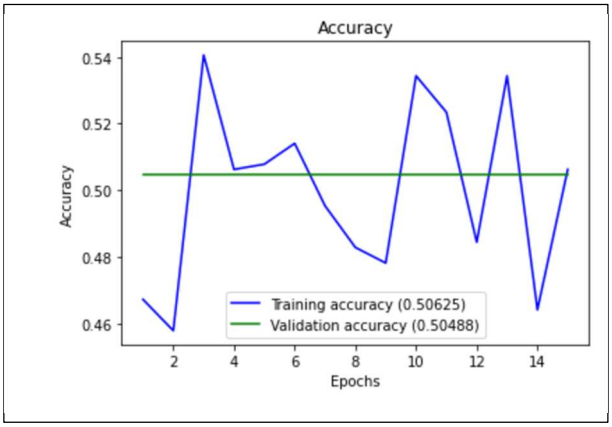
```
625/625 - 1s - loss: 0.6707 - accuracy: 0.6858

Test accuracy: 0.6858000159263611
```

Model 4: Text Level Graph Neural Network

CNN model with 1 layers
Optimizer used: Adam Optimizer (learning rate = 0.01)
Loss Function – Binary Cross Entropy
Validation Split – 20%
Batch size = 128
No of Epochs = 15
n = 0.5
p = 2

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 100)               10100
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 101
=================================================================
Total params: 10,201
Trainable params: 10,201
Non-trainable params: 0
_________________________________________________________________
```



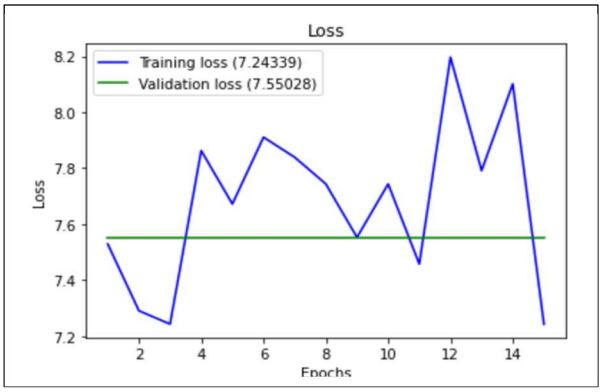


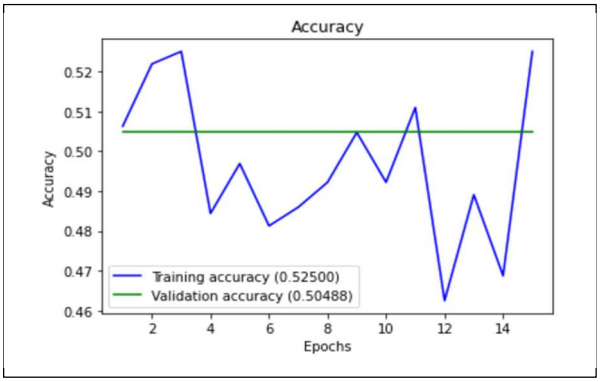
Model Evaluation

```
625/625 - 1s - loss: 7.5918 - accuracy: 0.5021

Test accuracy: 0.5021499991416931
```

Model 5: Text Level Graph Neural Network

Same as model 4
Batch size = 128
No of Epochs = 15
n = 0.05
p = 2



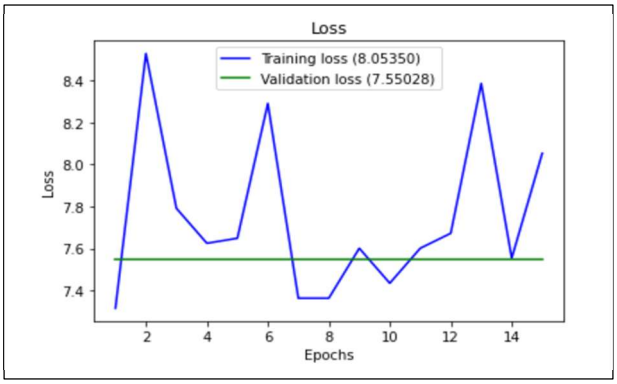


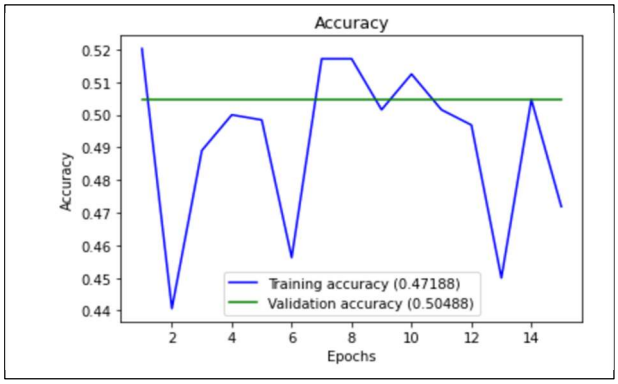
Model Evaluation

```
625/625 - 0s - loss: 7.5918 - accuracy: 0.5021

Test accuracy: 0.5021499991416931
```

Model 6: Text Level Graph Neural Network

Same as model 4
Batch size = 128
No of Epochs = 15
n = 0.9
p = 2





Model Evaluation

```
625/625 - 0s - loss: 7.5918 - accuracy: 0.5021

Test accuracy: 0.5021499991416931
```

Table - 1

CNN Models Result Analysis								
Model	No of Layer	Dropouts	Batch Size	Training Accuracy	Testing Accuracy	Loss Function	Training Loss	Validation Loss
Model 1	3	0.5 (Layer 2)	128	72.03%	71.86%	Binary Cross Entropy	0.601	0.593
Model 2	3	0.9 (Layer 4)	128	72.50%	71.85%	Binary Cross Entropy	0.638	0.6
Model 3	3	0.5 (Layer 2)	32	68.75%	68.52%	Binary Cross Entropy	0.621	0.6752

Table - 2

Text Level Graph Neural Network Result Analysis								
Model	No of Layer	Value of n	Batch Size	Training Accuracy	Testing Accuracy	Loss Function	Training Loss	Validation Loss
Model 4	1	0.5	128	50.63%	50.21%	Binary Cross Entropy	7.529	7.55
Model 5	1	0.05	128	52.50%	50.21%	Binary Cross Entropy	7.24	7.55
Model 6	1	0.9	128	47.18%	50.48%	Binary Cross Entropy	8.05	7.59

Chapter 6

Observation:

From the Table 1 & Table 2 it can be concluded that both the approach do-not result in any underfitting or overfitting issues. It may be notice from the analysis that in case of CNN model the training and testing loss is gradually decreasing as the epochs progress, same observation may be seen for the training and validation accuracy for CNN. However, for Text Level GNN both the training and validation loss as well as the training accuracy is oscillating throughout the epochs. The training loss and validation loss in case of Text Level GNN is almost 10 times the loss compared to CNN for the same loss function. The interesting observation is regarding the testing accuracy which remains constant irrespective of changes to the value of n.

Conclusion:

From the above observation it may be concluded that for the twitter sentiment dataset CNN model gives better accuracy. The learning improves as the epochs progress in case of CNN. None of the models whether CNN or Text Level GNN has shown any sign of underfitting or over fitting issue.

Direction of Future Work:

Due to the time limit provisioned for this project as well as the availability of the computing resources the Graph CNN model cannot be built and tested for the data set. It would be an interesting idea to use the Graph CNN model for the entire corpus and to load the entire graph to the memory to be fed to the CNN model. Apart from that further analysis can be carried out on the Text Level GNN by tunning the value of p to 3,4 etc and by training the parameter n by assigning the unigram probability for the word represented by the node instead of the constant value for all the text level graph.

Appendix-A

Project Plan & Deliverables

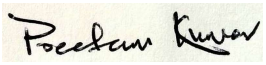
#	Task	Expected date of completion	Actual date of Completion	Names of Deliverables	Status
1	Literature review	20/04/2021	20/04/2021		Complete
2	Analysis and Design	20/05/2021	20/05/2021	Design Document	Complete
3	Data Collection	25/05/2021	31/05/2021	Data Set	Complete
4	Model Building	15/06/2021	30/06/2021	Completion of Model Building (Code Completion)	Complete
5	Experiments & Result Analysis	25/06/2021	25/07/2021	Result Analysis	Complete
6	Documentation and Review	30/06/2021	31/07/2021	Results Compilation and Observation	Complete
8	Technical Report preparation	15/07/2021	08/07/2021	Dissertation Final Report	Complete

Illustrative examples of citation of Bibliography / References:

1. Text Level Graph Neural Network for Text Classification, Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang and Houfeng WANG, MOE Key Lab of Computational Linguistics, Peking University, Beijing, 100871, China
2. Graph Representation Learning *by William L Hamilton*
3. Data Set- https://www.kaggle.com/kazanova/sentiment140
4. Mining of Text Data by Charu C Aggarwal (Chapter 6, page 163 to 222)
5. https://nlp.stanford.edu/projects/glove/
6. http://web.stanford.edu/class/cs224n/index.html - Natural Language Processing with Deep Learning
7. Graph Convolutional Networks for Text Classification *by Liang Yao, Chengsheng Mao, Yuan Luo*
8. Character-level Convolution Network for Text Classification *by Xiang Zhang, Junbo Zhao & Yann Lecum*
9. **GloVe: Global Vectors for Word Represent**ation *by Jeffrey Pennington, Richard Socher, Christopher D. Manning*
10. *https://www.tensorflow.org/*
11. *http://web.stanford.edu/class/cs224w/ - Machine Learning with Graphs*

Check list of items for the Final report

Is the Cover page in proper format?	**Y** / ~~**N**~~
Is the Title page in proper format?	**Y** / ~~**N**~~
Is the Certificate from the Supervisor in proper format? Has it been signed?	**Y** / ~~**N**~~
Is Abstract included in the Report? Is it properly written?	**Y** / ~~**N**~~
Does the Table of Contents page include chapter page numbers?	**Y** / ~~**N**~~
Does the Report contain a summary of the literature survey?	**Y** / ~~**N**~~
Are the Pages numbered properly?	**Y** / ~~**N**~~
Are the Figures numbered properly?	**Y** / ~~**N**~~
Are the Tables numbered properly?	**Y** / ~~**N**~~
Are the Captions for the Figures and Tables proper?	**Y** / ~~**N**~~
Are the Appendices numbered?	**Y** / ~~**N**~~
Does the Report have Conclusion / Recommendations of the work?	**Y** / ~~**N**~~
Are References/Bibliography given in the Report?	**Y** / ~~**N**~~
Have the References been cited in the Report?	**Y** / ~~**N**~~
Is the citation of References / Bibliography in proper format?	**Y** / ~~**N**~~



Signature of Student **Signature of Supervisor**