Towards fulfillment for Undergraduate Degree Level Programme
Bachelor of Technology in Computer Engineering

A Project Report on:

Sentiment Analysis of Social Media Data

Prepared by :

Admission No.	Student Name
U16CO010	Deval Pandya
U16CO014	Yash Patel
U16CO015	Jaykumar Tandel
U16CO023	Shreyas Patel

Class : B.TECH. IV (Computer Engineering) 8th Semester

Year : 2019-2020

Guided by : Dr. Rupa G. Mehta

DEPARTMENT OF COMPUTER ENGINEERING
SARDAR VALLABHBHAI NATIONAL INSTITUTE OF TECHNOLOGY,
SURAT - 395 007 (GUJARAT, INDIA)

Student Declaration

This is to certify that the work described in this project report has been actually carried out and implemented by our project team consisting of

Sr.	Admission No.	Student Name
1	U16CO010	Deval Pandya
2	U16CO014	Yash Patel
3	U16CO015	Jaykumar Tandel
4	U16CO023	Shreyas Patel

Neither the source code there in, nor the content of the project report have been copied or downloaded from any other source. We understand that our result grades would be revoked if later it is found to be so.

Signature of the Students:

Sr.	Student Name	Signature of the Student
1	Deval Pandya	
2	Yash Patel	
3	Jaykumar Tandel	
4	Shreyas Patel	

Certificate

This is to certify that the project report entitled Sentiment Analysis on Social Media` *is prepared and presented by*

Sr.	Admission No.	Student Name
1	U16CO010	Deval Pandya
2	U16CO014	Yash Patel
3	U16CO015	Jaykumar Tandel
4	U15CO023	Shreyas Patel

Final Year of Computer Engineering and their work is satisfactory.

SIGNATURE :

GUIDE JURY HEAD OF DEPT.

Abstract

In today's world, social media is a very important platform where people can share their views on anything they want to ranging from wars of past to some currently trending topic in the world. A number of companies and individuals use the social media to understand behavior of various users and then use that data to determine various aspects of information about the users. Using Sentiment Analysis, companies can establish how positively or negatively inclined that person is, depending on the contents they post on social media. This data can be used in many ways like advertising, consulting, recommendations, understanding thoughts of people on certain topics, etc. In this project, we are implementing sentiment analysis along with Neo4j as a database, in order to model the connections of social media into the database. Neo4j being a graph based database gives a considerable advantage over relational database, when it comes to storing and retrieving such situations where all the entities can be related to each other in some or the other way.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Sentiment Analysis (also known as Opinion Mining or Emotion AI) is the task of identifying positive and negative opinions, emotions, and evaluations using natural language processing, text analysis, computation semantics and biometrics. Written sentiment terms signifies either temperament of their author or point of view regarding a given object, since people are sharing their thoughts on social media often.

1.1 Application

Problem Statement

Sentiment Analysis of Social Media Text using graph based database.

1. Consulting people : By analyzing the thoughts shared by people on social media, sentiment values can be calculated. With the help of these values, the state of mind of people can be determined and they can be consulted.

2. Voice Of Customer : Sentiment analysis of social media feedback, references and studies assist to show the opinions of user to the firm. This way the firm recognizes, how people feel regarding facilities. The firm may utilize this data for ad purpose.

3. Contextual Semantic Search(CSS) : To achieve actionable perceptions, the company needs to know about which of its sides users are discussing. For example, Flipkart would want to segregate messages related to late deliveries, billing problems, raise related inquiries, product reviews, etc.
   CSS algorithm takes two inputs, millions of messages and a concept (like salary) then segregates all the messages that closely match with the given concept.

1.2 Motivation

In today's world almost half of the population suffers from anxiety, depression due to peer pressure, competition, serious illness, conflict or dispute within family members or friends, abuse. Folks share mostly on social media what they feel about certain topics. Sentiment analysis is exceptionally useful in social media observing as it allows us to achieve an outline of the wider public belief behind particular matters. Using sentiment analysis, state of mind of folks can be determined. If the sentiment value is negative then we can take further steps properly. We can consult them in better way.

1.3 Objectives

With the help of twitter API, data is acquired which is stored in neo4j graph database in order to do fast graph traversal and retrieval. Using sentiment analysis, sentiment values of tweets are calculated from which emotional status of the person can be projected.Furthermore, tweets are categorized in various topics, such as Education, Business, Politics, Entertainment and Sports such that mental health of the person can be analysed effectively.

1.4 Organization of Project Report

In Chapter 2, Literature survey is discussed. In Chapter 3, problem of sentiment analysis is described and different techniques of doing various sentiment analysis techniques are discussed. The importance of graph database, Neo4j is explained. Different type of word embedding is described thoroughly. Two used machine learning algorithms, such as SVM and LSTM are explicated in detail. In Chapter 3, an efficient model to do the sentiment analysis of twitter data along side implementing the data storage in Neo4j for faster tweet retrieval are proposed.

Chapter 2

Literature Survey

There has been a lot of research done in area of sentiment analysis. Walaa Medhat, Ahmed Hassan and Hoda Korashy [1] has done an extensive survey on all the existing sentiment analysis algorithms.

Alec Go, Richa Bhayani and Lei Huang[2] showed that among machine learning algorithms SVM, Naïve Bayes and Maximum Entropy, SVM works the best when handling twitter data. Moreover, they described that preprocessing steps are needed in order to achieve high accuracy.

Lija Mohan and Sudheep Elayidom[4] Concentrated on predicting who have greater chances to win the Delhi Legislative Elections held February 2015 in India using sentiment analysis. They used lexicon based approach to classify the tweets in negative or positive sentiments.

Fotis Aisopos, George Papadakis, Theodora Varvarigou[5] looked at the use of the n-gram graphs model in the Social Media Sentiment Analysis context. They explained n-gram graphs model over preliminary classification techniques like Naïve Bayes Model and C4.5. Dr. S. Vijayarani, Ms. J. Ilamathi, Ms. Nithya[7] did an ample survey on all existing text preprocessing techniques. They explained how to make large scale text data most useful for certain applications. They explained different techniques like stop words elimination and all type of stemming algorithms.

Tanasanee Phienthrakul, Boonserm Kijsirikul, Hiroya Takamura and Manabu Okumura[9] implemented text classification for sentiment analysis using SVM. They used svm with different types of linear and non linear kernels. Those kernels are used on consumer ratings to see if a rating is positive or negative.

Maha Heikal, Marwan Torki and Nagwa El-Makky [11] showed that deep-learning - based sentiment classification methods, in particular those based on either CNN or RNN, showed outstanding performance as well as significantly outperformed traditional lexicon-based or other machine-learning classification models of sentiments.

Dipti Mahajan, Dev Kumar Chaudhary[12] has shown how the stanford library can be used to enhance the machine's ability to identify more data from twitter using the Twitter API.

Subarno Pal, Soumadip Ghosh, Amitava Nag[13] employed Recurrent Neural Network for the Sentiment Analysis field of study. LSTM RNNs have been shown to be more accurate for sentiment analyzes than Deep Neural Networks and traditional RNNs. They utilized straightforward LSTM first. nd then, increases in precision are seen by raising the number of layers.

Waleed Zaghloul, Sang M. Lee, Silvana Trimi[14] evaluated the performance as text classifiers of NN and SVM. They concluded that using NN, we can get more accuracy even using the small dataset.

Jeffrey Pennington, Richard Socher, Christopher D. Manning[15] consider the GloVe to be a modern universal log-bilinear regression method for the unregulated training of word representations that outclasses existing frameworks in word comparison, similarity, entity recognition function.

Stephen Jini, P. Prabu [16] has suggested an important method for detecting depression levels in people on twitter. Calculated sentiment scores can be equipped with different emotions to get a better method of calculating levels of depression which can assist in evaluating particular user depression.

Nadeem, Moin[17] investigated the capacity of online users of social media to anticipate Major Depressive Disorder (MDD), before it really starts. They used a crowd - sourced technique to create a list of depression-diagnosed twitter users. Using bag of word approach upon this tweets collected over a year, different statistical classifiers were then utilized to give some estimates of the risk of depression.

Through a Lexicon-based approach, Andrew G. Reece, Andrew J. Reagan, Katharina L. M. Lix, Peter Sheridan Dodds, Christopher M. Danforth and Ellen J. Langer[19] proposed a method to analysing the feelings from twitter and facebook writings. Diveesh Singh, Aileen Wang [18] created model to find the depression in particular user by creating the model with the help of collection of tweets of depressed people.

Nowak J., Scherer R., Taspinar A. [20] showed how to classify data with the help of Long Term Term Memory (LSTM) and its adjustments, i.e. Bidirectional LSTM network.

Chapter 3

Theoretical Background

We will be surveying posts of people on social media like Twitter. Our main focus will be on analyzing the tweets of users to see whether their overall sentiment is positive or negative.

3.1 Sentiment Analysis

Sentiment analysis can be performed in mainly 2 ways.

3.1.1 Supervised Machine Learning

For this method we need a large dataset with data which is already labelled with sentiments. This data is then used to train ML models. ML models learn the features of the data which is provided to them while training. Then these models are used to predict sentiment value of some data which don't already have some known sentiment value. Apart from this, unsupervised machine learning can also be used to give labels to large unlabeled datasets using methods like clustering.

Decision Tree and Random Forest Classifier

Major advantage of Decision Tree classifiers is their speed as compared to other probabilistic or linear classifiers. Ensemble of various decision tree classifiers and combining them with other methods like weighing their outputs will form a Random Forest Classifier. It is a naïve but really good approach for datasets with less number of features. In problems like sentiment analysis where each and every word can be considered as a feature (there can be thousands of features), these methods can turn out to be pretty slow.

Probabilistic Classifiers

Basically these classifiers provide us with a probability whether the tweet is positive or negative. Generative Models where the joint probability distribution of a sample and its label is learned and it is used to "Generate" a new data point using methods like sampling. Naïve Bayes Classifier is an example of Generative classifiers, where X being the word and Y being the label, calculates P(Y|X) by finding P(Y) and P(X|Y). Various

Algorithms like Naïve Bayes, Bayesian Networks and Maximum Entropy Classifier can be used.

Linear Classifiers

In this method, the feature space is divided into multiple regions which are labelled as different class. The boundaries separating these regions can be a Line, plane or a hyperplane based on different cases. There various methods for linear classification like Support Vector Machines and Neural Network.

- Ideally SVM is much efficient in working with text because a nonlinear decision surface can be generated to separate features which are very sparse.
- Neural Networks are the most famous and versatile machine learning which can be used for training with almost any kind of data. Neural networks, just like SVMs, also perform equally good on textual data.

3.1.2 Lexicon based approach

These are some rigid techniques depending on statistics to determine the sentiment value. This is done by creating data structure called a "Dictionary" of words which are coupled with their Semantics Polarity and Sentiment Strength. As these methods depend on pure calculations to determine the output, the "precision" is pretty high whereas "recall" is relatively lower due their rigid rules. This can further be classified into 2 approaches.

Dictionary Based Approach

It starts with a small set of seed words, which have their sentiment values stated already, and is then expanded into a larger set using the synonyms and antonyms. These new words are added to the main list and along with their sentiment values. This process continues till there are no newer words being added to the main list. The main disadvantage of this approach is that it cannot find words related to one specific domain. For example, the sentiment of the word "positive" can depend on where it is being used. It terms of some review it simply means it is "positive", but in terms of a medical test, "positive" actually means that the patient has the disease which is a negative sentiment.

Corpus Based Approach

This also starts with a set of seed words and the analysis is done using the context specific orientations of these seed words. These methods not only depend on the seed words, but also the list of words surrounding them. Using this method alone is not good enough as compared to dictionary based approach. It can also be performed in two ways i.e. Statistical and Semantic.

3.2 Database

Database is an important part of any project used for analysis of data. Databases can be of any type like relational, NoSQL, graph-based, etc. When dealing with highly related data like social media, using a relational database can degrade the performance of your

application because of the time it takes to create and work on relations between various table. In NoSQL databases, it is tough to specify relations. Whereas in graph databases are perfect for such use because of the way in which data is stored in them and how easy it is to use them. Graph databases are also really flexible when it comes to adding new data. Many times the developer does not know what kind of data is going to be added to the database when the project has just started. In a graph database, adding new data and creating new relationships does not disturb other parts of database at all. We are using Neo4j as a graph based database.

3.2.1 Neo4j Graph Database

Neo4j is a Graph database developed on Java but it can be used on various platforms like Python, Java, etc. We are using neo4j in python using its library "py2neo". Some of really good features of Neo4j are:

Cypher Query Language

Cypher is a query language which right now works exclusively in Neo4j. It is a simple and easy to understand language. The similarity between Cypher and the actual way of presenting graphs is what makes it a great choice for working with Neo4j.

(:ACTOR{name:"David"}) – [:PLAYS] -> (:CHARACTER{name:"Doctor Who"})

The above text is an example of Cypher query which shows the graph of node actors named "David" who have played a role called "Doctor Who". Here "ACTOR" and "CHARACTER" are called nodes of the graph and "PLAYS" is called the relationship between "ACTOR" and "CHARACTER".

API endpoints

Neo4j also provides API endpoints which can be REST API or Bolt API. These Endpoints can be called using programs written any language. Hence these APIs can be used universally by any language. This is what makes Neo4j really accessible and adaptable.

3.3 Word Embedding

Word Embedding is method if extracting features from text data. In this method either a word or set of words (e.g. document or paragraph) are transformed into vectors that are made of real numbers. In basic terms, embeddings are text converted into numbers. There can be only distinct numerical expression for the same text.

3.3.1 Why word Embedding is required?

Almost every machine learning algorithms and deep learning algorithms work on features made of numeric values. These algorithms can't take row text data as input. Somehow, we need to convert these text data to numeric data. E.g. implementing classification or regression algorithms on text data in its row form is nearly impossible. That's why we need some algorithms that could convert these text data to numeric data.

3.3.2 Different types of Word Embeddings

Word Embeddings can be categorized into two major parts based on it's functioning.

1. Frequency based Embedding
2. Prediction based Embedding

Frequency based Embedding

Under this class, we usually find three types of vectors.

1. **Count Vector :**
   In this method, we define certain list of N words or tokens, which we call corpus C. This corpus is defined for documents that are going to be converted into numerical vectors. We call these D documents d1,d2,. . . ,dD. A matrix M of dimension D*N is created which is called count vector matrix M. Each row (i) of this matrix represent numeric form of d(i) document. The tile in matrix M, M(i,j) is a frequency of jth word from a corpus (cj) in ith document (di).

   Now, while preparing the above matrix M, there may be quite a few variances.

   (a) The way dictionary is prepared.
   We might have a corpus of real world applications that includes millions of documents. These documents need a corpus that can get the nearest possible numerical presentation. For that, we might need a corpus that is very large. If we do that, most of the entries in count vector will be 0 which is clearly waste of storage. The alternative approach that has been proposed is to find top few words that are important in specific application and use those words for a corpus C.

   (b) The way each entry is made in vector.
   Each number in a vector representation can be a frequency or a proof of existence of certain word in a document. In first approach each entry can be any natural number based on frequency of a word in a document. In second approach we limit the entries to 0 or 1 based on existence of a word in a document.

2. **TF-IDF Vector :**
   In this approach, while making the vector for a certain document, it takes frequency of certain word in all documents instead of just one.
   Some very common words in English grammar like “a”, “an”, “is” ,”this” occurs in written text document way often opposed to the words that really matter to a certain text. For instance, in comparison with other documents, a document A on Lionel Messi would include many instances of the word "Messi" But commonly used words such as "the" etc. will also be available in almost every document on a higher frequency.
   Basically, we would like to prefer more priority to the words appearing in subgroups of documents instead of words appearing more frequently in every document.
   TF-IDF approach gives lower priorities to words that appear commonly in each document. At the same time, it emphasizes terms like Messi in a specific document.

TF(Term t, Document doc) = (No. of times t appears in a doc.)/(No. of terms in the doc.)
TF is counted for single document. It gives the weightage of a term in a given document.
IDF(Term t) = log(Number of Documents / Number of Documents with t)
IDF is counted for a certain term with respect to all the documents. If a certain number seems to appear in all the documents then IDF of that term will be 0.
TF-IDF(t, document) = TF(t, document) * IDF(t)
Using this formula, we can find TF-IDF of a certain word in a given document. This formula takes TF and IDF both into account. So, a weightage of a word in given document depends on not only the frequency of a word in given document but also the availability of a word in all the document. If a word is available in all the document (assuming words like 'a','an','is'... appear in all documents) , it gets 0 weightage.

3. **Co-Occurrence Vector :**
   This approach tries to save connection between words that occur together. It helps to preserve context of a document instead of just frequencies. In previous methods, there were no sign of context of a word. Those methods just take frequencies into account. The word "apple" when used in context of a fruit and when used in context of a tech-giant are same for previous methods. This method tries to differentiate these two words. There is a concept of a Context Window in this approach that suggest how many neighbors from which direction we need to take in consideration in order to build the context. Size 2 context window can be shown below

Figure 3.1: Context Window of Size 2

Below is an example of co-occurrence matrix :

Corpus : He is not lazy. He is intelligent. He is smart.

Prediction based Embedding

We've seen deterministic ways of identifying word vectors. But these methods are way too rigid and limited. Then word2vec was invented. This method is based on probabilities of a word and can do the tasks like word-similarities. In this method the vectors we get can be added and subtracted with context. Addition of vector of happy and vector of man gives a vector that suggest "happy man" in some multidimensional vector space.

1. **Word2Vec**
   Word2vec stands for concepts in the expression of vector space. Words are expressed in the form of vectors, and positioning is done in such a way that words of similar meaning appear together and words of dissimilar meaning are distant. This is also called a semantic relationship.
   Word2Vec is created using two algorithms – CBOW and skipgram. These algorithms are based on neural network. In these neural networks, words are assigned

	He	is	not	lazy	intelligent	smart
He	0	4	2	1	2	1
is	4	0	1	2	2	1
not	2	1	0	1	0	0
lazy	1	2	1	0	0	0
intelligent	2	2	0	0	0	0
smart	1	1	0	0	0	0

Figure 3.2: Co-Occurence Matrix

to the target variable. This target variable can also be word(s). Both of these algorithms obtain vector of some float numbers that serve as a representation of a word hence word vectors.

(a) **CBOW Model**
CBOW stands for Continuous Bag Of Words. In CBOW, the context of the word is taken as input and based on that, CBOW tries to predict the word, which is of similar meaning.
As shown in Figure 3.3, Input for this model is context word which is first converted into One-Hot Encoded vector of size V. Then Weight matrix W will map inputs to hidden layer which is consist of N neurons and so weight matrix W' will map output of hidden layer to output layer with softmax activation function.

As shown in Figure 3.4, if we change the input to multiple context words, then also CBOW can be used to predict a similar context word. In that case, the hidden layer will take an average of all context word mapping values, and further, it will be the same as usual.
CBOW takes very little time to train, and it performs well for the words which are more frequent.

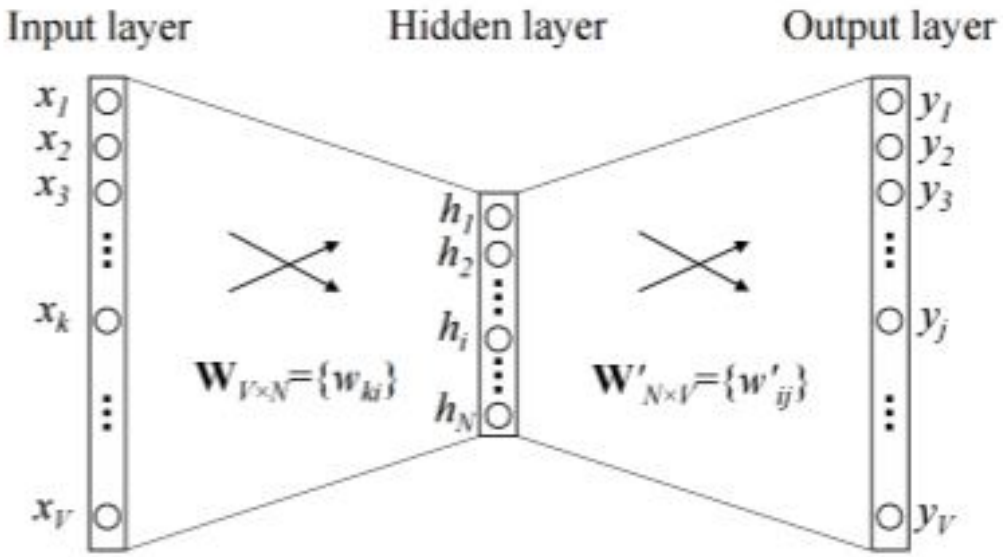

Figure 3.3: CBOW Model

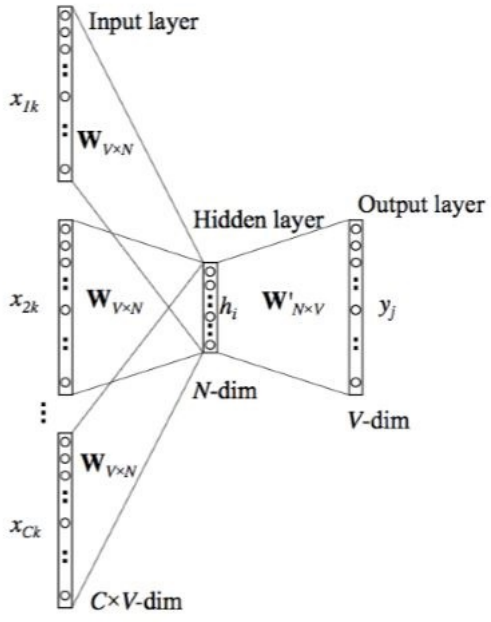

Figure 3.4: CBOW Model for multiple context words

(b) **Skip-Gram Model**
Skip-Gram Model is totally opposite of CBOW Model. In Skip-Gram Model, The input will be the target word, and it will try to predict the context, and it's representation.

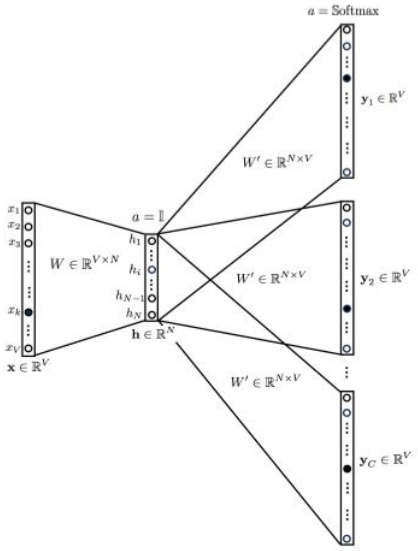

Figure 3.5: Skip-Gram Model

As shown in Figure 3.5, Input for this model is target word which is first converted into One-Hot Encoded vector of size V. Then weight matrix W will map input layer to hidden layer which is consist of N neurons and W' will map output of a hidden layer to output layer which is C probability distributions of vector of size V, one for each word.
Skip-Gram models give a good result when trained on smaller dataset and words which comes rarely are represented well.

2. **GloVe**
GloVe stands for Global Vectors. GloVe is combines the advantages of two major model families in the literature named as global matrix factorization and local context window. In other words GloVe combines advantages of Word2Vec model and matrix factorization. Main idea behind GloVe is that words that are appearing next to each other carries more information than individual.[15]

Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	67.5	54.3	60.3
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	64.8	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	80.8	61.5	70.3
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW†	300	6B	63.6	67.4	65.7
SG†	300	6B	73.0	66.0	69.1
GloVe	300	6B	77.4	67.0	71.7
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	**81.9**	**69.3**	**75.0**

Figure 3.6: Comparision of Word Representation Methods[15]

As we discussed above GloVe posses all the advantages of Word2Vec and addition to that GloVe works well for both small corpus and large corpus. Training time also small compared to all the techniques. In Figure 3.6, All the comparision is showned.

3.4 Support Vector Machine (SVM)

"Support Vector Machine" (SVM) is really a supervised technique for M.L., which could indeed be utilized to analyze classification and regression. I it is primarily utilized in classification. In the SVM algorithm, we graph each data item as a point through n - dimensional space (in which n is the quantity of attributes you possess), for each feature becoming the value of a particular co - ordinate. After which, by finding the hyper-plane which differentiates the two classes very well, we perform classification.[10]
there seem to be several possible hyperplanes that may be chosen to differentiate among the two forms of data locations.Our purpose is to discover the highest possible edge of a plane, such that, the highest distance between the data points of both categories. Optimizing the margin from the edges provides some clarification such that near term data sets can be more confidently assessed.

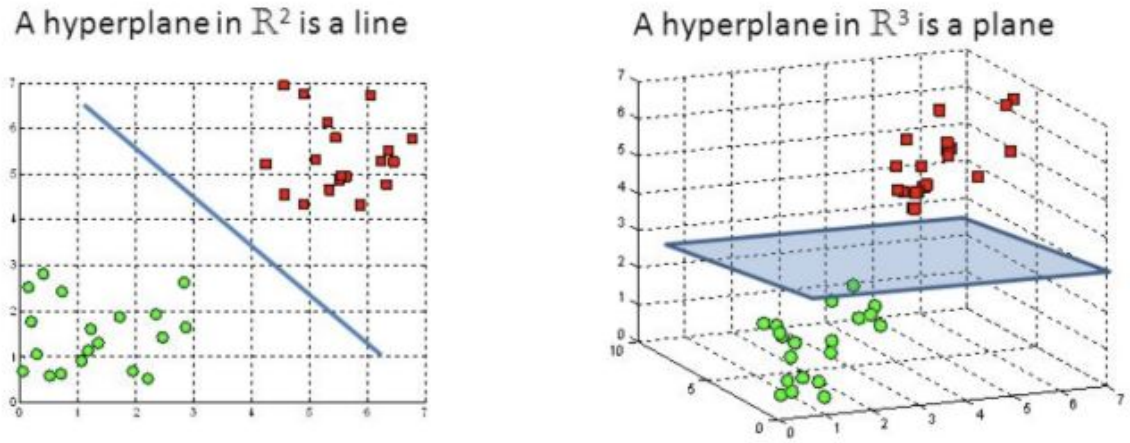

Figure 3.7: Hyperplane in SVM

Hyperplanes are constraints to making decisions that help differentiate data points. Points can be appointed to different groups that lie on either portion of its hyperplane. The dimension of the hyper - plane sometimes varies based on how many features it seems to have. Support vectors are data locations nearer to hyperplane, which influence the hyperplane's direction and orientation. Using these vectors for help we optimize the classifier's range. Removing the support vectors will alter the hyperplane location. SVM can be created using these points.[9]

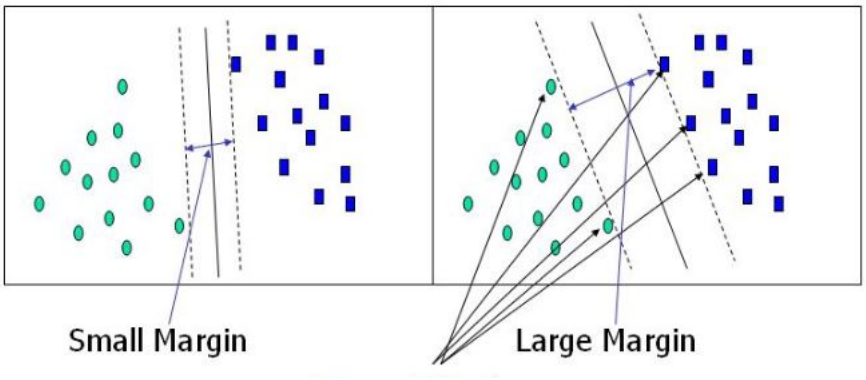

Figure 3.8: Support Vectors in SVM

3.4.1 Kernels

svm techniques employ the collection of computational operations identified as kernel. The kernel role is to start taking info as input and turn it to the necessary type. Various svm techniques employ distinct kernel function kinds. There may be various variations of those functions. Like Radial Basis Function, Sigmoid, Polynomial.

1. Polynomial kernel
   That's also widely known in the image processing technique.

$$f(x_i, x_j) = (x_i * x_j + c)^d \tag{3.1}$$

   Where d is quadratic level.

2. Gaussian kernel
   This is a kernel of overall need, utilized since there is no previous information of a data.

$$k(x, y) = exp(-\frac{||x - y||^2}{2\sigma^2}) \tag{3.2}$$

3. Radial basis function (RBF)
   This is a kernel of overall need, utilized since there is no previous information of a data.

$$k(x_i, x_j) = exp(-\gamma||x_i - x_j||^2) \tag{3.3}$$

4. Sigmoid kernel
   Can be utilize it as neural network proxy.

$$k(x, y) = tanh(\alpha x^T y + c) \tag{3.4}$$

3.5 Recurrent Neural Network (RNN)

Neural Networks are collection of algorithms which accurately represent the actual neural network and are established to spot patterns. They interpret sensory inputs via the classification of perceptions or the clustering of raw information from a system. Specific patterns can be identified with the help of Neural Networks, from the vectors that are needed to be transfer from real-world data, such as images, sound, text and time series. Artificial neural networks consist of an abundance of greatly interlinked (neuron) processing units functioning cooperatively to solve a problem.[11]
Recurrent Neural Network (RNN) is an internally memorized generalization of feed forward neural network. In general, RNN is recurrent, as it conducts the alike purpose for each data input while the output of the contemporary input relies on the prior one. An output is duplicated and brought back into the recurrent networks after it is generated. This ponders the present input and the feedback it has accumulated from the prior one for making a decision. [12]
Different from feed forward neural network, RNN is allowed to operate input sequences utilizing their internal state (memory) which facilitate them for activities, such as unsegmented, text analysis, linked recognition of handwriting or recognition of voice. All inputs are autonomous one from the other in other neural networks. As far as RNN is concerned, all the inputs are interrelated.

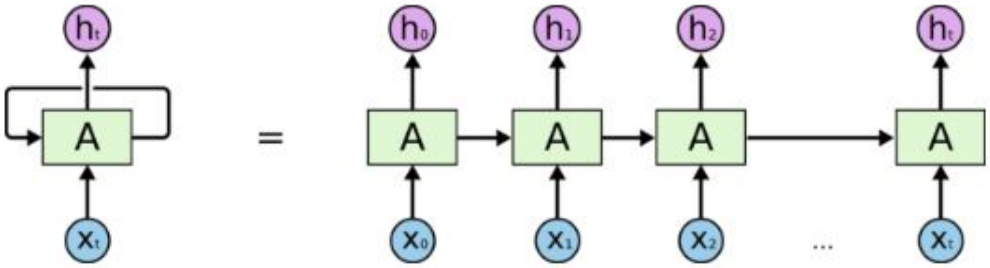

Figure 3.9: RNN in unrolled form

Initially, it requires the X(0) from the input vector, and then outputs h(0) which would be the input for the next step alongside with X(1) therefore the input for following step is h(0) and X(1). Likewise, h(1) from the next step is X(2) input for the subsequent step, and so forth. In this manner, it maintains on memorizing the context during training.
For the present state the formula is:

$$h_{\mathrm{t}} = f(h_{\mathrm{t\text{-}1}}, x_{\mathrm{t}}) \tag{3.5}$$

Applying an Activation function :

$$h_{\mathrm{t}} = tanh(W_{\mathrm{hh}} h_{\mathrm{t}} + W_{\mathrm{xh}} x_{\mathrm{t}}) \tag{3.6}$$

W is weight, h is the hidden layer, Whh is the weight at the prior hidden state, Whx is the weight at the present input state, tanh is an activation function, which introduces a Non-linearity that crushes activations to the range[-1.1].

$$y_{\mathrm{t}} = W_{\mathrm{hy}} h_{\mathrm{t}} \tag{3.7}$$

The output state is Yt. The output weight is Why at state. Returning to inscribing each time and upgrading its weights is indeed a slow operation. It draws computation power as well as time. You may run into some problems during back propagation. One of which is Vanishing Gradient.

3.5.1 Vanishing Gradient

When using backpropagation over time, you'll find that an error is the difference between the actual and the predicted model.
If slight derivation of an error is less than 1, then multiply it by the learning rate that is also much lower. Compared to the prior iteration, multiplying the learning rate with slight error derivation would not be a significant change then.

For instance, Let's assume the value fell as 0.863 -> 0.532 -> 0.356 -> 0.192 -> 0.117 -> 0.086 -> 0.023 -> 0.019.

You can observe that the previous three iterations don't make much of a change. This disappearing of gradient is known vanishing gradient. This vanishing gradient problem also effects in a neglect of long-term dependencies throughout training phase. During several years, various solutions to the problem of the vanishing gradient have been suggested. The best known are the LSTM and GRU modules but this is still an emerging area of research.

3.5.2 LSTM (Long Short Term Memory)

For such a long period, LSTM's have such an aspect of knowing info is ones default conduct.[13] LSTM has a three step Process:

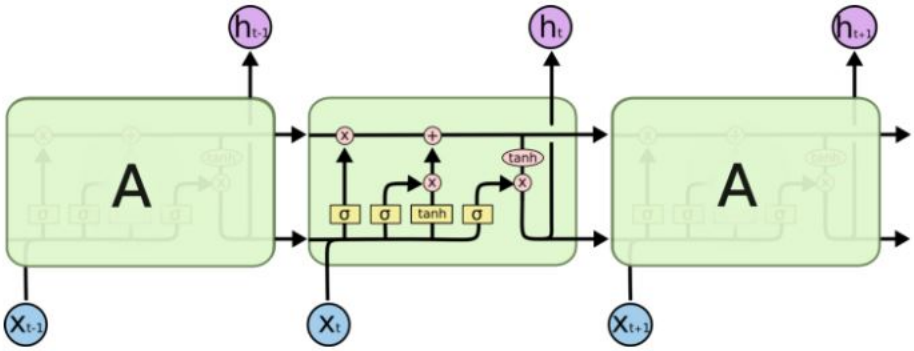

Figure 3.10: LSTM model

Look at the Figure 3.10, Which tells every other LSTM configuration has 3 gates called Forget gate, Input gate, Output gate.

Forget Gate

This gate determines what information from the block to remove. The sigmoid feature decides upon it. Consider the prior state (ht-1) and content input (Xt) and outcomes a

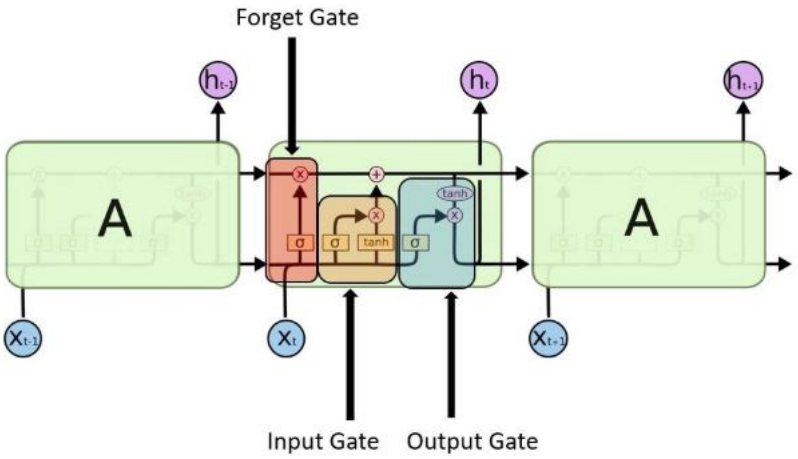

Figure 3.11: LSTM Gates

quantity for each number in Ct-1 cell state between 0 (exclude this) and 1 (keep this).

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{3.8}$$

Input Gate

This input gate governs that what input value to modify the memory should be used. Sigmoid function determines which values to allow by 0 to 1 and tanh function allows weighting of the passed values to calculate their degree of significance from -1 to 1.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{3.9}$$
$$C_t = tanh(W_C[h_{t-1}, x_t] + b_C) \tag{3.10}$$

Output Gate

The output is calculated using the input and the block memory. The Sigmoid function determines which values are to be allowed by 0,1. The tanh function then gives weighting of the values passed to determine their significance ranging from -1 to 1 and multiplied by Sigmoid output.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{3.11}$$
$$h_t = o_t * tanh(C_t) \tag{3.12}$$

3.5.3 Bi-LSTM (Bi-directional LSTM)

Bidirectional recurrent neural networks(RNN) really merely put together two separate RNNs. This model allows the systems to include information about the sequence both backwards and forward at each time interval.
Having bidirectional can execute the inputs in two directions, one is from past to future and one from future to past, and what distinguishes this method from unidirectional would be that you retain information from the future in the LSTM that runs backwards and use the two hidden states merged to retain information from the past and also the future at any moment in time.

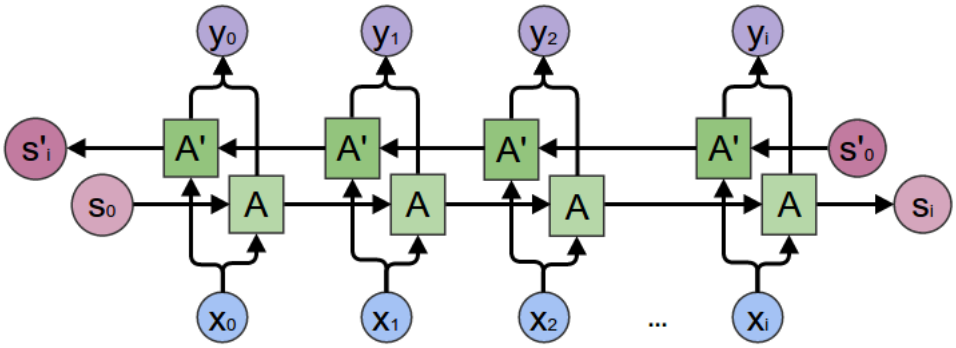

Figure 3.12: Bi-LSTM

3.6 LSTM over SVM

Many NNs are trained to classify and compute the effectiveness of the same set of text documents with SVMs. The performance of these two methods is then statistically compared. The performance of NNs is statistically comparable with that of text classification (TC) SVMs. Even if a substantially reduced document size is being used, their performance can be compared. This research finds that NNs are good alternative TC tools with performance that is comparable to SVMs. Moreover, NNs also require a reduced document size. Successful use of NNs to classify reduced text documents will be one of its greatest advantages because it can bring substantial savings in computation and cost.[14]

3.7 Random Forest (RF)

Random forests or random decision forests are an ensemble learning technique that works by building a large number of decision trees at training time. Instead of just averaging the prediction of trees (that we might term a "forest"), this model utilizes two core points that gives this algorithm name random:

1. When building trees, sample training data points randomly.
2. When splitting nodes, random subsets of features are taken into account.

Random sampling of training observations

Every tree learns during the training process from a random sample of the data sets in a random forest. The samples are selected with replacement, regarded as bootstrapping, which indicates that many samples are used in a single tree several times. The main idea here is that each tree is trained on different samples. Although each tree may have high variance with respect to specific data, the whole forest will be having lower variance, of course at the expense of higher bias. During testing phase, predictions are made by an average of each decision tree's predictions (in regression problem) and mode of the predicted classes of each decision tree (in classification problem). This procedure to train each individual learner on various bootstrapped data subsets and then to average the predictions is termed as bagging.

Random Subsets of features for splitting nodes

The other core aspect in the random forest is that for splitting each node into each decision tree, only a subset of all the features is considered. This is usually set to square root of number of features for classification, indicating that if there are 25 features, only 5 random features will be taken into consideration for splitting the node at each node in each tree. (This is common in regression that the random forest can also be trained, analysing all the features at each node. These alternatives can be controlled in the implementation.)

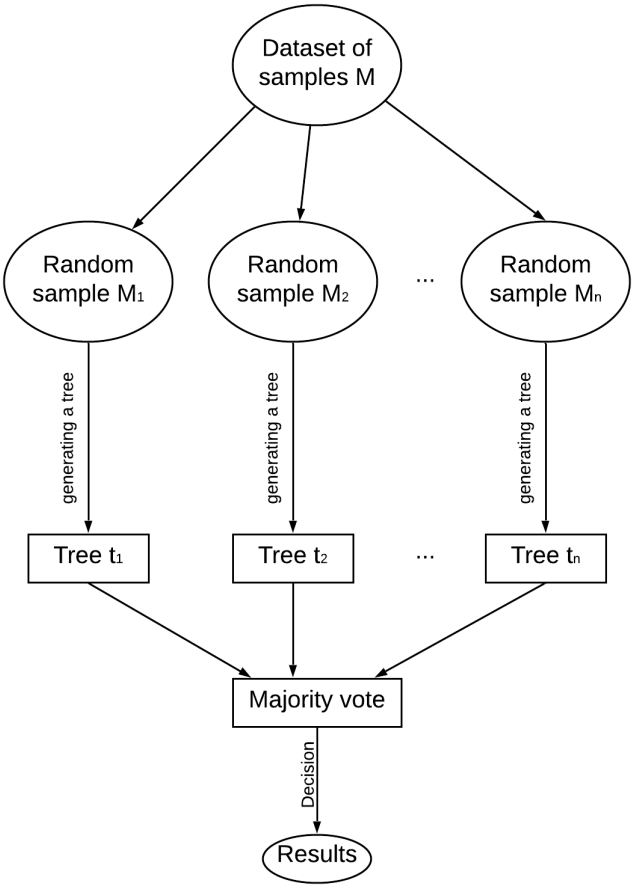

Figure 3.13: Struture of RF[24]

Chapter 4

Proposed Work

The goal is to build a network of Twitter users using Neo4j as a database and perform sentiment analysis on tweets of these users in order to see whether their behavior is positive or negative.

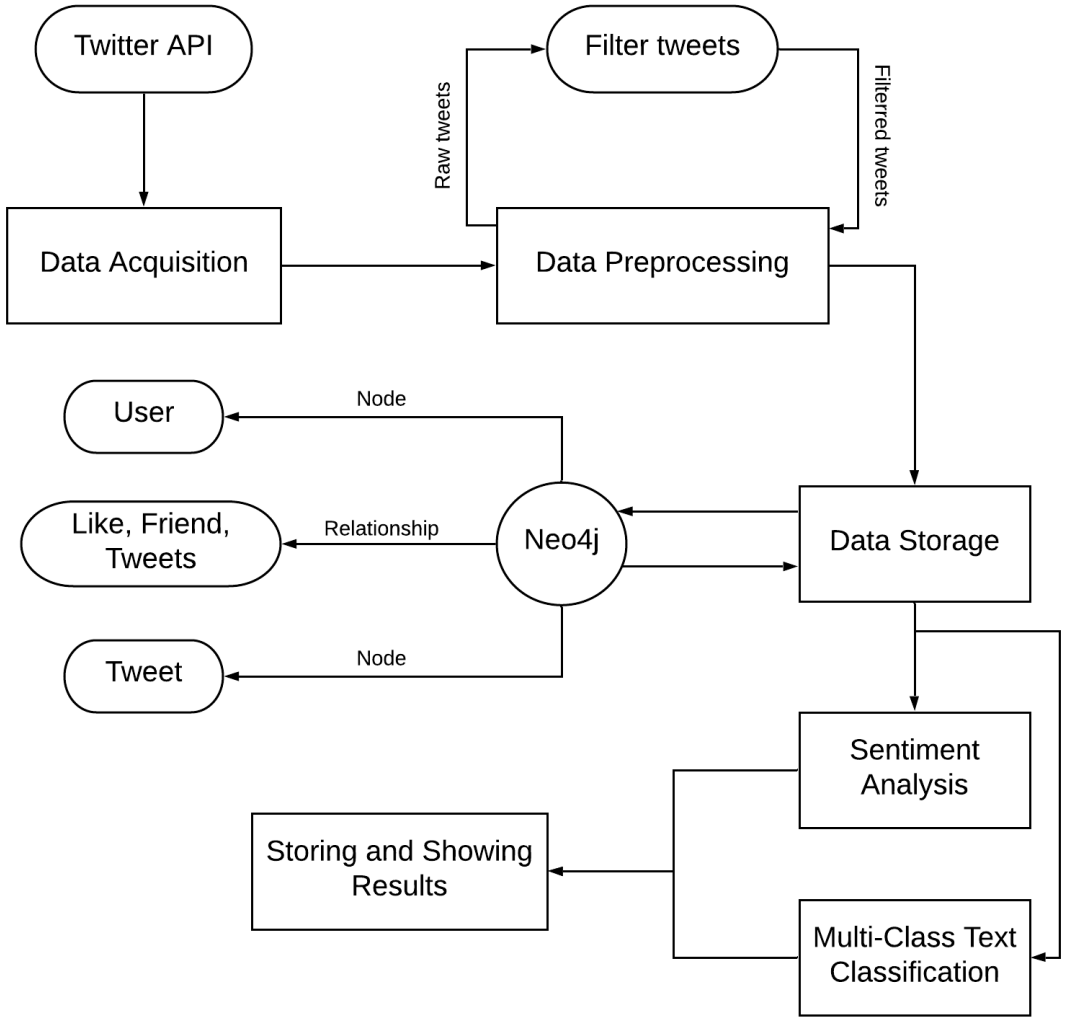

Figure 4.1: Flow Diagram

4.1 Data acquisition and storage

This step includes gathering of data on which sentiment analysis is to be performed from various sources. We are using Twitter API for gathering tweets. Twitter API is a great tool for gathering tweets of Twitter. Once the account for Twitter Developer is set up, one can easily extract information like tweets by a particular user, friends & followers of a user, tweets showing on the home page of a user, etc. For instance, someone wants to get data of all the users of institute. They can easily search for users who are in that specific area using the "Geocode" feature of the API.
As mentioned earlier, we are using Neo4j as a database for storing the data collected using Twitter API. We are using the library "py2neo" for this purpose. This library helps us to access the Bolt API server of Neo4j from Python. The database has various nodes like "User" and "Tweet". The nodes can be related using one or more relationships. Some of the relationships can be "Tweets" which exists between "User" and "Tweet" nodes. There are other relationships like "Likes", "Retweets", "Follows", "Friends", etc.

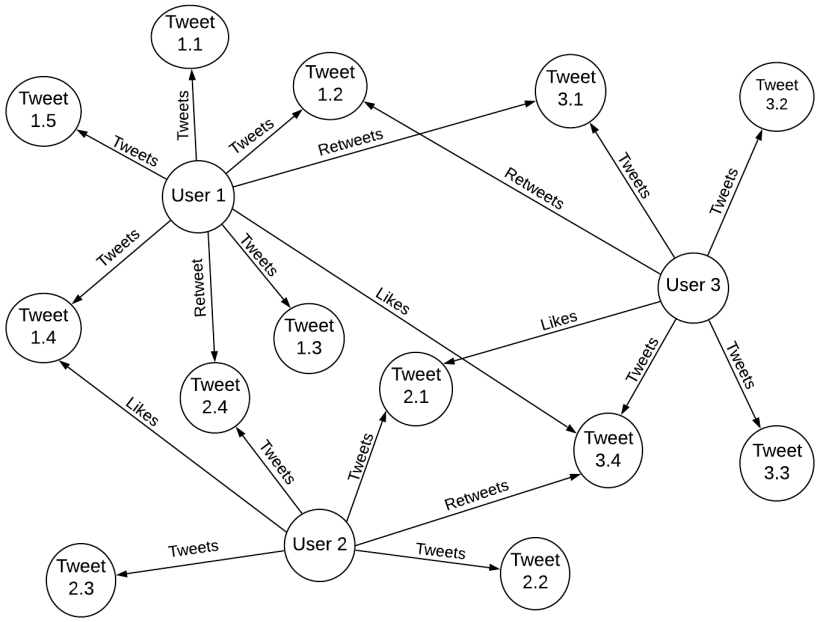

Figure 4.2: Database Diagram

4.2 Data Preprocessing

As stated in [2], the information gathered via the Twitter API is incredibly raw and full of uncertainties. If that data is transferred straight to the model of machine learning, it would not be able to work properly. Therefore, the collected data is passed through several phases of filtration. Here is a short description of the potential processes.

4.2.1 Preprocessing Using tweet-preprocessor library

Preprocessor is a Tweet Preprocessing python - based library. A preprocessing is required when constructing Machine Learning systems based on tweet data. This library makes

the cleaning, parsing or tokenizing of tweets easy.
Currently supports cleaning, tokenizing and parsing:

- URLs
- Hashtags
- Mentions
- Reserved Words (RT, FAV)
- Emojis
- Smileys

```
import preprocessor as p
for i in range(num):
  f_data['filtered_text_01'][i] = p.clean(data['text'][i])
```

Figure 4.3: Code Snippet for Tweet-Preprocessor

```
0  @switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer.  You shoulda got David Carr of Third Day to do it. ;D
1  is upset that he can't update his Facebook by texting it... and might cry as a result  School today also. Blah!
2  @Kenichan I dived many times for the ball. Managed to save 50%  The rest go out of bounds
3  my whole body feels itchy and like its on fire
4  @nationwideclass no, it's not behaving at all. i'm mad. why am i here? because I can't see you all over there.
5  @Kwesidei not the whole crew
6  Need a hug
7  @LOLTrish hey  long time no see! Yes.. Rains a bit ,only a bit  LOL , I'm fine thanks , how's you ?
8  @Tatiana_K nope they didn't have it
9  @twittera que me muera ?
```

Figure 4.4: Input Tweet-Preprocessor

```
0  - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it.
1  is upset that he can't update his Facebook by texting it... and might cry as a result School today also. Blah!
2  I dived many times for the ball. Managed to save % The rest go out of bounds
3  my whole body feels itchy and like its on fire
4  no, it's not behaving at all. i'm mad. why am i here? because I can't see you all over there.
5  not the whole crew
6  Need a hug
7  hey long time no see! Yes.. Rains a bit ,only a bit LOL , I'm fine thanks , how's you ?
8  nope they didn't have it
9  que me muera ?
```

Figure 4.5: Output Tweet-Preprocessor

4.2.2 Extra Preprocessibng After Tweeter-preprocessor Library

This method will clear some remains of the tweets left unfinished by tweet-preprocessor and double-check emoji and emoticon's since some older version of emoticons from mobile is not supported in the clean system of tweet preprocessor.

```
import nltk
nltk.download("stopwords")
nltk.download("punkt")
from nltk import word_tokenize
import string

def clean_tweets(tweet):

    stop_words = set(stopwords.words('english'))
    word_tokens = word_tokenize(tweet)
#after tweepy preprocessing the colon symbol left remain after      #removing mentions
    tweet = re.sub(r':', '', tweet)
    tweet = re.sub(r',Ä¶', '', tweet)
```

Figure 4.6: Code Snippet for Extra Preprocessing

```
#replace consecutive non-ASCII characters with a space
    tweet = re.sub(r'[^\x00-\x7F]+',' ', tweet)
#remove emojis from tweet
    tweet = emoji_pattern.sub(r'', tweet)
#filter using NLTK library append it to a string
    filtered_tweet = [w for w in word_tokens if not w in stop_words]
    filtered_tweet = []
#looping through conditions
    for w in word_tokens:
#check tokens against stop words , emoticons and punctuations
        if w not in stop_words and w not in emoticons and w not in string.punctuation:
            filtered_tweet.append(w)
    return ' '.join(filtered_tweet)
```

Figure 4.7: Code Snippet for Extra Preprocessing

```
0   - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it.
1   is upset that he can't update his Facebook by texting it... and might cry as a result School today also. Blah!
2   I dived many times for the ball. Managed to save % The rest go out of bounds
3   my whole body feels itchy and like its on fire
4   no, it's not behaving at all. i'm mad. why am i here? because I can't see you all over there.
5   not the whole crew
6   Need a hug
7   hey long time no see! Yes.. Rains a bit ,only a bit LOL , I'm fine thanks , how's you ?
8   nope they didn't have it
9   que me muera ?
```

Figure 4.8: Input for Extra Preprocessing

```
0   Awww 's bummer You shoulda got David Carr Third Day
1   upset ca n't update Facebook texting ... might cry result School today also Blah
2   I dived many times ball Managed save The rest go bounds
3   whole body feels itchy like fire
4   's behaving 'm mad I ca n't see
5   whole crew
6   Need hug
7   hey long time see Yes.. Rains bit bit LOL I 'm fine thanks 's
8   nope n't
9   que muera
```

Figure 4.9: Output for Extra Preprocessing

4.2.3 Convert Tweets to Lower Cases

Lowering all of the text data, though commonly neglected, is one of the easiest and most direct pre-processing of text. This applies to most problems with text mining which NLP, and can help in cases where the dataset is not very large and greatly helps with planned performance quality.

```
for i in range(num):
  f_data['filtered_text_02'][i]=f_data['filtered_text_02'][i].lower()
```

Figure 4.10: Code Snippet for Lowering the Case

```
0   Awww 's bummer You shoulda got David Carr Third Day
1   upset ca n't update Facebook texting ... might cry result School today also Blah
2   I dived many times ball Managed save The rest go bounds
3   whole body feels itchy like fire
4   's behaving 'm mad I ca n't see
5   whole crew
6   Need hug
7   hey long time see Yes.. Rains bit bit LOL I 'm fine thanks 's
8   nope n't
9   que muera
```

Figure 4.11: Input for Lowercase

```
0   awww 's bummer you shoulda got david carr third day
1   upset ca n't update facebook texting ... might cry result school today also blah
2   i dived many times ball managed save the rest go bounds
3   whole body feels itchy like fire
4   's behaving 'm mad i ca n't see
5   whole crew
6   need hug
7   hey long time see yes.. rains bit bit lol i 'm fine thanks 's
8   nope n't
9   que muera
```

Figure 4.12: Output for Lowercase

4.2.4 Lemmatization

Lemmatization[7] typically refers to doing things correctly using a vocabulary and morphological examination of words, generally aimed solely at eliminating inflection endings and restoring the basic term referenced as lemma. Lemmatization is done by using different plug-in feature to the indexing process, and there are a variety of these modules, commercial as well as open source.

```
import spacy
nlp = spacy.load('en', disable=['parser', 'ner','tagger'])
def Lemmatization(text):
    doc = nlp(text)
    ans = " ".join([token.lemma_ for token in doc])
    return ans
```

Figure 4.13: Code Snippet for Lemmatization

```
0   awww 's bummer you shoulda got david carr third day
1   upset ca n't update facebook texting ... might cry result school today also blah
2   i dived many times ball managed save the rest go bounds
3   whole body feels itchy like fire
4   's behaving 'm mad i ca n't see
5   whole crew
6   need hug
7   hey long time see yes.. rains bit bit lol i 'm fine thanks 's
8   nope n't
9   que muera
```

Figure 4.14: Input for Lemmatization

```
0   awww 's bummer you shoulda get david carr 3 day
1   upset ca not update facebook texting ... may cry result school today also blah
2   i dive many time ball manage save the rest go bound
3   whole body feel itchy like fire
4   's behave ' be mad i ca not see
5   whole crow
6   need hug
7   hey long time see yes .. rain bite bite lol i ' be fine thank 's
8   no not
9   que muera
```

Figure 4.15: Output for Lemmatization

4.3 Training ML model

There are various methods in Machine Learning alone to use for sentiment analysis. But the most popular and effective are 3 methods, namely Naïve Bayes Classifier, Support Vector Machines (SVM) and Neural Networks. We will be training ML models using all 3 methods to check which model performs well on our data. Based on the performance results we get we will decide to go ahead with one of these methods.

4.3.1 Sentiment Analysis

Using the ML model which was trained as described earlier, we predict sentiment values of the tweets stored in our database. The obtained sentiment values are also stored in the database with their respective tweets. These sentiment values are then analyzed to see behavior patterns of the users. If some user is showing worrying signs, then we will get to know about it.

We have currently trained two different classifiers SVM and Bidirectional LSTM for sentiment analysis.

SVM

We converted all tweets to vector form using Glove embeddings. We are appending vector of each word to vector of corresponding tweet. This way we don't lose the sequence of words in a tweet.
We applied soft margin SVM with different kernels on a dataset of 10,000 tweets. Following table consists of training and testing accuracies for each kernel at different values of C.

Kernel	C	Testing Accuracy
Linear	0.1	64%
Linear	1	65%
Linear	10	65%
Linear	100	65%
Rbf	0.1	63%
Rbf	1	66%
Rbf	10	66%
Rbf	100	65%
Sigmoid	0.1	64%
Sigmoid	1	56%
Sigmoid	10	55%
Sigmoid	100	55%
Poly	0.1	55%
Poly	1	61%
Poly	10	63%
Poly	100	63%

Table 4.1: SVM Result of Sentiment Analysis.

As we can see in Table 4.1, at C =1, most of the kernels gives best accuracies respectively.
In following Figure 4.16 we are comparing different kernels at C = 1. As we can see in the Figure 4.16, Kernel RBF gives the best result (66 %) at C=1. In C = 10, We are also getting 66% accuracy, but we have observed that it too closely fits to training data so it shows the sign of over-fitting.

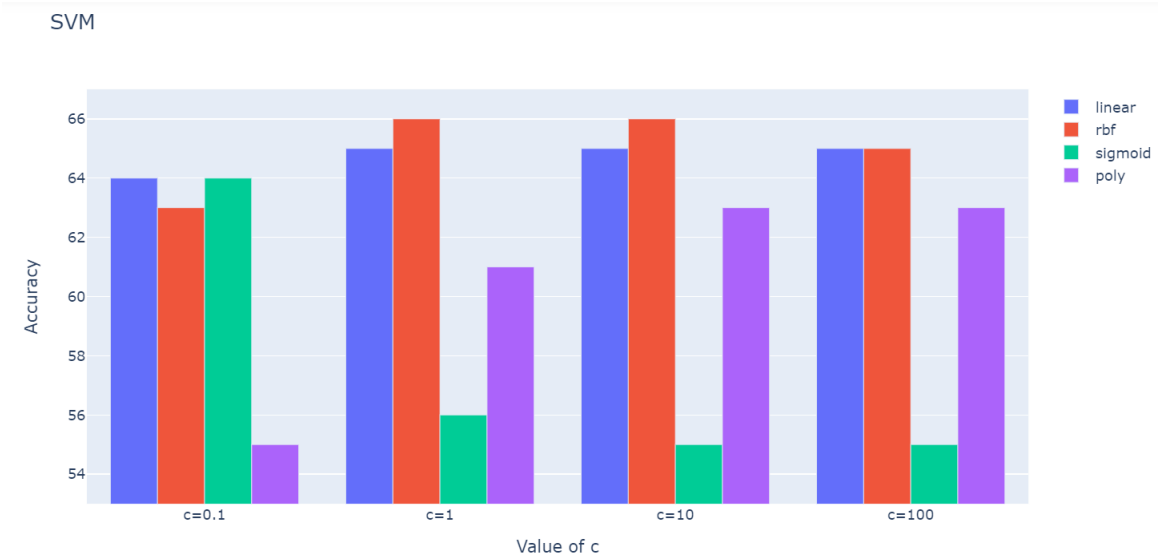

Figure 4.16: Comparison of different kernels in SVM

LSTM

The LSTM model has an Embedding Layer at the front followed by a bidirectional LSTM layer and a Dense layer. The Embedding layer has input shape equal to the longest sentence in the dataset and output shape equal to the dimensions used in the glove embedding used. We are using the pre-trained embeddings of GloVe (Global Vectors)

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 118, 100)          119351500
_________________________________________________________________
bidirectional_2 (Bidirection (None, 256)               234496
_________________________________________________________________
dense_3 (Dense)              (None, 512)               131584
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 1026
=================================================================
Total params: 119,718,606
Trainable params: 367,106
Non-trainable params: 119,351,500
```

Figure 4.17: LSTM Model

as weights of the Embedding layer. As shown in the table below, the accuracy and complexity of the model increases as we increases the size of training data and vocabulary size. We have used two different GloVe embeddings namely:

1. Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors, 822 MB)

2. Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB)

The GloVe Twitter Embedding has a dataset of 27 Billion tokens which contains slangs which are commonly used while writing a tweet. Hence GloVe Twitter is favourable when dealing with Twitter Data.
As shown in the table, we have collected results by varying many different parameters like dataset size, glove embeddings and their dimensions and also some hyperparameter tuning.

Processor	Data Size	Glove Vectors Used		Time Taken (Minutes)	Testing Accuracy
		No. of tokens	Dimensions		
Local GPU	1,00,000	6 Bil.	100	62.5	76.7 %
Colab GPU	1,00,000	6 Bil.	100	2.5	77.4 %
Colab GPU	2,00,000	6 Bil.	100	7.5	78.0 %
Colab GPU	2,00,000	6 Bil.	300	7.5	78.9 %
Colab GPU	3,00,000	27 Bil.	100	25	82.6 %
Colab GPU	4,00,000	27 Bil.	100	26.25	82.7 %

Table 4.2: LSTM Result of Sentiment Analysis.

Result Comparison

We compared SVM and LSTM for dataset sizes 10k to 70k. On each of the dataset size the performance of LSTM was much better compared to SVM. As dataset size increases the performance of both the model increases gradually. From Figure 4.18 , we can conclude that LSTM works better in terms of accuracy.

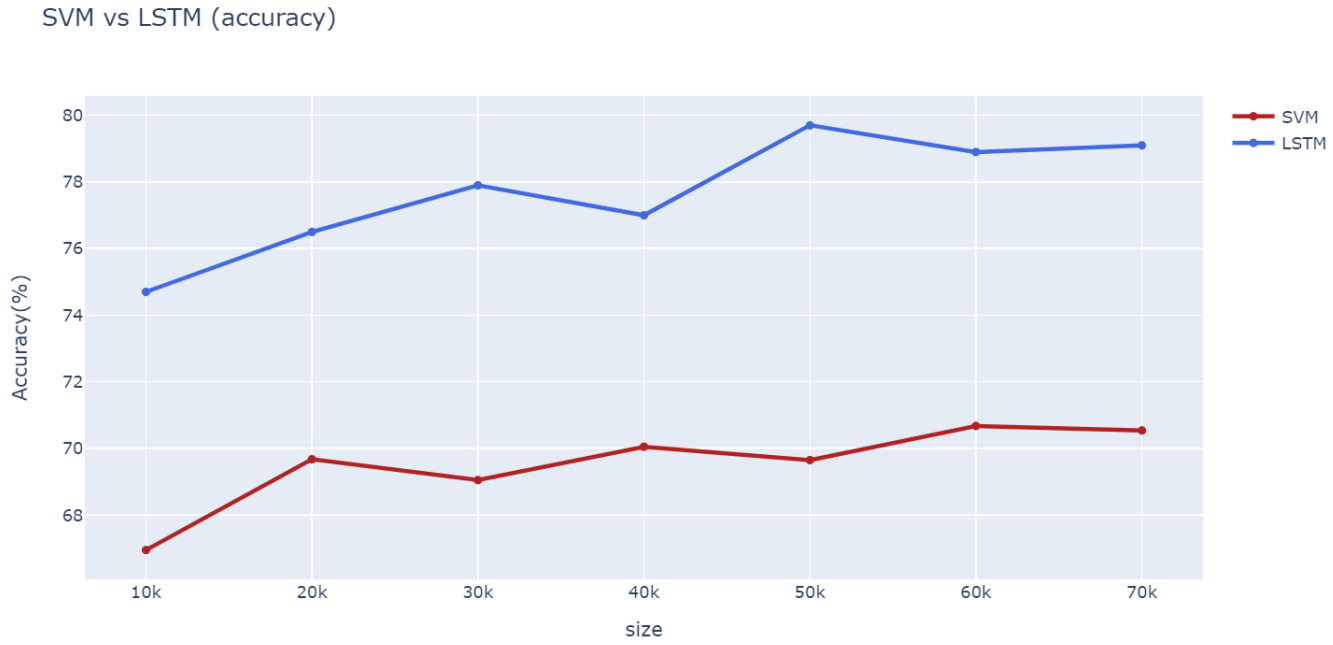

Figure 4.18: SVM vs LSTM w.r.t. Accuracy

4.3.2 Categorisation of Tweets

In order to classify tweets under 5 different categories we have trained a different LSTM model which classifies the tweets into following categories:

- Entertainment
- Education
- Politics
- Sports
- Business

By classifying these tweets into these categories, we will be able to find the cumulative sentiment of a user regarding any of these particular topics. The construction of the LSTM model is similar to the one created for Sentiment Analysis. There are some changes which have been done to accommodate the model for multi-class classification like changing to loss function to "Categorical Cross-entropy" instead of "Binary Cross-entropy". The dataset for this model is prepared by searching the tweets of respective categories from the Twitter API. The Embedding layer uses the pre-trained GloVe Twitter Embedding as shown in Sentiment Analysis.
Moreover, this is also going to be used in finding out about the cumulative sentiment of a community about one of these topics. We applied Random Forest and LSTM with different values of parameters on a dataset of around 43000 tweets of various topics such as education, entertainment, politics, sports and business. Following two tables consist of classification reports of Random Forest and LSTM models.

	Class	Precision	Recall	F1 - score
	ENTERTAINMENT	0.76	0.84	0.80
	POLITICS	0.78	0.83	0.81
	SPORTS	0.83	0.69	0.75
	BUSINESS	0.74	0.65	0.69
	EDUCATION	0.80	0.81	0.81
Accuracy				0.78

Table 4.3: Classification Report of LSTM Model for Multi-Class Text Classification

	Class	Precision	Recall	F1 - score
	ENTERTAINMENT	0.62	0.58	0.60
	POLITICS	0.54	0.84	0.65
	SPORTS	0.74	0.27	0.40
	BUSINESS	0.51	0.27	0.36
	EDUCATION	0.62	0.69	0.65
Accuracy				0.58

Table 4.4: Classification Report of RF Model for Multi-Class Text Classification

After observing Table 4.3 and Table 4.4, Accuracy of LSTM model is better way better than of Random Forest model. Often, we assume both precision and recall demonstrate the model's accuracy. Though that is somewhat correct, each of these terms has a deeper, distinct meaning. Precision is the measurement that notifies us that what proportion of positive identifications was actually correct and recall is the measurement that notifies us that what proportion of actual positives was identified correctly. From Table 4.4, as recall of SPORTS and BUSINESS class is very low, we can state that Random Forest model performs very poor in identifying SPORTS and BUSINESS classes. As we can see from Table 4.3, LSTM model exhibits consistent score of recall and precision throughout mentioned five classes. LSTM model performs great in recognizing all the class with overall accuracy of 78 %. In a nutshell, we can conclude that LSTM works better.

Furthermore, using twitter API, more tweets of five mentioned categories are acquired. We trained LSTM model on the aggregated dataset of approximately 130000 tweets of five mentioned topics. The machine learning model and its performance is as shown in Figure 4.19 and Table 4.5. As shown in Table 4.5, On improving the LSTM model a little, we achieved an overall accuracy of 81%.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 110, 100)          119351500
_________________________________________________________________
bidirectional_2 (Bidirection (None, 256)               234496
_________________________________________________________________
dense_3 (Dense)              (None, 512)               131584
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 5)                 2565
=================================================================
Total params: 119,720,145
Trainable params: 368,645
Non-trainable params: 119,351,500
```

Figure 4.19: LSTM Model For Multi-class Text Classification

	Class	Precision	Recall	F1 - score
	ENTERTAINMENT	0.77	0.76	0.77
	POLITICS	0.79	0.85	0.82
	SPORTS	0.86	0.82	0.84
	BUSINESS	0.83	0.80	0.81
	EDUCATION	0.77	0.79	0.78
Accuracy				0.81

Table 4.5: Classificaion Report of LSTM Model for Multi-Class Text Classification

4.4 Analyzing tweets of the Network

In this module, We have created Web-App to fully utilize the sentiment data that we have calculated and stored in the previous modules. This web-app can be helpful to institute counselors to visualize the mental condition of students. We are proposing analysis on two levels.

4.4.1 User-level Analysis

In this part we are going to red-flag the students who are feeling depressed or low for some continuous period of time. As explained in the previous module,we have acquired tweets and categorised them into different categories(Education, Sports, Entertainment, Business, Politics) and calculated sentiment value. To find the overall sentiment value of each user, we are using cumulative mean(Weighted mean).

$$CumulativeMean = \frac{\sum_{i=1}(W_{\mathrm{i}} * X_{\mathrm{i}})}{\sum_{i=1}(W_{\mathrm{i}})} \tag{4.1}$$

Where :
W_{i} = Weight of the Tweet i
X_{i} = Sentiment Value of Tweet i

The value of cumulative mean will be 0 to 1. We are giving more weightage to latest tweets than the older tweets. So the significance of older tweets in overall cumulative sentiment will be much lesser than the latest one. This will help to determine the recent mood of the user. If the user continuously posts negative tweets, then the cumulative sentiment will decrease gradually.[18]

After finding cumulative sentiment of each user, We have displayed the users in ascending order of their cumulative sentiment. We have also assigned colors to each user according to cumulative sentiments. The more critical user whose sentiment value is constantly decreasing will be assigned color changing from yellow to red (0.5 to 0.0). Inversely, the happier user whose sentiment value is constantly increasing will be assigned colors from yellow to green (0.5 to 1.0).

Figure 4.20: Visualization for Different Sentiment

We have also created functionality to check the activity of individual users over the time. We are showing the user's cumulative sentiment value over time in the graph form.

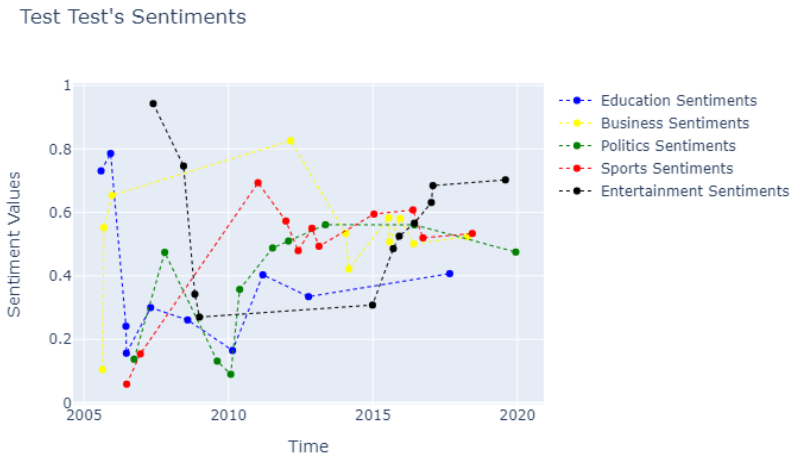

Figure 4.21: Category-wise Cumulative Sentiment's Graph of Particular User

As shown in figure 4.21, We have created traces for each category of tweets. Which will help in finding user's inclination in particular categories. If over the period of time, this inclination changes to lower sentiments, then counsellor may know about him/her feeling down in a particular category. From increasing sentiment, Counsellor can also figure out the user's growing interest in a particular category. We have also provided functionality to check the more recent sentiment value to understand the current status of the user.

4.4.2 Community-level Analysis

We will be creating functionality to see the overall cumulative sentiment of the whole community(In our case Institute) over the different categories (Education, Sports, Entertainment, Politics, Business). This will help the counselor visualize the inclination of the whole user community to different categories. This will help in describing the nature of the whole institute on different events. For example, we will be able to see the change in sentiments of education category during exams. During elections, change in politics category. In short, When some major event occurs in any category, the graph of that certain category will change accordingly. From that change we will be able to figure out how the community is feeling about that particular event.

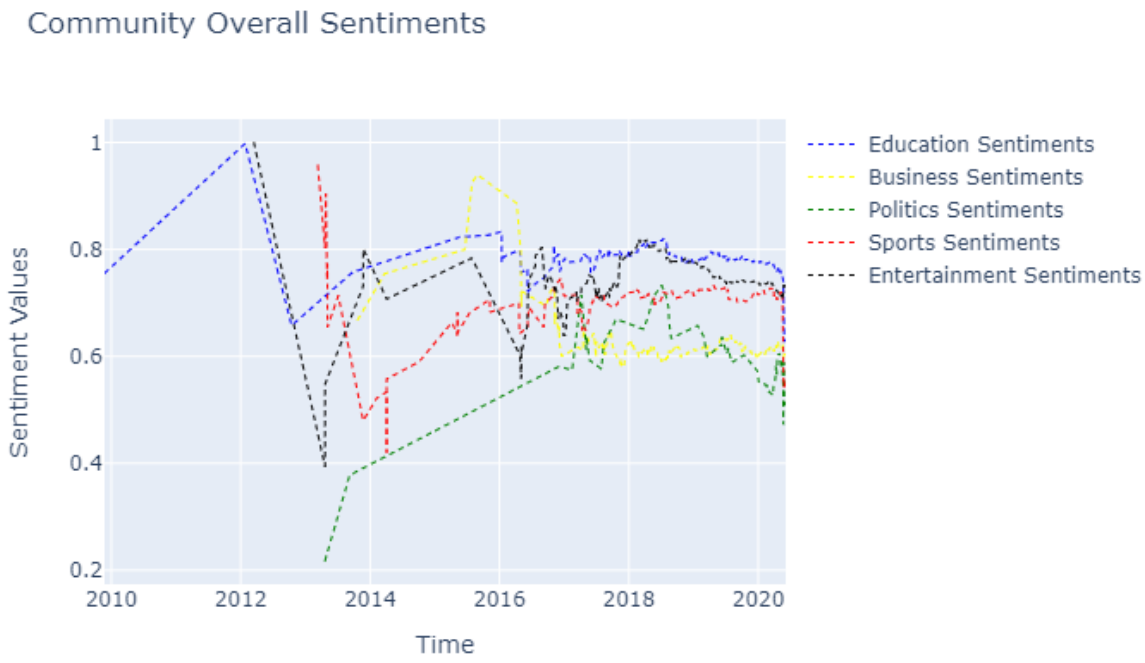

Figure 4.22: Category-wise Cumulative Sentiment's Graph of Community

Chapter 5

Conclusion

In this project, we examined the use of machine learning and graph database (Neo4j) for performing sentiment analysis to calculate and predict abnormal behaviour of users based on their tweets. On providing these data and analysis reports to experts, we can make sure to provide counselling to students who are consistently being flagged down for their negative sentiments. Moreover, with the help of our multi-class classification model, we can get to know the sentiments of users in a particular category too. Multi-class classification also helps in calculating the overall sentiment of a community regarding any of the topics stated before. By extending this, we can calculate the general sentiment of what people are thinking about the current trends of social media.

Bibliography

[1] Walaa Medhat, Ahmed Hassan, Hoda Korashy, "Sentiment analysis algorithms and applications: A survey". *Ain Shams Engineering Journal Volume 5, Issue 4, December 2014, Pages 1093-1113*

[2] Alec Go, Richa Bhayani, Lei Huang, "Twitter Sentiment Classification using Distant Supervision". *CS224N Project Report, Stanford, 2009*

[3] Efthymios Kouloumpis, Theresa Wilson, Johanna Moore, "Twitter Sentiment Analysis: The Good the Bad and the OMG!". *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*

[4] Lija Mohan, Sudheep Elayidom, "Predicting the Winner of Delhi Assembly Election, 2015 from Sentiment Analysis on Twitter Data-A Big Data Perspective". *The International Arab Journal of Information Technology, Vol. 16, No. 5, September 2019*

[5] Fotis Aisopos, George Papadakis, Theodora Varvarigou, "Sentiment Analysis of Social Media Content using N-Gram Graphs". *ICCS, National Technical University of Athens, Greece {fotais, gpapadis, dora}@mail.ntua.gr L3S Research Center, Germany {papadakis}@L3S.de*

[6] K.L.Sumathy, M.Chidambaram, "Text Mining: Concepts, Applications, Tools and Issues – An Overview". *International Journal of Computer Applications (0975 – 8887) Volume 80 – No.4, October 2013*

[7] Dr. S. Vijayarani, Ms. J. Ilamathi, Ms. Nithya, "Preprocessing Techniques for Text Mining - An Overview ". *Dr.S.Vijayarani et al , International Journal of Computer Science & Communication Networks,Vol 5(1),7-16*

[8] Zhou Yao , Cao Ze-wen, "Research on the construction and filter method of stopword list in text preprocessing ". *2011 Fourth International Conference on Intelligent Computation Technology and Automation*

[9] Tanasanee PhienthrakulBoonserm KijsirikulHiroya TakamuraManabu Okumura, "Sentiment Classification with Support Vector Machines and Multiple Kernel Functions ". *International Conference on Neural Information Processing*

[10] Ali Hasan ,Sana Moin ,Ahmad Karim, Shahaboddin Shamshir , "Machine Learning-Based Sentiment Analysis for Twitter Accounts ". *Math. Comput. Appl. 2018, 23(1), 11;*

[11] Maha Heikal, Marwan Torki, Nagwa El-Makky, "Sentiment Analysis of Arabic Tweets using Deep Learning ". *The 4th International Conference on Arabic Computational Linguistics (ACLing 2018), November 17-19 2018, Dubai, United Arab Emirates*

[12] Dipti Mahajan ; Dev Kumar Chaudhary, " Sentiment Analysis Using Rnn and Google Translator". *2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, 2018, pp. 798-802.*

[13] Subarno Pal,Soumadip Ghosh,Amitava Nag, " Sentiment Analysis in the Light of LSTM Recurrent Neural Networks". *International Journal of Synthetic Emotions, January 2018*

[14] Waleed Zaghloul,Sang M. Lee,Silvana Trimi , " Text classification: neural networks vs support vector machines ". *Industrial Management & Data Systems Vol. 109 No. 5, 2009 pp. 708-717*

[15] Jeffrey Pennington, Richard Socher, Christopher D. Manning , " GloVe: Global Vectors forWord Representation. ". *Association for Computational Linguistics, October, 2014.*

[16] Stephen, Jini & P., Prabu. " Detecting the magnitude of depression in Twitter users using sentiment analysis." *International Journal of Electrical and Computer Engineering (IJECE). 9. 3247. 10.11591/ijece.v9i4.pp3247-3255.*

[17] Nadeem, Moin. (2016). "Identifying Depression on Twitter. "

[18] Diveesh Singh, Aileen Wang. "Detecting Depression Through Tweets" *Standford University CA 9430, ;pp.1-9.*

[19] Andrew G. Reece, Andrew J. Reagan, Katharina L. M. Lix, Peter Sheridan Dodds, Christopher M. Danforth & Ellen J. Langer " Forecasting the onset and course of mental illness with Twitter data" *scientific Reports, pp. 1-11, 11 October 2017.*

[20] Nowak J., Taspinar A., Scherer R. "LSTM Recurrent Neural Networks for Short Text and Sentiment Classification." *Artificial Intelligence and Soft Computing. ICAISC 2017. Lecture Notes in Computer Science, vol 10246. Springer, Cham*

[21] Andy Liaw, Matthew Wiener. "Classification and Regression by randomForest" *Merck Research Laboratories, Vol. 2/3, December 2002*

[22] Park, M., Cha, C. & Cha, M."Depressive moods of users portrayed in Twitter." *ACM SIGKDD Workshop on healthcare informatics (HI-KDD) (pp. 1–8) (2012).*

[23] Baoxun Xu, Xiufeng Guo, Yunming Ye, Jiefeng Cheng. "An Improved Random Forest Classifier for Text Categorization" *JournaL of Computers, Vol. 7, No. 12, December2012*

[24] M. Cerrada, et al. "Fault diagnosis in spur gears based on genetic algorithm and random forest" *Mech. Syst. Signal Process. (2015)*

Acknowledgement

We would like to express our sincere gratitude to our guide Dr.Rupa G. Mehta, Professor in Computer Engineering Department, SVNIT, Surat for providing her invaluable guidance, comments, suggestions, constant support and motivation throughout the course of this project work.

We would also like to thank our Head of Department Dr. Mukesh A. Zaveri, Computer Engineering Department, SVNIT, Surat for providing us with the opportunity to work on this project. This project helped us in gaining sufficient knowledge about our course.

seni_visulization

ORIGINALITY REPORT

14%	4%	6%	11%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Liverpool John Moores University Student Paper	1%
2	Submitted to National College of Ireland Student Paper	1%
3	Submitted to University of Ulster Student Paper	1%
4	"Proceedings of ICRIC 2019", Springer Science and Business Media LLC, 2020 Publication	1%
5	www.emerald.com Internet Source	<1%
6	Submitted to College of Engineering Trivandrum Student Paper	<1%
7	brage.bibsys.no Internet Source	<1%
8	Submitted to University of Edinburgh Student Paper	<1%
9	Submitted to University of Paisley	

	Student Paper	<1%
10	export.arxiv.org Internet Source	<1%
11	Submitted to Symbiosis International University Student Paper	<1%
12	Chuanxu Wang, Guofeng Hu, Yun Liu. "Multi-views Action Recognition on Deep Learning and K-SVD", Journal of Physics: Conference Series, 2019 Publication	<1%
13	www.irjet.net Internet Source	<1%
14	Submitted to Indian Institute of Information Technology, Allahabad Student Paper	<1%
15	Submitted to Leeds Beckett University Student Paper	<1%
16	Submitted to Universiti Teknologi Petronas Student Paper	<1%
17	Hualei Dong, Jian Wang, Hongfei Lin, Bo Xu, Zhihao Yang. "Predicting Best Answerers for New Questions: An Approach Leveraging Distributed Representations of Words in Community Question Answering", 2015 Ninth	<1%

International Conference on Frontier of Computer Science and Technology, 2015
Publication

18	Submitted to University of Portsmouth Student Paper	<1%
19	"Social Media Processing", Springer Science and Business Media LLC, 2017 Publication	<1%
20	Ben Aouicha, Mohamed, Mohamed Ali Hadj Taieb, and Abdelmajid Ben Hamadou. "LWCR: multi-Layered Wikipedia representation for Computing word Relatedness", Neurocomputing, 2016. Publication	<1%
21	Submitted to Nanyang Technological University Student Paper	<1%
22	Submitted to University of Warwick Student Paper	<1%
23	Deebha Mumtaz, Bindiya Ahuja. "A Lexical Approach for Opinion Mining in Twitter", International Journal of Education and Management Engineering, 2016 Publication	<1%
24	Ahmad Fathan Hidayatullah, Anisa Miladya Hakim, Abdullah Aziz Sembada. "Adult Content Classification on Indonesian Tweets using	<1%

	LSTM Neural Network", 2019 International Conference on Advanced Computer Science and information Systems (ICACSIS), 2019 Publication	
25	Submitted to University College London Student Paper	<1%
26	rcciit.org Internet Source	<1%
27	pypi.org Internet Source	<1%
28	Submitted to University of Southampton Student Paper	<1%
29	"Biomedical Informatics", Springer Science and Business Media LLC, 2006 Publication	<1%
30	Submitted to Xianjiatong-Liverpool University Student Paper	<1%
31	Submitted to University of Dundee Student Paper	<1%
32	Submitted to Bogazici University Student Paper	<1%
33	Submitted to University of Westminster Student Paper	<1%
34	Sepp Hochreiter, Jürgen Schmidhuber. "Long	<1%

	Short-Term Memory", Neural Computation, 1997 Publication	
35	www.l3s.de Internet Source	<1%
36	Liao, Y.. "A neural network model with bounded-weights for pattern classification", Computers and Operations Research, 200408 Publication	<1%
37	Submitted to University of Sydney Student Paper	<1%
38	cacm.acm.org Internet Source	<1%
39	"Artificial Intelligence and Soft Computing", Springer Science and Business Media LLC, 2017 Publication	<1%
40	Submitted to Savitribai Phule Pune University Student Paper	<1%
41	"Advanced Computing Technologies and Applications", Springer Science and Business Media LLC, 2020 Publication	<1%
42	sutir.sut.ac.th:8080 Internet Source	<1%

43	pt.scribd.com Internet Source	<1%
44	library.seg.org Internet Source	<1%
45	Sandeep Bhongade, Supriya Golhani. "HIF detection using wavelet transform, travelling wave and support vector machine", 2016 International Conference on Electrical Power and Energy Systems (ICEPES), 2016 Publication	<1%
46	Submitted to International University - VNUHCM Student Paper	<1%
47	Yong Shi, Luyao Zhu, Wei Li, Kun Guo, Yuanchun Zheng. "Survey on Classic and Latest Textual Sentiment Analysis Articles and Techniques", International Journal of Information Technology & Decision Making, 2019 Publication	<1%
48	Jini Jojo Stephen, Prabu P.. "Detecting the magnitude of depression in Twitter users using sentiment analysis", International Journal of Electrical and Computer Engineering (IJECE), 2019 Publication	<1%
49	www.mdpi.com	

	Internet Source	<1%
50	Submitted to Mahidol University Student Paper	<1%
51	baadalsg.inflibnet.ac.in Internet Source	<1%
52	Submitted to University of North Texas Student Paper	<1%
53	Submitted to Koc University Student Paper	<1%
54	sites.dartmouth.edu Internet Source	<1%
55	161.200.80.210 Internet Source	<1%
56	eprints.umm.ac.id Internet Source	<1%
57	Submitted to Indian Institute of Corporate Affairs Student Paper	<1%
58	Submitted to United International College Student Paper	<1%
59	Submitted to University of Alabama at Birmingham Student Paper	<1%
	airccse.org	

60	Internet Source	<1%
61	Ghelmar Astoveza, Randolph Jay P. Obias, Roi Jed L. Palcon, Ramon L. Rodriguez, Bernie S. Fabito, Manolito V. Octaviano. "Suicidal Behavior Detection on Twitter Using Neural Network", TENCON 2018 - 2018 IEEE Region 10 Conference, 2018 Publication	<1%
62	kraj3.com.np Internet Source	<1%
63	"Advances in Information Retrieval", Springer Science and Business Media LLC, 2020 Publication	<1%
64	"Progress in Advanced Computing and Intelligent Engineering", Springer Science and Business Media LLC, 2019 Publication	<1%
65	Submitted to ESCP-EAP Student Paper	<1%
66	Lija Mohan, M. Sudheep Elayidom. "Collective tweet analysis for accurate user sentiment analysis - a case study with Delhi Assembly Election 2015", International Journal of Big Data Intelligence, 2018 Publication	<1%

67	Oumaima Oueslati, Erik Cambria, Moez Ben HajHmida, Habib Ounelli. "A review of sentiment analysis research in Arabic language", Future Generation Computer Systems, 2020 Publication	<1%
68	ijcsit.com Internet Source	<1%
69	as.wiley.com Internet Source	<1%
70	Industrial Management & Data Systems, Volume 109, Issue 5 (2009-05-10) Publication	<1%
71	Submitted to International Institute of Information Technology Student Paper	<1%
72	Submitted to The University of Manchester Student Paper	<1%
73	"Innovations in Smart Cities Applications Edition 3", Springer Science and Business Media LLC, 2020 Publication	<1%
74	Submitted to Birkbeck College Student Paper	<1%
75	Fotis Aisopos, George Papadakis, Theodora Varvarigou. "Sentiment analysis of social media	<1%

content using N-Gram graphs", Proceedings of the 3rd ACM SIGMM international workshop on Social media - WSM '11, 2011
Publication

76	"Deep Learning-Based Approaches for Sentiment Analysis", Springer Science and Business Media LLC, 2020 Publication	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		