Twitter Network Analysis

Team Members:
19BCE0990 (Neel Choksi)
17BCE0467 (Venkata reddy)
Reg. No (Name)
Reg. No (Name)
Reg. No (Name)

(Maximum five members)

Report submitted for the
First Project Review of

Course Code: CSE3021
Social and Information Network

Slot: A2 Slot

Professor: Dr. Ilanthenral Kandasamy

1. Introduction:

Social network analysis is a field where the boundary specification plays a vital role.Majority of the people in the world are on social networking websites for most of the time of the day. There are many online communities formed which consist of like minded people interacting among each other. The analysis of such networks can be used for various application including identifying a target audience for the business, healthcare, advertisements and to identify the behavior of people around the world to certain major events that take place .Twitter is a social media platform where people are freely allowed to express their emotions against issues and events in the society. Twitter lets the users post a tweet which is a message containing 280 characters along with tags. Tags are used to target a specific event. The users can follow other users and post tweets to express their views. Analysis of the sentiments of people becomes very vital when the issues and events are very sensitive.

In this project we will be attempting to perform sentiment analysis on twitter tweets and building a model for the same. An analysis of the twitter social network will be done using the graph representation of the social network with the nodes as the tweets and the links as the relation between the tweets. We will attempt to optimize the method of finding the relations between two tweets. A study on graph neural networks will be carried out which will be aimed at representation of a social network in a vector form which is used in various machine learning algorithms.

2. Literature Review Summary Table

Authors and Year (Reference)	Title (Study)
Martino, Francesco & Spoto, Andrea. (2006) [1]	Social Network Analysis: A brief theoretical review and further perspectives in the study of Information Technology.
Otte, Evelien & Rousseau, Ronald. (2002). [2]	Social Network Analysis: A Powerful Strategy, also for the Information Sciences.
Grandjean, Martin. (2016)[3]	A social network analysis of Twitter: Mapping the digital humanities community.
R. Abascal-Mena, R. Lema and F. Sèdes, 2014 .[4]	From tweet to graph: Social network analysis for semantic information extraction.
P. M. Dudas, 2013[5]	Cooperative, dynamic Twitter parsing and visualization for dark network analysis

Tan Q, Liu N and Hu X (2019) [6]	Deep Representation Learning for Social Network Analysis

ResearchPaper 1:

1.Authors and Year (Reference):

Martino, Francesco & Spoto, Andrea. (2006)

2.Title(Study):

Social Network Analysis: A brief theoretical review and further perspectives in the study of Information Technology

3.Concept/Theoretical model/ Framework:

Social Network Analysis is used in psychology and social science.It involves the study of relations between individuals.This paper covers the review of the perspective of viewing social networks as a set of relations. It gives the description of resources and principal topics of Social Network Analysis(SNA).It also states the relationship of social networks with information technology with finally showing how this approach is useful to study some aspects of the web.

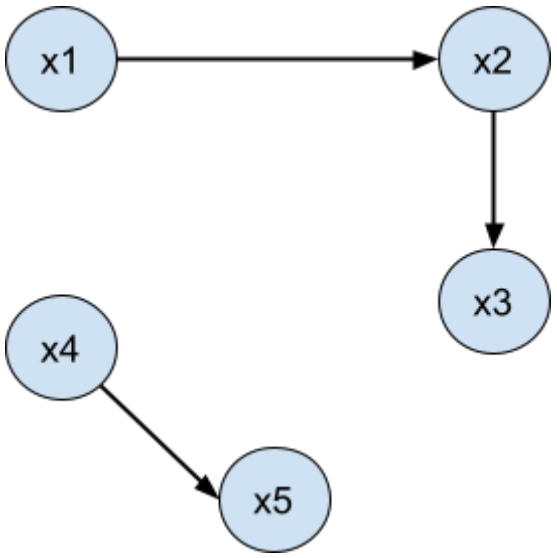
Social aggregate a group of people gathered at a place having no particular thing in common other than their location. The phenomenon of formation of social aggregates is known as social aggregation. It can be represented in terms of units composing aggregation and relation between the units can be identified.This is the representation of social structure known as a social network.

Every unit is considered to be a social actor.The social actor is called a node. There is a relation between the nodes in a social network . This relation is called a linkage which determines the flow between units. The set of possible relations in a social network is infinite. The relations can be of different types. They can be based on acquaintance,kinship,evaluation of another person, need of commercial exchange, physical connections, presence in a web-page or a link to another page.The objects under observation are not the units but its attributes,relationships between individuals and their structure.This leads to the analysis of social processes as product of relationships among social entities.In the SNA there are two types of variables namely the structural and composition . The structural variables represent different kinds of ties between social actors and the composition variables consist of the attributes of the actors.

This paper further describes the history,types of studies involved,resources ,sampling and boundary specification for different types of networks in the field of Social Network Analysis.

The methods suggested for the study of social networks are set theoretic relation, graph theory, algebraic semigroup theory, statistical models and Dynamic Network Analysis. These can be broadly classified into two groups namely the graph theory and the Algebraic Theory of Semi-Groups. Their common base is explained.

A relation in a set of units can be represented as a set of ordered pairs of units. For instance {(x1,x2),(x2,x3),(x4,x5)}. The same can be represented as a graph using the units as nodes and relations as links between the nodes.



The same can be represented using a matrix

		to				
		x1	x2	x3	x4	x5
from	x1	0	1	0	0	0
	x2	0	0	1	0	0
	x3	0	0	0	0	0
	x4	0	0	0	0	1
	x5	0	0	0	0	0

Graph theory has been used to show the structural properties of a network. Degree , centrality , betweenness are the evaluation measures of a graph.

Another possible definition could be consideration of a set of relations R ={R1, R2, ...Rn} and units x and y are structurally equivalent if and only if they are for all nodes xRk iff yRk and kRx iff kRy.

4.Dataset details/Analysis:

Mention of the study of the first dataset by Freeman in 1986. It was a dataset consisting of measurements of computer mail interactions.The types of networks identified are internet network, hyperlink network, computer mediated network.

5.Relevant Finding:

Formal aspects of the topic are complex,advanced and are based on strong and consolidated mathematical theories. Graph theory, semi-groups algebra. There are various benefits and potentials of these approaches.Social Network Analysis is a powerful tool for examining social aspects of the web.It is easy to examine, has a simple sampling procedure.Algebra plays a powerful role in determining the structure of networks.Actors in a network use different communication instruments to relate to each other. The relation would be constructed on the basis of the used instrument.

6.Limitations/Future Research/Gaps identified:

The real meaning of hyperlink as a communication relation is yet to be investigated.A dynamic way of measuring the evolution of social networks over time can be identified.

ResearchPaper2:

1.Authors and Year of Reference:

Otte, Evelien & Rousseau, Ronald. (2002).

2.Title(Study):

Social Network Analysis: A Powerful Strategy, also for the Information Sciences.

3.Concept/Theoretical model/ Framework:

Social Network Analysis is not a formal theory in sociology , it is a strategy for investigating social structures.Idea is applied in many fields.We study the influence in information.Information scientists study publication, citation, co-citation networks, collaboration structures, and other forms of social interaction networks.Internet represents a social network of huge scale .SNA is applied to it and further theories concerning free market economy and geography and transport networks are studied. A graph of Social Network

Analysis is documented.A co author network of SNA is drawn and the centrality measures are calculated. Network studies include the internet which is one large network but the internet here acts as medium.

Social network theory directly influences the way researchers formulate ideas of the web . The field of sociology is closely related to the social network study.Every network can be represented as a graph .A structure studied since Euler solved the Konigsberg bridge problem.

4.Dataset details/Analysis:

The paper explains the concepts of Social Network Analysis, Directed graphs, undirected graphs, components, density, centrality , cliques.Development and growth of SNA. Explains a co-authorship network and shows bibliometric analysis of SNA database.Finally depicts the use of network analysis in information sciences.

5.Relevant Finding:

SNA is an example of an idea that can be applied in many fields .It requires mathematical graph theory basis.It has become an approach adopted and integrated with sociology , information sciences, computer sciences, geography . Growth of SNA within sociology is documented.SNA can be linked to work in information sciences.It has brought more information scientists to be interested in the field of Internet including webometrics and cybernetics.

6.Limitations/Future Research/Gaps identified:

SNA will find a place in Information Sciences. Small world theory by Manfred Kochen shows that he was aware about the rise of social networks ahead of his time.Relationship between the networks, percolation theory and informetric laws are pointed out which can be an area of further research for computer and information scientists.

ResearchPaper3:

1.Authors and Year of Reference:

Grandjean, Martin. (2016).

2.Title(Study):

A social network analysis of Twitter: Mapping the digital humanities community.

3.Concept/Theoretical model/ Framework:

This paper states that defining digital humanities is an endless debate if the discussion is about the boundaries of concept as academic discipline.It is possible to analyse actors and this field through Twitter which is a social media widely used in community of practice.The analysis is done on 2500 users.A visualization who follows who in the network.Graph is used since it allows us to highlight the structure of the networks relationships and identify the users who hold a particular position in the network.Clustering will be done based on the linguistic groups in the network.

Social network is the main place for digital humanity.The goal is to observe it in order to offer a transversal view of the movement.It is difficult to map it with traditional methods.

4.Dataset details/Analysis:

Twitter is identified as a growing field of study. A small world is identified.The concept of followers is discussed . The difference in the relations when someone follows you and when you follow someone. Geography of linguistic communities is identified. The structural features of the network are identified.They include in-degree, out-degree, betweenness, eigen vector.

5.Relevant Finding:

It was found out that defining digital humanities as a “community” solves the endless debates on disciplinary boundaries.It does not allow us to know who is practicing them today. A small number of individuals and institutions are gaining attention and the graph is homogeneous around them.Many types of behavior can be deduced from a graph.Structural characteristics of networks help us identify the users holding specific positions. Language factor affects the structure of the graph. French and german speakers' influence was detected.

6.Limitations/Future Research/Gaps identified:

Any quantification leads to a form of objectification which means that quantifying something can be done only when some parameters and limits are identified that support the quantification.

ResearchPaper4:

1.Authors and Year of Reference:

R. Abascal-Mena, R. Lema and F. Sèdes, 2014

2.Title(Study):

From tweet to graph: Social network analysis for semantic information extraction

3.Concept/Theoretical model/ Framework:

It represents the study and analysis of social network and relation with contents communicated among users. Twitter data is carefully selected and hashtags are fixed. It is studied with other contents that users bring to connection. Separate network of hashtags can be constructed. Networks can be analysed using advanced packages, here they are done using gephi package.The measures provided are degree, betweenness,centrality, communities, longest path, evolution of communication around specified concepts is identified. The study done is in trend of analysis of social networks reveals the social categories, linguistics and their dynamics.Linguistics practices are evolving , the use of emoticons and messages is increasing. Oral and writing relations are the new way of computer mediated communication. Social networks consist of nodes which are equivalent to persons and groups and relationships between these persons determined by their common points of views. Twitter is used for social media and microblogging . 280 character messages are posted on the platform.# are known as hashtags and are used to categorise the tweets , they are aimed towards a particular event or issue.Tags evolve over time.Tweets also involve the use of @ which addresses to another account on the platform. They are a direct determinant for a relationship between two nodes.Hashtags are used more often than long descriptive sentences.Information diffused in twitter can be in variations of diffusion and characterised under various themes.

Twitter communities have evolved using hashtags and retweets.It helps form an ad-hoc public around specific issues.

4.Dataset details/Analysis:

This paper uses hashtags as basic elements of composition of meassaged, and tries to interpret the meaning of relations between concepts and hashtags over time.Visualization of the information network is carried out, quantitative and qualitative analysis of political and social movements is carried out.

5.Relevant Finding:

Twitter is a tool which has impacted society on a deeper level and transformed many cultures. It is a new platform form decentralized and unfiltered participation of the people. There are about 350 million active users on twitter.This article proposes a methodology allowing identification of groups where interaction of words is particularly intense.Subgroups communities and clusters are identified.

6.Limitations/Future Research/Gaps identified:

Selection of tweets should be done from a corpus based on centrality and subgroups to eliminate tweets that do not have any relevance with the movement.User group study is crucial so it should be researched upon more. Real time information retrieval techniques should be adopted.

ResearchPaper5:

1.Authors and Year of Reference:

P. M. Dudas, 2013

2.Title(Study):

Cooperative, dynamic Twitter parsing and visualization for dark network analysis

3.Concept/Theoretical model/ Framework:

Network is based on twitter data for Social Network Analysis.There is a need for real time analysis. Dark Network Analysis is a special field focusing on criminal, terrorist,and people of interest networks where the evaluation of information is done quickly and decision making is crucial.

4.Dataset details/Analysis:

It is a platform for visualization named Dynamic Twitter Analysis. It uses real time information with Twitter. Tweets are tagged on google maps. Storage for long term analysis is also accommodated. Social Network Analysis visualization is carried out here. User Interface built is dynamic and up to date and geographic specific, user can select based on properties built from key dark network utilities.

5.Relevant Finding:

It is a cooperative, dynamic and interactive visualization tool used to collect and navigate through tweets.Dynamic network visualization can be done . It has a tweet wall post, mapped with geotagged tweets.Homogeneous visualization allows the user to explore the data set using timestamps.

6.Limitations/Future Research/Gaps identified:

Real time information is provided but the information retrieval can be optimized.

ResearchPaper6:

1.Authors and Year of Reference:

Tan Q, Liu N and Hu X (2019)

2.Title(Study):

Deep Representation Learning for Social Network Analysis

3.Concept/Theoretical model/ Framework:

Social Network Analysis is an important problem of data mining. To understand a social network it has to be represented into low dimensional form .Network embedding is the technique to represent the network data into low dimensional forms like nodes and links , preserving the topology structure and the attribute information of the nodes.The applications of representing a network embedding involve classification, link prediction, anomaly detection, clustering.

Deep neural networks are gaining traction.This paper is a comprehensive review of the current literature in the network representation learning and shows the utilization of neural network models. First the basic models for learning node representations are portrayed , then the extensions of the base models are shown.More complex scenarios are taken into account including attributed networks, heterogeneous networks , dynamic networks.Techniques for embedding subgraphs, applications of network representation learning are shown.

Social networks have connected web users across the world.Patterns can be observed from the links formed between the users (nodes).The analysis from the social network can be valuable for various aspects like finding the target

audience for businesses, healthcare, marketing using recommendation systems,academic networks. The problem to tackle is how to convert the non-linear data into a form which the machine understands . Once it is converted into a machine understandable form , machine learning algorithms can be applied on the data.The methods which exist to convert network data into structured form involve the use of graph statistics involving clustering coefficients and degrees.The goal is to design a transformation function that can map the data into a vector representation . Machine learning algorithms for classification and clustering can be applied on the vectorized inputs.This paper shows the various models of representation of networks using neural networks.There are three kinds of models: embedding look-up based, autoencoder based, graph convolutional based.

4.Dataset details/Analysis:

The notations and problem definitions are shown .Neural network based models are identified consisting encoder perspective, embedding look-up table models, and auto encoder techniques.Graph Convolutional technique, Subgraph embedding are discussed. Applications include Node classification , link prediction , anomaly detection , node clustering.

5.Relevant Finding:

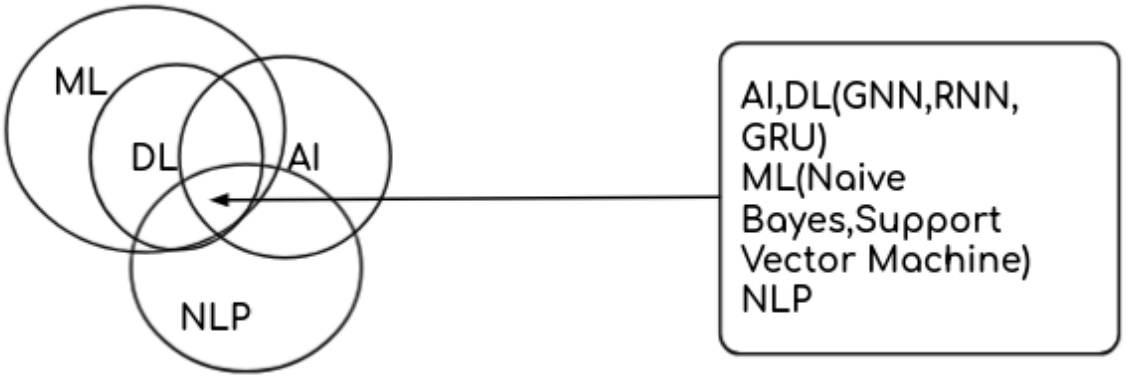
Dynamic networks, hierarchical network structure, heterogeneous networks.

6.Limitations/Future Research/Gaps identified:

Scalability is there but can be improved . It has interpretability but more results can be drawn using this approach of graph neural networks.

3. Objective of the project:

The main objective of this project is to propose a graph representational structure for the twitter social network so that it could be fed into a Graph Neural Network to perform tasks like Node Classification and graph classification. The objective is also to deduce sentiments of the tweets which would be playing a role as edge attributes in the graph representation.Areas covered are Graph Neural Networks, LSTM and GRU, Machine Learning algorithms including Support Vector Machine and Naive Bayes.



4. Innovation component in the project:

There are a lot of datasets available to cluster the twitter data but only few to represent it as a graph.This project carries out sentiment analysis on twitter data using Support Vector Machine, Naive Bayes and Feed forward Neural Network along with visualization of the word embeddings using LSTM. This project proposes a methodology to represent twitter as a graph so that it can be fed into a Graph Neural Network.

5. Work done and implementation

a. Methodology adapted:

Naive Bayes , Support Vector Machine and a feed forward Neural network was used for the classification of tweets to represent for a graph. Word Embedding Visualization.

Preprocessing : Bag of words

dictionary: {0:"recurrent",1:"neural",2:"network",3:"artificial",4:"intelligence"}

example sentence: "artificial neural network and the recurrent neural network"

we don't remove the stopwords here since Recurrent neural networks are based on sequential data so here the sequence of all the words matter even the stopwords.

	artificial	neural	network	and	the	recurrent	neural	network	final vector
recurrent	0	0	0	0	0	1	0	0	1
neural	0	1	0	0	0	0	1	0	2

network	0	0	1	0	0	0	0	1	2
artificial	1	0	0	0	0		0	0	1
intelligence	0	0	0	0	0	0	0	0	0

final vector: {1,2,2,1,0}

The final vector is the one hot encoding of the sentence with respect to the vocabulary which is a dictionary containing a bag of words in the whole corpus.

Support Vector Machine

Methodology

1. takes in the vectors of the words
2. transforms it into a higher dimensional space and where the words can be compared to each other.
3. Finds the similar words using the kernel function .
4. Kernel function is derived by scaling the input points to a higher dimensional vector space and a model is formed that transforms the points to a higher dimensional vector space.
5. The points that are close to each other are given more priority and a hyperplane is formed which separates most of the points.
6. Training: It finds a decision boundary(hyperplane) and maximizes the margin between the boundary and the support vectors . support vectors are the vectors nearest to the boundary.
7. To train a Support Vector Machine Model the words of the tweets are represented as vectors. The vectors are in n dimensional space since a bag of word encoding is used.All the word vectors have the same dimension. Support Vector Machine moves the data into a higher dimensional space.
8. Now, it separates the data into groups using hyperplanes. Kernel function is used to build the hyperplane by finding support vectors in higher dimensions.
9. The support vectors are the vectors nearest to the hyperplane for a class. The best hyperplane is found using the cross validation which means the loss is calculated for each hyperplane and parameters are modified to find the best hyperplane separating the data.

Naive Bayes

Methodology

Bayes Theorem

$P(A|B) = P(B|A).P(A)\}/P(B)$

for our case:

$P(y|X) = P(X|y).P(y)P(X)$

1. y : class label
2. X : feature vector

feature vector X :

$X = (x_{1},x_{2},x_{3},....,x_{n})$

Assume :

all the features are mutually independent

$P(y|X) = P(x_{1}|y).P(x_{2}|y)...P(x_{n}|y).P(y)/P(X)$

P(y|X) :posterior probability

$P(x_{i}|y)$: class conditional probability

P(y) : Prior probability of y

P(X) : Prior probability of X

select the class with the highest probability

$y=argmax_{y}P(y|X) =\{argmax_{y}P(x_{1}|y).P(x_{2}|y)..P(x_{n}|y).P(y)\}/\{P(X)\}$

ignore the denominator since it does not depend on the class label

y = argmax_{y}P(x_{1}|y).P(x_{2}|y)..P(x_{n}|y)

Since the probability is small , we can apply log to calculate the max.

argmax means that the argument y, x values will be passed for the entry in the dataset . x1 .. xn are the features and the y is the label for the entry in the dataset . The max value y is found for the data

y = argmax_{y}log(P(x_{1}|y))+log(P(x_{2}|y))+..+log(P(x_{n}|y))+log(P(y))

Prior Probability : P(y) : frequency

Class Conditional probability :P(x_{i}|y)

In the case of categorical variables, such as counts or labels, a multinomial distribution can be used

Feed Forward Neural Network

Weights , biases and activation functions comprise most of the part of a neural network . The neural network is trained using the following methodology.

- Data Loading :
- Preprocessing:
- Train Test Split:
- Model Declaration
- Training Pipeline
    - Forward pass: prediction, loss
    - Backward pass: gradients
    - Update weights: gradient descent
- Validation and accuracy
- layer 1: x1 = W1 * X + b1
- layer 1(activation): h1 = Relu(x1)
- layer 2: x2 = W2 * h1 + b2log()
- output : p=sigma(x2)
- loss : -(ylog(p)+(1-y)log(1-p))
- gradient : d L(W1,b1,W2,b2)/ dW1 = (dL/dp) * (dp/dx2) * (dx2/dh1) * (dh1/dx1) * (dx1/dW1)
- optimizer : Parameter update :
- W1 =W1 - alpha (dL/dW1)

The Feed forward neural network , naive bayes classifier, and support vector machines do not consider the order of the words . The solution to that is Recurrent Neural Networks.

Word Embeddings visualization using LSTM.

Representation of the words

1. Bag of words , unique words using all words. one -hot encoding using the dictionary that you created. sentence as a vector with 1 and 0 for occurrence and non occurrence of the term in the sentence. Length of the encoding is the same as the length of the dictionary.
2. Integer encoding. assign a number to each unique word. for the sentence, the words will be represented according to the number

Cons:

- cannot portray the relationships between the words
- reason: basis in higher dimensional space, vectors are orthogonal, dot product is 0. No projection on any axis therefore they cannot show the relationships.
- The vectors are too sparse.
  Solution:

3. Word Embedding New space, where the words are transformed to , it is a hyperparameter of the model like the number of hidden layers in a neural network.

Dot products can be taken and the relations between words can be represented.

strong correlation of words. The model takes words,puts through the embedding layers., good or bad review,matches the training label.and then back propagates through the model and changes parameters.

Model now can predict positive and negative and can also show a correlation between words.

b. Dataset used:

a. Dataset used is Sentiment140 dataset [7].

c. Tools used:

The tools used to implement the preprocessing are spacy[8] which is a library in python that helps carry out preprocessing tasks.

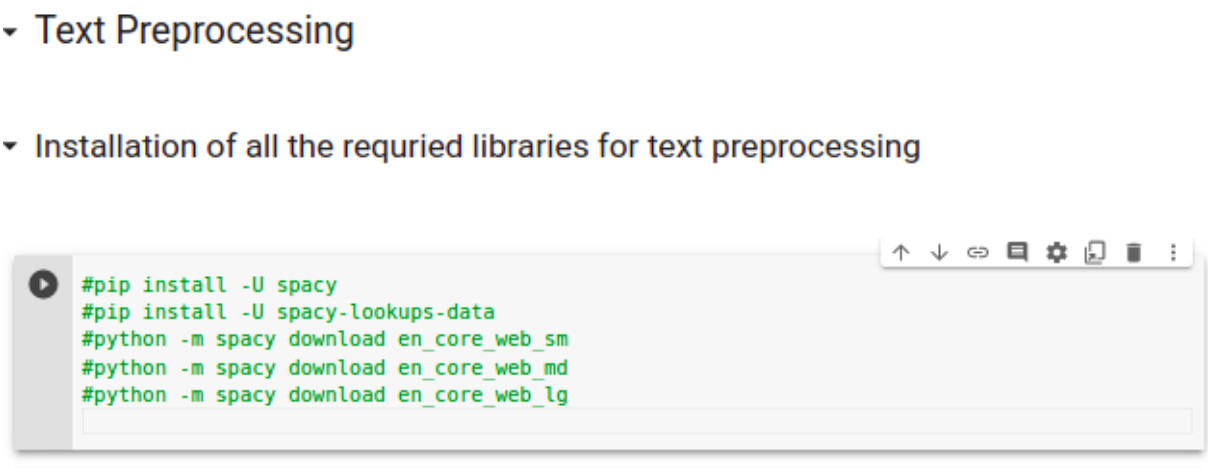
PyTorch[9] library was used to implement the Feed Forward Neural Network.

TensorFlow[10] library was used to carry out the visualization of the word embeddings . Using tensorflow the metadata and vector tsv files were created. These files were further imported into the TensorFlow Embedding Projector [13]which displays the embeddings in 2D plane using Principal Component Analysis.

The Sklearn[11] library was used to implement Naive Bayes classifier and Support Vector Machine.

Matplotlib[12] was used to plot graphs for the Word Embedding LSTM.

d. Screenshot and Demo along with Visualization:




```
import pandas as pd

import numpy as np


import spacy

from spacy.lang.en.stop_words import STOP_WORDS
```

Data set:

KazAnova. (2017 September) . Sentiment140 dataset with 1.6 million tweets, Version 2. Retrieved 9 May 2021 from https://www.kaggle.com/kazanova/sentiment140 .

```
from google.colab import drive

drive.mount('/content/drive')

direc = "/content/drive/My Drive/datasets/twitterData/"

df = pd.read_csv(direc+"training.1600000.processed.noemoticon.csv",encoding='latin-1',header=None)
```

```
df.head() #shows the first five rows of the dataframe
```

	0	1	2	3	4	5
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....

```
[ ] df = df[[5,0]] # columns 5 and 0 stored in df
```

```
[ ] #rename the columns
    df.columns = ['tweets','sentiment_label']
    df.head()
```

	tweets	sentiment_label
0	@switchfoot http://twitpic.com/2y1zl - Awww, t...	0
1	is upset that he can't update his Facebook by ...	0
2	@Kenichan I dived many times for the ball. Man...	0
3	my whole body feels itchy and like its on fire	0
4	@nationwideclass no, it's not behaving at all....	0

```
[ ] df['sentiment_label'].value_counts()

    # 0:

    4    800000
    0    800000
    Name: sentiment_label, dtype: int64
```

labels :

- 0 : positive
- 4 : negative

```
[ ] #lowercasing all the words in the tweet
    df['tweets']=df['tweets'].apply(lambda x: x.lower())
    df.head()
```

	tweets	sentiment_label
0	@switchfoot http://twitpic.com/2y1zl - awww, t...	0
1	is upset that he can't update his facebook by ...	0
2	@kenichan i dived many times for the ball. man...	0
3	my whole body feels itchy and like its on fire	0
4	@nationwideclass no, it's not behaving at all....	0

```
[ ] #contraction to expansion :
    #converting the words in their contracted form to their extracted form eg. he'll to he will
    #using the cont_to_exp() and a dictionary:{key: contractions,value:expansion}
    contractions = {
    "ain't": "am not",
    "aren't": "are not",
    "can't": "cannot",
    "can't've": "cannot have",
    "'cause": "because",
    "could've": "could have",
    "couldn't": "could not",
    "couldn't've": "could not have",
    "didn't": "did not",
    "doesn't": "does not",
    "don't": "do not",
    "hadn't": "had not",
    "hadn't've": "had not have",
    "hasn't": "has not",
    "haven't": "have not",
    "'ve": " have",
    "wasn't": "was not",
    " u ": " you ",
    " ur ": " your ",
    " n ": " and "}
    def cont_to_exp(x):
        if type(x) is str:
            for key in contractions:
                value = contractions[key]
                x = x.replace(key,value)
            return x
        else :
            return x
    df['tweets'] = df['tweets'].apply(lambda x:cont_to_exp(x))
```

```
[ ] df.head()
```

	tweets	sentiment_label
0	@switchfoot http://twitpic.com/2y1zl - awww, t...	0
1	is upset that he cannot update his facebook by...	0
2	@kenichan i dived many times for the ball. man...	0
3	my whole body feels itchy and like its on fire	0
4	@nationwideclass no, it is not behaving at all...	0

```
#removing all the emails in the tweets

import re

df['tweets']=df['tweets'].apply(lambda
x:re.sub(r'[a-zA-Z0-9._-]+@[a-zA-Z0-9._-]+\.[a-zA-Z0-9_-]+','',x))
```

```
df.head()
```

	tweets	sentiment_label
0	@switchfoot http://twitpic.com/2y1zl - awww, t...	0
1	is upset that he cannot update his facebook by...	0
2	@kenichan i dived many times for the ball. man...	0
3	my whole body feels itchy and like its on fire	0
4	@nationwideclass no, it is not behaving at all...	0

```
[ ] # Removing the urls from the tweets
df['tweets']=df['tweets'].apply(lambda x: re.sub(r'(http|ftp|https)://([\w_-]+(?:(?:\.[\w_-
df.head()
```

	tweets	sentiment_label
0	@switchfoot - awww, that is a bummer. you sh...	0
1	is upset that he cannot update his facebook by...	0
2	@kenichan i dived many times for the ball. man...	0
3	my whole body feels itchy and like its on fire	0
4	@nationwideclass no, it is not behaving at all...	0

Removal of RT:

RT is a keyword present in the tweets used to retweet a tweet to spread it across the network

```
[ ] df['tweets']=df['tweets'].apply(lambda x: re.sub('RT','',x))
    df.head()
```

	tweets	sentiment_label
0	@switchfoot - awww, that is a bummer. you sh...	0
1	is upset that he cannot update his facebook by...	0
2	@kenichan i dived many times for the ball. man...	0
3	my whole body feels itchy and like its on fire	0
4	@nationwideclass no, it is not behaving at all...	0

```
[ ] # Removal of special characters from the tweets
    df['tweets']=df['tweets'].apply(lambda x:re.sub(r'[^0-9a-zA-Z *]','',x))
```

These special characters are removed for this part but for future research the mentions will be used to construct edge relations between the nodes.

```
[ ] df.head()
```

	tweets	sentiment_label
0	switchfoot awww that is a bummer you should...	0
1	is upset that he cannot update his facebook by...	0
2	kenichan i dived many times for the ball manag...	0
3	my whole body feels itchy and like its on fire	0
4	nationwideclass no it is not behaving at all i...	0

```
[ ] #Removal of mulitple spaces between the words in the tweet
    df["tweets"]=df["tweets"].apply(lambda x: " ".join(x.split()))
    df.head()
```

	tweets	sentiment_label
0	switchfoot awww that is a bummer you shoulda g...	0
1	is upset that he cannot update his facebook by...	0
2	kenichan i dived many times for the ball manag...	0
3	my whole body feels itchy and like its on fire	0
4	nationwideclass no it is not behaving at all i...	0

```
[ ] #Removal of HTML Tags: from the tweets
    from bs4 import BeautifulSoup
    df['tweets'] = df['tweets'].apply(lambda x:BeautifulSoup(x,'lxml').get_text())
```

```
[ ] df.head()
```

	tweets	sentiment_label
0	switchfoot awww that is a bummer you shoulda g...	0
1	is upset that he cannot update his facebook by...	0
2	kenichan i dived many times for the ball manag...	0
3	my whole body feels itchy and like its on fire	0
4	nationwideclass no it is not behaving at all i...	0

Stopword Removal

Stopwords are the words that appear quite frequently in a sentence and do not have a significant contribution to the meaning of the sentence. Therefore they can be removed.

+ Code + Text

```
import spacy

df['tweets'] = df['tweets'].apply(lambda x:" ".join([t for t in x.split() if t not in STOP_WORDS ]))
```

Word Cloud visualization

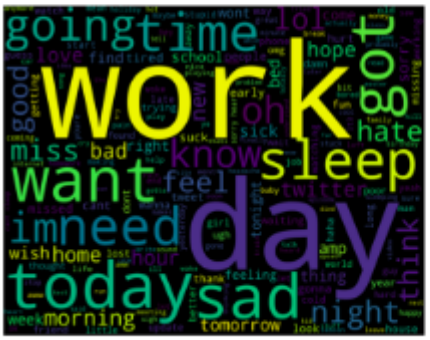
```
[ ] !pip install wordcloud
```

```
Requirement already satisfied: wordcloud in /usr/local/lib/python3.7/dist-packages (1.5.0)
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: pillow in /usr/local/lib/python3.7/dist-packages (from wordcl
```

```
from  wordcloud import WordCloud
import matplotlib.pyplot as plt
%matplotlib inline


df_positive = df[df['sentiment_label'] == 4]
bag_of_words_positive =' '.join(df_positive['tweets'])
bag_of_words_positive = bag_of_words_positive.split()


df_negative = df[df['sentiment_label'] == 0]
bag_of_words_negative =' '.join(df_negative['tweets'])
bag_of_words_negative = bag_of_words_negative.split()
x = ' '.join(bag_of_words_positive[:20000])
len(bag_of_words_positive)
print(x)


y = ' '.join(bag_of_words_negative[:20000])
len(bag_of_words_negative)
print(y)
```

```
#for positive:
wc = WordCloud(width=1800,height=1400).generate(x)
plt.imshow(wc)
plt.axis("off")
plt.show()
```


```
[ ] #for negative:
    wc = WordCloud(width=1800,height=1400).generate(y)
    plt.imshow(wc)
    plt.axis("off")
    plt.show()
```



Bag Of Words

```
[ ] print(df.shape)
    print(df.columns)
```

```
(1600000, 2)
Index(['tweets', 'sentiment_label'], dtype='object')
```

```
[ ] #sampling data:
    df_class0 = df[df['sentiment_label']==0].sample(2000)
    df_class4 = df[df['sentiment_label']==4].sample(2000)
```

```
[ ] df_sampled=df_class0.append(df_class4)
```

```
[ ] df_sampled.shape
```

```
(4000, 2)
```

```
[ ] df_sampled.head()
```

	tweets	sentiment_label
96419	need ride school tomorrow	0
523381	angry grr	0
585485	lonely wants followers follow peopleif hunt ur...	0
469355	experiencing cold symptoms away darn headache ...	0

```
[ ] #labels:
    y=df_sampled['sentiment_label']
    y.shape
```

```
(4000,)
```

```
[ ] from sklearn.feature_extraction.text import CountVectorizer
    cv = CountVectorizer()

    #features
    X = cv.fit_transform(df_sampled['tweets'])
    X.toarray().shape
```

```
(4000, 9754)
```

```
[ ] df_bag_of_words = pd.DataFrame(X.toarray(),
                                   columns=cv.get_feature_names())
```

```
[ ] df_bag_of_words.head()
```

	020394	040407	050	09	0mie	10	100	1000	1000m	101	1013	1030	10411	105	10am	10clc
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

5 rows × 9754 columns

Machine Learning models

1. NB
2. SVM
3. Metrics:accuracy,precision,recall

```
[ ] from sklearn.linear_model import SGDClassifier
    from sklearn.svm import LinearSVC
    from sklearn.naive_bayes import MultinomialNB
```

```
[ ] svm = LinearSVC(random_state=42,max_iter=200)
    nb = MultinomialNB()
```

```
[ ] clf = {'SVM':svm,'NB':nb}
```

```
[ ] clf.keys()
```

```
    dict_keys(['SVM', 'NB'])
```

```
[ ] for key in clf.keys():
      print(clf[key])
```

```
    LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
              intercept_scaling=1, loss='squared_hinge', max_iter=200,
              multi_class='ovr', penalty='l2', random_state=42, tol=0.0001,
              verbose=0)
    MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

```
from sklearn.preprocessing import MinMaxScaler

from sklearn.model_selection import train_test_split

from sklearn.metrics import confusion_matrix,accuracy_score

#used to center the data

def classify(X,y):

 scaler = MinMaxScaler(feature_range=(0,1))

 X = scaler.fit_transform(X)


          X_train,          X_test,          y_train,y_test          =
train_test_split(X,y,test_size=0.2,random_state = 42)


 for key in clf.keys():

   clf[key].fit(X_train,y_train)

   y_pred = clf[key].predict(X_test)
```

```
    accuracy = accuracy_score(y_test,y_pred)

    conf_matrix = confusion_matrix(y_test,y_pred)

    print(key,":accuracy : ",accuracy*100)

    print("\n")

    print(key,":confusion_matrix:\n",conf_matrix)


print("precision:",conf_matrix[0][0]*100/(conf_matrix[0][0]+conf_matrix
[1][0]))


print("recall:",conf_matrix[0][0]*100/(conf_matrix[0][0]+conf_matrix[0]
[1]))
```

```
[ ] classify(df_bag_of_words,y)
```

```
SVM :accuracy :  70.75


SVM :confusion_matrix:
 [[301 121]
 [113 265]]
precision: 72.70531400966183
recall: 71.32701421800948
NB :accuracy :  69.25


NB :confusion_matrix:
 [[300 122]
 [124 254]]
precision: 70.75471698113208
recall: 71.09004739336493
```

precision

precision = tp/(tp+fp)
recall = tp/(tp+fn)

3 TensorFlow

- Word Embeddings

Word Embeddings

Example in TensorFlow using movies review dataset in keras dataset repositories.

representation of the words

1. Bag of words , unique words using all words. one -hot encoding using the dictionary that you created. sentence as a vector with 1 and 0 for occurence and non occurence of the term in the sentence. Length of the encoding is same as the length of the dictionary.
2. Integer encoding. assign number to each unique word. for the sentence, the words will be represented according to the number

Cons:

- cannot portray the relationships between the words
- reason: basis in higher dimensional space, vectors are orthogonal, dot product is 0. No projection on any axis therefore they cannot show the relationships.
- the vectors are too sparse.
  Solution:

3. Word Embedding New space, where the words are transformed to , it is a hyperparameter of the model like the number of hidden layers in a neural network.

dot products can be taken and the relations between words can be represented.

strong correlation of words. model takes words,puts through the embedding layers., good or bad review,matches the training label.and then backpropogates through the model and changes parameters.

model now can predict positive negative and can also show a correlation between words

```
import io

import matplotlib.pyplot as plt

import tensorflow as tf


from tensorflow import keras

from tensorflow.keras import layers

import tensorflow_datasets as tfds


embedding_layer = layers.Embedding(1000,5)
```

```
result = embedding_layer(tf.constant([1,2,3]))
```

```
[ ] print(result.numpy())
    print(result.numpy().shape)
```

```
[[ 0.03767145  0.0077535  -0.01841074 -0.04676837 -0.00963044]
 [ 0.0321715  -0.01822264 -0.02493215 -0.00038576  0.04014523]
 [-0.01095762  0.03429672  0.01967445  0.03140387 -0.00852419]]
(3, 5)
```

5 columns since we gave (1000, 5) as input in the Embedding layer

3 rows since we gave tf.constant([1,2,3])

```
[ ] #loading dataset
    (train_data,test_data),info = tfds.load('imdb_reviews/subwords8k',
                                            split=(tfds.Split.TRAIN,tfds.Split.TEST),
                                            with_info=True,as_supervised=True)
    encoder = info.features['text'].encoder
    print(encoder.subwords[:20])
```

```
WARNING:absl:TFDS datasets with text encoding are deprecated and will be removed in a future
Downloading and preparing dataset imdb_reviews/subwords8k/1.0.0 (download: 80.23 MiB, genera
Dl Completed...: 100% 1/1 [00:06<00:00, 6.42s/ url]
Dl Size...: 100% 80/80 [00:06<00:00, 12.55 MiB/s]
```

```
def get_batch_data():

 #loading dataset

 (train_data,test_data),info = tfds.load('imdb_reviews/subwords8k',


split=(tfds.Split.TRAIN,tfds.Split.TEST),


with_info=True,as_supervised=True)

 encoder = info.features['text'].encoder

 print(encoder.subwords[:20])

 #lengths of all the reviews are different: we append zeros at the end
using padded shapes

 padded_shapes = ([None],())

 train_batches = train_data.shuffle(1000).padded_batch(10,


padded_shapes=padded_shapes)

 test_batches = test_data.shuffle(1000).padded_batch(10,
```

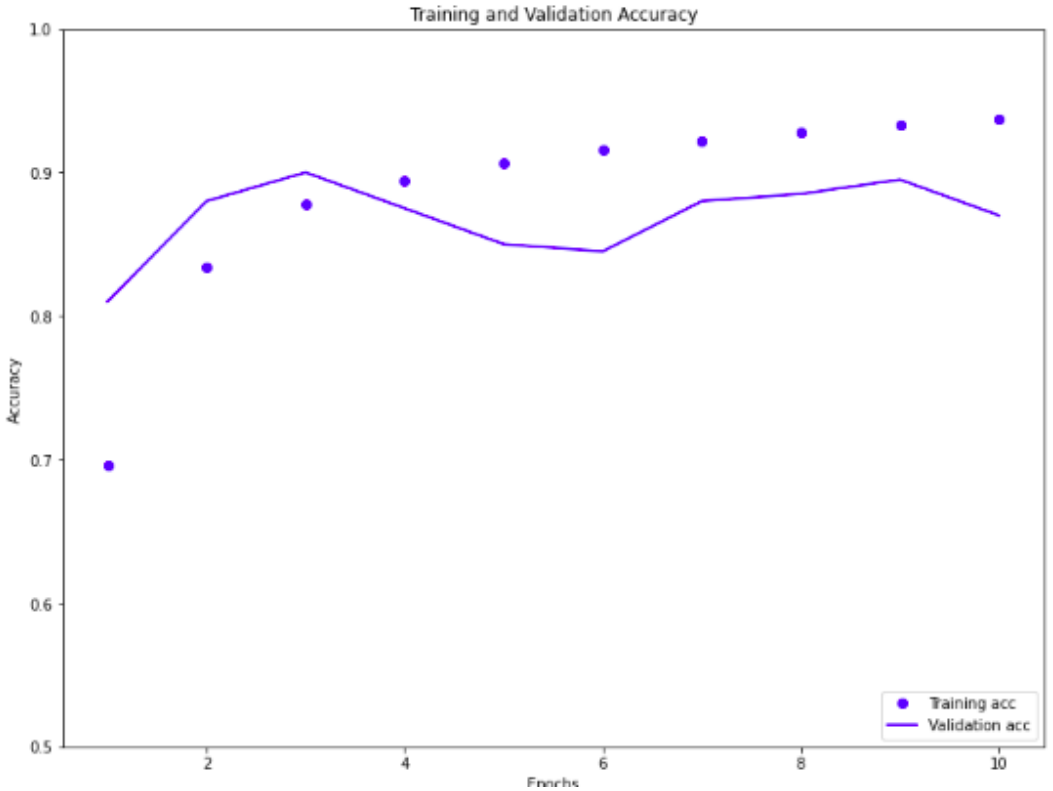
```
padded_shapes=padded_shapes)
 return train_batches,test_batches,encoder
def get_model(encoder,embedding_dim=16):
 #defining a model:
 #number of dimensions for our embedding layer:
 # embedding_dim = 16
 model = keras.Sequential([

layers.Embedding(encoder.vocab_size,embedding_dim),
                            layers.GlobalAveragePooling1D(),
                                  layers.Dense(1,activation='sigmoid') #
probablity that the review is positive
 ])
 model.compile(optimizer='adam',loss='binary_crossentropy',
               metrics=['accuracy']
 )
 return model
def plot_history(history):
 #convert the hsitory to a dictionary
 history_dict = history.history
 #accuracy :
 acc = history_dict['accuracy']
 validation_acc= history_dict['val_accuracy']
 epochs= range(1,len(acc)+1)
 plt.figure(figsize=(12,9))
 plt.plot(epochs,acc,'bo',label='Training acc')
 plt.plot(epochs,validation_acc,'b',label='Validation acc')
```

```
 plt.title('Training and Validation Accuracy')
 plt.xlabel('Epochs')
 plt.ylabel('Accuracy')
 plt.legend(loc='lower right')
 plt.ylim((0.5,1))
 plt.show()
def retrieve_embeddings(model,encoder):
 out_vectors = io.open('vectors.tsv','w',encoding='utf-8')
 out_metadata = io.open('metadata.tsv','w',encoding='utf-8')
 weights = model.layers[0].get_weights()[0] #weights of 0th layer
 for num,word in enumerate(encoder.subwords):
   vec = weights[num+1]
   out_metadata.write(word+'\n')
   out_vectors.write('\t'.join([str(x) for x in vec])+'\n')
 out_vectors.close()
 out_metadata.close()


 try:
   from google.colab import files
   files.download('vectors.tsv')
   files.download('metadata.tsv')
 except Exception:
   pass
```

```
train_batches,test_batches,encoder= get_batch_data()
model=get_model(encoder)
history = model.fit(train_batches,epochs=10,validation_data=test_batches,validation_steps=20)
```

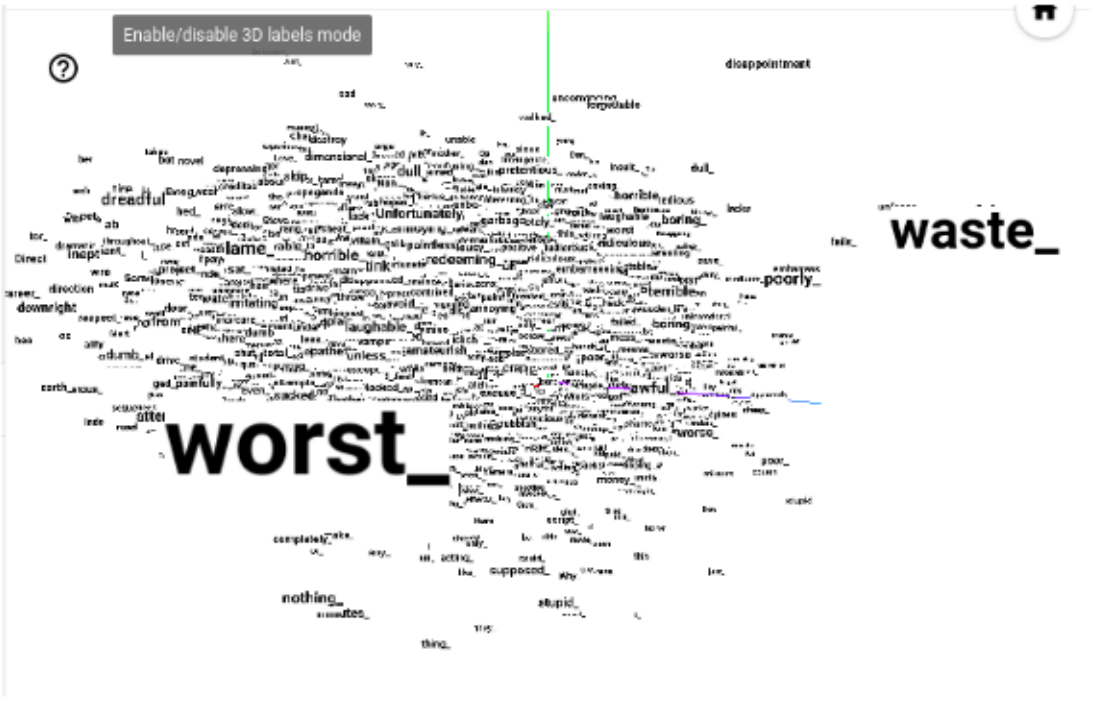
```
WARNING:absl:TFDS datasets with text encoding are deprecated and will be removed in a future version. Instead, you should use the plain text versi
['the_', ', ', '. ', 'a_', 'and_', 'of_', 'to_', 's_', 'is_', 'br', 'in_', 'I_', 'that_', 'this_', 'it_', ' /><', ' />', 'was_', 'The_', 'as_']
Epoch 1/10
2500/2500 [==============================] - 13s 5ms/step - loss: 0.6395 - accuracy: 0.6957 - val_loss: 0.5444 - val_accuracy: 0.8100
Epoch 2/10
2500/2500 [==============================] - 14s 5ms/step - loss: 0.4688 - accuracy: 0.8341 - val_loss: 0.3937 - val_accuracy: 0.8800
Epoch 3/10
2500/2500 [==============================] - 14s 6ms/step - loss: 0.3660 - accuracy: 0.8776 - val_loss: 0.3120 - val_accuracy: 0.9000
Epoch 4/10
2500/2500 [==============================] - 12s 5ms/step - loss: 0.3097 - accuracy: 0.8946 - val_loss: 0.3548 - val_accuracy: 0.8750
Epoch 5/10
2500/2500 [==============================] - 11s 4ms/step - loss: 0.2751 - accuracy: 0.9063 - val_loss: 0.3556 - val_accuracy: 0.8500
Epoch 6/10
2500/2500 [==============================] - 11s 4ms/step - loss: 0.2490 - accuracy: 0.9159 - val_loss: 0.4147 - val_accuracy: 0.8450
Epoch 7/10
2500/2500 [==============================] - 11s 4ms/step - loss: 0.2309 - accuracy: 0.9216 - val_loss: 0.3630 - val_accuracy: 0.8800
Epoch 8/10
2500/2500 [==============================] - 11s 4ms/step - loss: 0.2173 - accuracy: 0.9285 - val_loss: 0.3219 - val_accuracy: 0.8850
Epoch 9/10
2500/2500 [==============================] - 12s 5ms/step - loss: 0.2020 - accuracy: 0.9332 - val_loss: 0.2750 - val_accuracy: 0.8950
Epoch 10/10
2500/2500 [==============================] - 11s 4ms/step - loss: 0.1891 - accuracy: 0.9376 - val_loss: 0.3468 - val_accuracy: 0.8700
```

```
[ ]  plot_history(history)
     retrieve_embeddings(model,encoder)
```



- to visualize the embeddings:

  1. http://projector.tensorflow.org/
  2. upload vectors.tsv, metadata.tsv



in the above PCA of the word embeddings:

1. from this angle , waste, worst, poorly can be seen which means that they are related

4. Bag or Words Feed forward - PyTorch

- Dataset and Loading
- Preprocessing - Word Embedding,Bag Of Words
- Model:
    - definition
    - loss function
    - learning rate
    - optimizer
    - automatic differenciation : auto_grad
- Evaluation and Metrics

PyTorch :

1. The computational graph is defined on the flow unlike tensorflow where first the computatinal graph is defined then the data is fed into it.
2. Helps to design complex networks

Models:

- Bag of words classifier
- GRU

Bag Of Words Classifier for text classifier using PyTorch

```
[ ] import pandas as pd
    import torch
    import torch.nn as nn
    import torch.nn.functional as F
    import torch.optim as optim
    from torch.utils.data import DataLoader, Dataset
    from sklearn.feature_extraction.text import CountVectorizer
    from tqdm import tqdm, tqdm_notebook
```

loading data from google drive:

```
# from google.colab import drive

# drive.mount('/content/drive')

# direc = "/content/drive/My Drive/datasets/twitterData/"
```

```
# df = pd.read_csv(direc+"training.1600000.processed.noemoticon.csv",encoding='latin-1',header=None)
```

```
[ ] df.sample(n = 6)
```

	tweets	sentiment_label
1174611	going spend today pool better laying sun good ...	4
38588	says plurvana hangs precarious balance	0
1203015	salvagua lauralaing sandrasrockinit paocamargo...	4
1537781	ok finally tired boos updates eghhhhhh gotta l...	4
1368982	juanmontoya cool dreams	4
781315	lovetheme arrrghhh im feeling dizzy stayed hom...	0

downsample the dataset to 1000 tweets for the training purpose

```
[ ] # df =df[[0,5]]
    # df.columns = ['tweets','sentiment_label']
    # df.head()
```

```
[ ] #create train test split:

    df_downsampled_bow = df.sample(10000)

    #test labels :
    test_labels = df_downsampled_bow['sentiment_label']
```

```
[ ] df_downsampled_bow
```

```
[ ] df_downsampled_bow
```

	tweets	sentiment_label
1367356	237 singing sesh spotify favourite kind singin...	4
1281560	bensue thank youu lovely daay ima saying day s...	4
498069	o im srry didnt know	0
108615	chrisdjmoyles miss djing week mornings hurry h...	0
1319674	ouhyeaaaaaahthis day school nightmare school y...	4
...	...	...
721300	kaitlynray sorry	0
587220	having absolute mare trying sleep	0
1059605	budsthrubullies hi hope things going south	4
1018195	heard dublin sold right omw	4
346560	sadly leaving business trip couple days missin...	0

1000 rows × 2 columns

```
[ ] test_labels
```

```
1367356    4
1281560    4
498069     0
108615     0
1319674    4
           ..
721300     0
587220     0
1059605    4
1018195    4
346560     0
Name: sentiment_label, Length: 1000, dtype: int64
```

Bag of words representation

dictionary: {0:"recurrent",1:"neural",2:"network",3:"artifical",4:"intelligence"}

example sentence: "artificial neural network and the recurrent neural network"

we dont remove the stopwords here since Recurrent neural networks are based on sequencial data so here the sequence of all the words matter even the stopwords.

	artificial	neural	network	and	the	recurrent	neural	network	final vector
recurrent	0	0	0	0	0	1	0	0	1
neural	0	1	0	0	0	0	1	0	2
network	0	0	1	0	0	0	0	1	2
artificial	1	0	0	0	0		0	0	1
intelligence	0	0	0	0	0	0	0	0	0

final vector: {1,2,2,1,0}

the final vector is the one hot encoding of the sentence with respect to the vocabulary which is a dictionary containing a bag of words in the whole corpus.

- creating the BagOfWords Classifier * Model:feed forward neural network
- layer 1: x1 = W1 * X + b1
- layer 1(activation): h1 = Relu(x1)
- layer 2: x2 = W2 * h1 + b2log()
- output : p=sigma(x2)
- loss : -(ylog(p)+(1-y)log(1-p))
- gradient : d L(W1,b1,W2,b2)/ dW1 = (dL/dp) * (dp/dx2) * (dx2/dh1) * (dh1/dx1) * (dx1/dW1)

- optimizer : Parameter update :
- W1 =W1 - alpha (dL/dW1)

Preprocess the text:

Preprocess the text:

```
class Sequences(Dataset):

 def __init__(self,df):
```

```
   df = df


                                                self.vectorizer                =
CountVectorizer(stop_words='english',max_df=0.99,min_df=0.005)

      #the words appear in the messages : here the message are the
documents.

   # the number of documents(messages) a word in the document(message)
appears is the document frequency of the word

   # in the corpus

    # to ignore the words appearing too many times across documents:
max_df = 0.99

   # to ignore the words appearing rarely in across docuemtns: min_df =
0.005



                                              self.sequences                =
self.vectorizer.fit_transform(df['tweets'].tolist())

   self.labels = df['sentiment_label'].tolist()

   self.token2idx= self.vectorizer.vocabulary_

          self.idx2token    =    {idx:    token    for    token,    idx    in
self.token2idx.items()}

 def __getitem__(self,i):

   return self.sequences[i,:].toarray(),self.labels[i]

 def __len__(self):

   return self.sequences.shape[0]
```

```
[ ] dataset = Sequences(df_downsampled_bow)
    len(dataset)
```

```
10000
```

```
[ ] dataset = Sequences(df_downsampled_bow)
    train_loader = DataLoader(dataset,batch_size=300) # loads 300 rows at a time
     # now our pandas dataset train sample is converted into a tensor
                                # dataset is the tensor
    # dataset_test = Sequences(test_labels)
    # test_loader = DataLoader(dataset_test,batch_size=300)
    print(dataset[5][0].shape)
```

```
(1, 187)
```

```
[ ] class BagOfWordsClassifier(nn.Module):
      def __init__(self,vocab_size,hidden1,hidden2):
        super(BagOfWordsClassifier,self).__init__()
        self.fc1 = nn.Linear(vocab_size,hidden1)
        self.fc2 = nn.Linear(hidden1,hidden2)
        self.fc3 = nn.Linear(hidden2,1)
      def forward(self,inputs):
        x = F.relu(self.fc1(inputs.squeeze(1).float()))
        x = F.relu(self.fc2(x))
        return self.fc3(x)
```

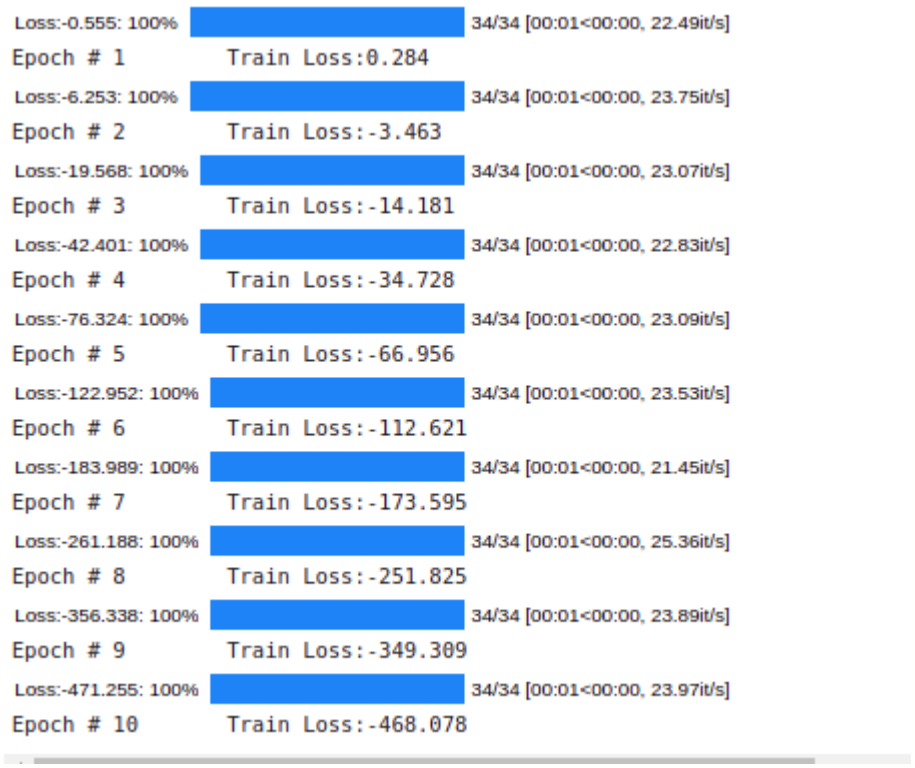
```
[ ] model = BagOfWordsClassifier(len(dataset.token2idx),128,64)
    model
```

```
BagOfWordsClassifier(
  (fc1): Linear(in_features=187, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=64, bias=True)
  (fc3): Linear(in_features=64, out_features=1, bias=True)
)
```

```
[ ] #loss function:
    criterion = nn.BCEWithLogitsLoss()
    #optimer declaration
    optimizer = optim.Adam([p for p in model.parameters() if p.requires_grad],lr=0.001)
```

```
#training loop:

model.train()

train_losses = []

for epoch in range(10):

 progress_bar = tqdm_notebook(train_loader,leave=False)

 losses = []

 total = 0

 for inputs, target in progress_bar:

   model.zero_grad()
```

```
    #forward pass
    output = model(inputs)
    #loss calculation
    loss = criterion(output.squeeze(),target.float())

    #backward pass:
    loss.backward()
    nn.utils.clip_grad_norm_(model.parameters(),3)

    #optimize at the end of the backward pass
    optimizer.step()

    progress_bar.set_description(f'Loss:{loss.item():.3f}')

    losses.append(loss.item())
    total +=1

epoch_loss = sum(losses)/total
train_losses.append(epoch_loss)

tqdm.write(f'Epoch # {epoch + 1}\t Train Loss:{epoch_loss:.3f}')
```



6. Results and discussion along with Visualization

SR.No.	Name of Model	Accuracy	Precision	Recall
1	SVM	70.75	72.7053	71.3270
2	NB	69.25	70.7547	71.0900

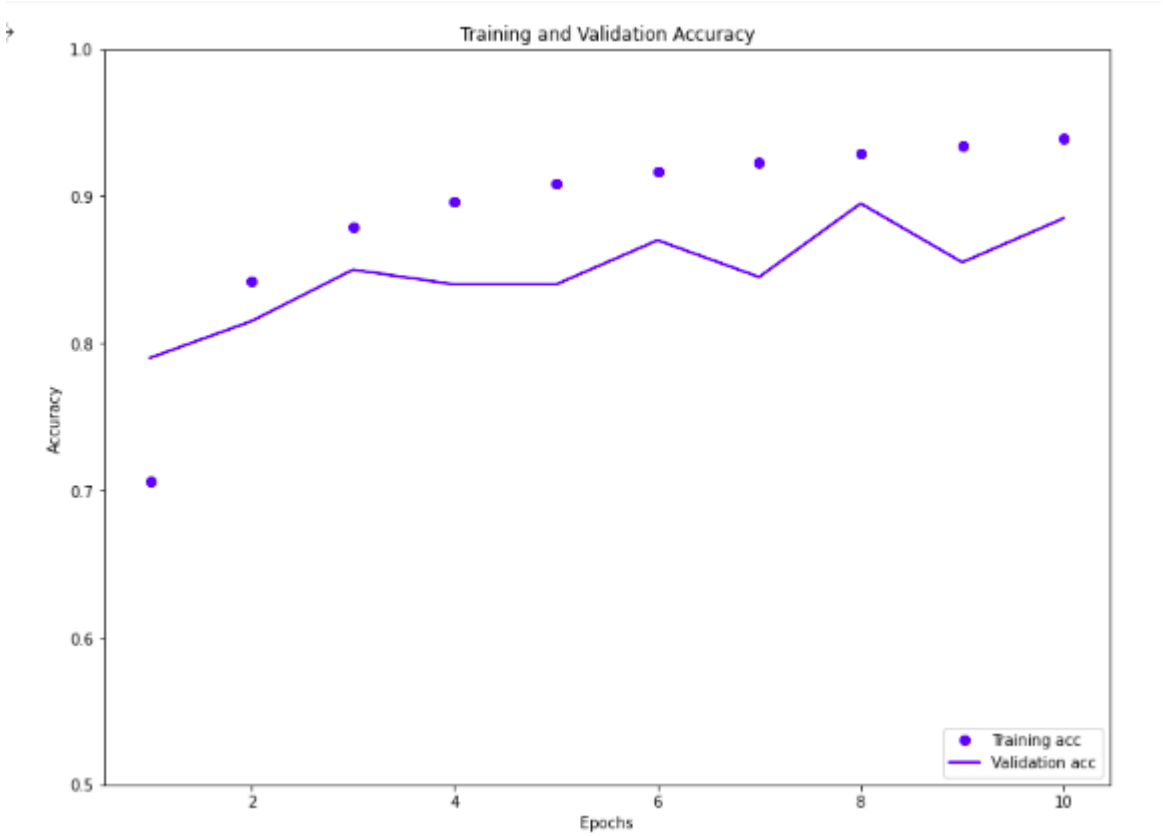
3. Feed Forward Neural Network

```
Loss:-0.555: 100% 34/34 [00:01<00:00, 22.49it/s]
Epoch # 1     Train Loss:0.284
Loss:-6.253: 100% 34/34 [00:01<00:00, 23.75it/s]
Epoch # 2     Train Loss:-3.463
Loss:-19.568: 100% 34/34 [00:01<00:00, 23.07it/s]
Epoch # 3     Train Loss:-14.181
Loss:-42.401: 100% 34/34 [00:01<00:00, 22.83it/s]
Epoch # 4     Train Loss:-34.728
Loss:-76.324: 100% 34/34 [00:01<00:00, 23.09it/s]
Epoch # 5     Train Loss:-66.956
Loss:-122.952: 100% 34/34 [00:01<00:00, 23.53it/s]
Epoch # 6     Train Loss:-112.621
Loss:-183.989: 100% 34/34 [00:01<00:00, 21.45it/s]
Epoch # 7     Train Loss:-173.595
Loss:-261.188: 100% 34/34 [00:01<00:00, 25.36it/s]
Epoch # 8     Train Loss:-251.825
Loss:-356.338: 100% 34/34 [00:01<00:00, 23.89it/s]
Epoch # 9     Train Loss:-349.309
Loss:-471.255: 100% 34/34 [00:01<00:00, 23.97it/s]
Epoch # 10    Train Loss:-468.078
```

4. Word Embedding visualization using LSTM.

```
loss: 0.6319 - accuracy: 0.7064 - val_loss: 0.5565 - val_accuracy: 0.7900
loss: 0.4618 - accuracy: 0.8422 - val_loss: 0.4461 - val_accuracy: 0.8150
loss: 0.3593 - accuracy: 0.8788 - val_loss: 0.4458 - val_accuracy: 0.8500
loss: 0.3082 - accuracy: 0.8962 - val_loss: 0.3950 - val_accuracy: 0.8400
loss: 0.2723 - accuracy: 0.9084 - val_loss: 0.3899 - val_accuracy: 0.8400
loss: 0.2491 - accuracy: 0.9166 - val_loss: 0.4054 - val_accuracy: 0.8700
loss: 0.2289 - accuracy: 0.9230 - val_loss: 0.4230 - val_accuracy: 0.8450
loss: 0.2142 - accuracy: 0.9288 - val_loss: 0.3558 - val_accuracy: 0.8950
loss: 0.1981 - accuracy: 0.9345 - val_loss: 0.3880 - val_accuracy: 0.8550
loss: 0.1897 - accuracy: 0.9390 - val_loss: 0.3364 - val_accuracy: 0.8850
```

Training and Validation Accuracy V/S Epochs



To Optimize the performance of LSTM , GRU can be used.

GRU: Gating Recurrent Units

Recurrent neural networks feed the output to itself , and can be done using a loop.
The type of rnn to be used:many to one

Traditional rnn has the vanishing gradient problem: the words appearing in the beginning start to lose their meaning in the final computation as more words are passed into the recurrent neural network.

RNN architecture used : GRU (Gated Recurrent Unit) uses update and reset gates to decide what is important to keep in the sentence uses different activation functions

1. inputs: takes in current state(xt) and the previous state inputs (h(t-1))
2. reset gate: rt = sigma(Wr * xt + Ur * h(t-1) +br
   br = reset bias
   trainable parameters: Wr, br

3. update gate: zt = sigma(Wz * xt + Uz * h(t-1) +bz)
   trainable parameters: Wz, bz
   bz = update bias
4. hidden state : ht = (1-zt) o h(t-1) + zt o (Wh*xt + Uh(rt o h(t-1))+ bh)

trainable parameters: Wh, bh

Future Research

1. Gated Recurrent Units to train a recurrent neural network to detect email spam.
2. LSTM : Long Short Term Memory to train a recurrent neural network to detect email spam.
3. Tensorboard Visualizations for the Pytorch Plots of training and testing using TensorboardX library.
4. **Graph Neural Networks**

   1.Node Embeddings

   Given a graph G(V,E).

   Each node v on the graph can be mapped to a d-dimensional embedding or representation . It has all node features, edge features connecting to the node. The features of nodes connecting to node v are proportional to the importance of each neighborhood node.

   Our goal is to keep the nodes similar to n closer to each other in the embedding space.

   2.Graph Convolutional Networks

   Approach: To average the neighbor messages and apply a neural network.

   Initially the 0th layer embeddings are equal to node features.

   Message Passing: To calculate the new embedding of the node after k layers of aggregation. Aggregation involves the nearby nodes to share their attributes to the node for which the embedding is calculated.

Training the model: A loss function is defined on the embeddings. We are trying to learn the contribution of neighbors and our own attributes. Embeddings should be fed into a loss function and stochastic gradient descent is used to train the weight parameters.

Ideology:

Graph Classification

Done by constructing a graph with the terms in a tweet from a user.

Node Classification

Done by considering the node attribute as the sentiment of the tweet.

Node Classification the Graph Neural Network will take in the nodes of the network graph that are the tweets and transform it to another embedding space. This embedding space has the nodes in a vector fashion with the relation between the nodes can be found using cosine similarity or dot product of vectors.

The tweet vectors in the embedding space are processed in the Graph Neural Network to classify them as positive or negative.

Graph Classification the Graph Neural Network will take in small graphs of each tweet word with the attributes : sentiment .

The tweets are classified using these graphs of the network.

7. References

[1]Martino, Francesco & Spoto, Andrea. (2006). Social Network Analysis: A brief theoretical review and further perspectives in the study of Information Technology. PsychNology Journal. 4. 53-86. https://www.researchgate.net/publication/220168913_Social_Network_Analysis_A_brief_theoretical_review_and_further_perspectives_in_the_study_of_Information_Technology

[2]Otte, Evelien & Rousseau, Ronald. (2002). Social Network Analysis: A Powerful Strategy, also for the Information Sciences. Journal of Information Science. 28. 441-453. 10.1177/016555150202800601.https://www.researchgate.net/publication/242401176_Social_Network_Analysis_A_Powerful_Strategy_also_for_the_Information_Sciences

[3]Grandjean, Martin. (2016). A social network analysis of Twitter: Mapping the digital humanities community. Cogent Arts & Humanities. 3. 1171458. 10.1080/23311983.2016.1171458. https://www.researchgate.net/publication/301330614_A_social_network_analysis_of_Twitter_Mapping_the_digital_humanities_community

[4]R. Abascal-Mena, R. Lema and F. Sèdes, "From tweet to graph: Social network analysis for semantic information extraction," 2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS), Marrakech, Morocco, 2014, pp. 1-10.

doi: 10.1109/RCIS.2014.6861047

URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6861047&isnumber=6860531

[5]P. M. Dudas, "Cooperative, dynamic Twitter parsing and visualization for dark network analysis," 2013 IEEE 2nd Network Science Workshop (NSW), West Point, NY, USA, 2013, pp. 172-176.

doi: 10.1109/NSW.2013.6609217

URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6609217&isnumber=6609181

[6]Tan Q, Liu N and Hu X (2019) Deep Representation Learning for Social Network Analysis. Front. Big Data 2:2. doi:

10.3389/fdata.2019.00002.https://www.frontiersin.org/articles/10.3389/fdata.2019.00002/full

[7]KazAnova. (2017 September) . Sentiment140 dataset with 1.6 million tweets, Version 2. Retrieved 9 May 2021 from https://www.kaggle.com/kazanova/sentiment140 .

[8] Spacy.https://spacy.io/
[9] PyTorch.https://pytorch.org/
[10] TensorFlow.https://www.tensorflow.org/
[11]sklearn.https://scikit-learn.org/stable/
[12]matplotlib.https://matplotlib.org/
[13]TensorFlow Embedding Projector .https://projector.tensorflow.org/