Early Depression Detection from Social Network Using Deep Learning Techniques

A Thesis

Submitted in partial fulfillment of the requirements for the Degree of Bachelor of Science in Computer Science and Engineering

Submitted by

Farzad Ahmed	**150204004**
Samir Sadek	**150204070**
Sajib Kumar Saha Joy	**150204076**
Rimon Shil	**150204077**

Supervised by

Mr. Faisal Muhammad Shah

Department of Computer Science and Engineering

Ahsanullah University of Science and Technology

Dhaka, Bangladesh

January 2020

CANDIDATES' DECLARATION

We, hereby, declare that the Thesis presented in this report is the outcome of the investigation performed by us under the supervision of Mr. Faisal Muhammad Shah, Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh. The work was spread over two final year courses, CSE4100: Project and Thesis I and CSE4250: Project and Thesis II, in accordance with the course curriculum of the Department for the Bachelor of Science in Computer Science and Engineering program.

It is also declared that neither this Thesis nor any part there of has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Farzad Ahmed
150204004

Samir Sadek
150204070

Sajib Kumar Saha Joy
150204076

Rimon Shil
150204077

CERTIFICATION

This Thesis titled, **"Early Depression Detection from Social Network Using Deep Learning Techniques"**, submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering in January 2020.

Group Members:

Farzad Ahmed	**150204004**
Samir Sadek	**150204070**
Sajib Kumar Saha Joy	**150204076**
Rimon Shil	**150204077**

Mr. Faisal Muhammad Shah
Assistant Professor & Supervisor
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

Professor Dr. Kazi A. Kalpoma
Professor & Head
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to our research supervisor Mr. Faisal Muhammad Shah, Assistant Professor, Department of Computer of Science and Engineering, Ahsanullah University of Science and Technology. Throughout our research he has been so kind, generous with his time and energy, and guided us to the right path to accomplish what we have done in this research work. We would also like to thank Professor Dr. Kazi A. Kalpoma, Head of the Department of Computer Science and Engineering of Ahsanullah University of Science and Technology and all other teachers, lab assistants and our friends for their motivation and moral support. Finally, we would like to thank our parents for their encouragement and support throughout our studies.

Dhaka
January 2020

Farzad Ahmed

Samir Sadek

Sajib Kumar Saha Joy

Rimon Shil

ABSTRACT

Depression is a mental illness that affects over 300 million people worldwide. A person who is depressed suffers from anxiety in day-to-day life, which affects that person in the relationship with their family and friends, leading to different diseases and in the worst-case death by suicide.With the growth of social network, most of the people share their emotion, their feeling, their thought in social media. If their depression can be detected early by analyzing their post, then by taking necessary steps, a person can be saved from depression related diseases or in best case he can be saved from committing suicide. Detecting depression from text is one of the hardest challenges in natural language processing. In this research work, a model has been proposed that can detect depression by analyzing user's posts. Deep learning algorithms were trained using the training data and then performance has been evaluated on the test data. The results acquired from the experiment are then compared to conclude the best approach.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Depression is a mental disorder that affects over 300 million people worldwide. A person who is depressed suffers from anxiety in day-to-day life, which affects that person in the relationship with their family and friends, leading to different diseases and in the worst-case death by suicide. Many obstacles prevent expert care reaching people suffering from depression in time. This leads to a lot of people committing suicide as they did not get proper expert help due to ignorance of the signs of depression owing to lack of awareness of the disease. In today's world, social media has become a platform where people share their views in their day-to-day life and express their feelings. This huge dataset about human feelings on social media gives researchers a big opportunity to analyze whether people are going through depression and whether they might commit suicide.

Therefore, we want to take this opportunity, and propose a model by analyzing previous works done on this area, that will probably help us analyze whether social media users are going through depression and going towards committing suicide.

1.2 Motivation

"Goodbye I'll die soon. (I) am gone," on Saturday, January 12 2019, the founder of Asiwaju Royal Furniture in Ojo, Lagos, Michael Asiwaju, popularly known as Mike Cash, has taken his own life after posting this post on tweet [12]. He was going through depression after being accused of rape. Asiwaju was prominent on the social media platform with about 24,700 followers. The post was the last of several tweets the deceased broadcast, between 5.52pm and 7.35pm, in response to the allegations. He was only 25 years old and later was found that the accusation was false. It was like a death signal to the rest of the world but no professional help was sent in time to save him. There are many other incidents like this on twitter and other Social Medias and there are no accurate mechanisms to attract professionals to save people in time.

Suicide is the second leading cause of death among 15 to 29 years old. Depression is the top cause in committing suicide. Close to 800,000 people, die due to suicide every year. 79 percent of global suicides occur in low- and middle-income countries. So we want to do something to stop this suicide from occurring at this alarming rate.

The following figure 1.1 shows on average, adjusted for age, the annual U.S. suicide rate increased 24 percent between 1999 and 2014, from 10.5 to 13.0 suicides per 100,000 people, the highest rate recorded in 28 years. Due to the stigma surrounding suicide, it is suspected that suicide generally is under reported.

Many respective researchers had done their work in this field but we want to try and hopefully do some further work on these extend method. By further researching we saw that depression Analysis on Social Media is the most effective approach in decreasing suicide rate.

Depression analysis is the process of identifying whether people are going through depression from their textual activity on social media. Identification of depression from social media has been framed as a classification problem in the field of Natural Language Processing (NLP). In this work we study NLP approaches that can successfully extract information from textual data to enhance identification of depression. These NLP approaches perform different feature extraction in order to build document representations.

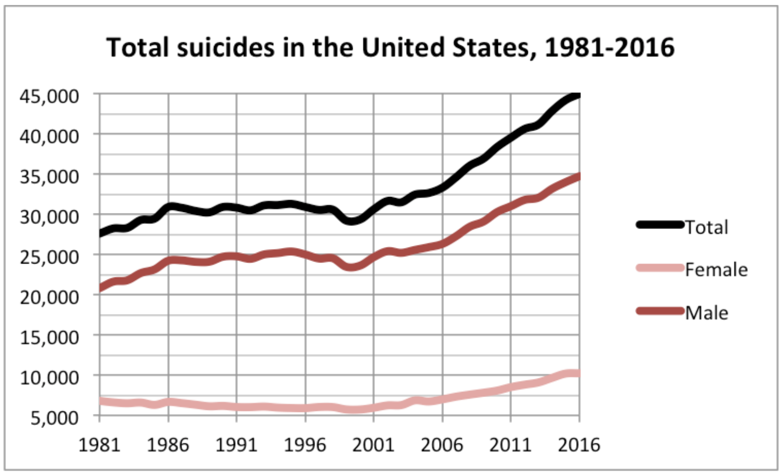


Figure 1.1: Total Suicides in the United States, 1981 to 2016 [1]

1.3 Objective

The main objective in our thesis is to detect depression by analyzing reddit post by reddit users. We accomplished our goal by the proposed methodology we will give ahead . We mainly focus on traditional machine learning approaches and deep learning approaches to detect depression from reddit user post. We not only focus on how correctly we classify the user but also how quickly we can classify a user depressed or not.

1.4 Thesis Contribution

Seventeen related studies were reviewed to understand the current workings of early depression detection on social media users. This thesis work includes both traditional machine learning classifiers and deep learning classifiers to classify tweets. In this research work, all the experiments were performed on the data published for the pilot task Early Detection of Depression in CLEF eRisk 2017 [13]. The organizers collected this data from Reddit, a social media and news aggregation website. We propose models not only traditional machine learning but also deep learning models. To preprocess the data, we have used Natural Language Processing techniques. For feature selection in machine learning, Term Frequency-Inverse Document frequency (TF-IDF), and in deep learning we have used GloVe [14], Word2Vec [15], FastText [16] and MetaData [17]. We have used traditional classifiers such as Naïve Bayes (NB), Support Vector Machine (SVM), Decision Tree (DT), and in deep learning we have used Bidirectional Long-Short Term memory (LSTM) and Dense Neural Network (DNN). Using our models we do not only classify users correctly but also try to classify them early. We used different evaluation matrices such as $ERDE_5$, $ERDE_{50}$ and $F_{latency}$ to find how quickly each model can predict depressed users.

1.5 Thesis Structure

- **Chapter 1 Introduction:** This chapter gives a short description of our proposed work, motivation behind choosing early detection of depression on reddit users and the objective of our work,
- **Chapter 2 Literature Review:** In this chapter, we have reviewed some recent works that our similar to our research. We have summarized some of the recent works and compared them in tabular approach. This chapter contains two major parts. One with machine learning approaches and another with deep learning approaches that are helpful for our work.

- **Chapter 3 Background Study:** This chapter contains some background studies that are important for our research purpose. Details on natural language processing, text mining, feature selection and machine learning classifiers. Also, we have discussed on various deep learning algorithms.
- **Chapter 4 Proposed Model:** In this chapter, we have briefly discussed the steps we took to achieve the detection of depressed user. Step by step architecture of Dataset collection, how preprocessiong was done, machine learning classifiers and deep learning model have been described.
- **Chapter 5 Experimental Results:** The result that we got from doing simulation of our models were shown in the chapter. We show the results of traditional approach and deeplearning approach and make comparison with existing works for performance evaluation of our model.
- **Chapter 6 Conclusion and Future Works:** The last chapter of our thesis book contains the conclusion of our work, limitations and scope of our methodology and possible future works.

1.6 Summary

In this chapter, we have given a short outline of our goal, discussed the motivation behind choosing user based depression detection, structure of our thesis and contribution. Therefore, this chapter gives an overall idea of the thesis.

Chapter 2

Literature Review

2.1 Overview

A lot of people share their thoughts about depression on social media. Many work has been done on depression detection on text obtained from social media. In this chapter, we are going to review some of the research articles we have studied and compare their result in tabular form.

2.2 Traditional Machine Learning and Deep Learning Approaches

Dulac-Arnold et al. tried to address a (sequential) text classification task of user based depression as a Markov Decision Process (MDP) with three actions: read (the next sentence), classify and stop [18]. A Support Vector Machine (SVMs) classifier were used which were trained to classify depressive or non depressive post based on the current state, s. The state s was represented by a feature vector, $\Phi(s)$, which contains the information about the term frequency and inverse-document frequency (tf-idf) representations of the current and previous sentences, and the categories assigned so far. Also use of MDP seems very good theoretically but using SVM along with $\Phi(s)$ shows that it does not tell the reason for classifying the input but also the reasons behind its decision to stop early. This paper has a F1 score of 90%.

Wang X, et al. had tried to detect depression from Sina Micro- blog (a social network service like Twitter) using features of depressed users derived from psychological observations [19]. At first, the authors preprocessed the dataset from Sina Micro- blog using sentence segmentation and word segmentation. Then a sentiment analysis method was constructed utilizing vocabulary and man-made rules to calculate the depression inclination of each micro-blog. The author used TF (Term Frequency) and IDF (Inverse Document Frequency) for keyword extraction, bag of words, linguistic rules and position of sub-sentence to detect the polarity of a sentence (positive or negative). The authors then constructed a depression detection model based on the features of depressed users derived from psychological research like use of words and use of emoticons. Three classifiers are used: NB (Naïve Bayes), DT (Decision Tree) and Rule Based. Then an application was developed within the proposed model for mental health monitoring online. The paper got a precision of 80%.

Munmun De Choudhury, et al. had tried to measure levels of depression of people who are using social media like twitter, facebook to share their opinion, emotion, activity [20]. Authors had used crowd sourcing methodology to build a large corpus of postings on Twitter that have been shared by individuals. Next, they developed a probabilistic model trained on this corpus to determine if posts could indicate depression. The designed model leverages signals of social activity, emotion, and language manifested on Twitter. Using that model, they introduced a social media depression index that may serve to characterize levels of depression in populations. They used a supervised learning setup for classifying whether or not a given post is depression-indicative. They represented Twitter posts as vectors of the features (e.g., emotion, time, n-grams, style, engagement features). Before feature extraction, the posts are lowercased, and numbers are normalized into a canonical form. Finally the posts are tokenized. After feature extraction, features that occur fewer than five times are removed in the step of feature reduction. Then they executed classification algorithm that is standard Support Vector Machine classifier to predict performance.

Nguyen at al. experimented with a wide range of NLP techniques in Live Journal social networking service to transform the text into a high-dimensional space and captured its topics and mood posted on online blogs [21]. He aimed to present a good predictive validity in the depression classification between the clinical and control groups. Affective features, Mood tags, LIWC and topics were used as features. Traditional Machine Learning classifiers were used.

Tsugawa et al. predicted depression from Twitter data in Japanese sample where he showed that the feature based on a topic modelling are useful in the tasks for recognizing depressive and suicidal users [22]. They used features such as bag of words, ratio of positive and negative words, posting hour, tweet frequency, mention url. They also used Support Vector Machines to classify depressed users with an accuracy of 66%.

Almeida et al. worked on the CLEF eRisk Pilot Task 2017 where a methodology was proposed where bigram tigrams were used to early predict depressed users [23]. It uses Information Retrieval(IR) that is extracting similar information and Supervised Learning (SL) using several different classifiers of deep learning. It F1 score of 0.53 and ERDE 50 of

12.29%.

Aldarwish MM, Ahmad HF had tried to detect depression level from user generated content of social media (Live Journal, Twitter, Facebook) using RapidMiner, a model consisting a number of processes to test all the classifiers, SVM (Support Vector Machine) Classifier, Naïve Bayes classifier and RNN [24]. Dataset is collected, using RapidMiner, the data is Tokenized, Filter Stop-words, Transform Cases, and Stemmed. RapidMiner buit-in stop-word list is used for Stop-word and Porter stemming algorithm for stemming. RapidMiner has two datasets, one for detecting depression and categorizing into the nine depression category (Sadness, Loss of Interest, Appetite, Sleep, Thinking, Guilt, Tired, Movement and Suicidal ideation). The second dataset consists of the patient SNS posts and it is changed for every individual to test the prediction of the model. This RapidMiner is used for both training and testing. The Apply Model in RapidMiner which connect the test dataset and training dataset to give us the final result of the prediction.

Bentoni at el. demonstrated the effectiveness of multi-task learning (MTL) models on mental health disorders with a limited amount of target data [25]. He used feed-forward multi-layer perceptrons and feed-forward multi-task models trained to predict each task separately as well as to predict a set of conditions simultaneously. They experimented with a feed-forward network against independent logisitic regression models to test if MTL would have performed well in the domain. They got a recall of 0.85.

Hassan, et al. had used social media data to diagnose depression using machine learning techniques and by extracting emotions from text [26]. The authors presented a procedure of teaching a machine to analyze the various grammatical manners, cultural variations, extract emotions and find sentiment and meaning behind those words using machine learning techniques. In the proposed methodology sentiment classification are done in different categories such as strong positive, positive, neutral, negative, strong negative. They used a voting approach amongst the classifiers to find out which label gets highest votes. Three classifiers are used. These are SVM (Support Vector machine), NB (Naïve Bayes), ME (Maximum Entropy) as inner learners and SVM achieved highest accuracy while predicting depression in documents.

Farig Sadeque, et al. had tried to detect early depression from user's post on reddit using the techniques employed for the University of Arizona team's participation [27]. The authors used depression lexicon and concept unique identifiers from Unified Medical Language System as features. They used the unigrams in depression lexicon, which has a high probability in appearing in depressive posts. Metamap was used for Unified Medical Language System features. They used both both sequential and non-sequential learning models for the prediction task. For the non-sequential model, they used a support vector machine. For sequential model, Recurrent Neural Network and Ensemble were used. The limitation of this paper was that it used preexisting lexicon instead of making a domain-specific depression lexicon. Also a better strategy would have been to start making predictions after observing some threshold n posts, allowing them to predict early for users with many posts, while waiting until they have more information for users with few posts instead of starting from the beginning.

Farig Sadeque, et al. also had tried tried to identify several issues with the currently popular Early Risk Detection Error (ERDE) metric [11]. The authors proposed a latency-weighted F1 metric that addresses these concerns. Then they applied this evaluation to several models from the recent eRisk 2017 shared task on depression detection, and show that their proposal can measure better. They proposed a general approach for improving the latency of detection models based on checking the consistency of a model's prediction. They performed experiments on the data published for the pilot task Early Detection of Depression in CLEF eRisk 2017. The organizers collected this data from Reddit and used four feature sets i.e. Words, Depressive Words, Depressive Embed and MetaMap (a highly configurable program developed by Dr. Alan Aronson at the National Library of Medicine (NLM) to map biomedical text to UMLS). They have used boths non-sequential (Support Vector Machine) and sequential model (neural network) to classify data. For risk window of size 23, using depressive word and metamap as features with GRU(Gated recurrent unit), it got $ERDE_5$ of 15.2, $ERDE_{50}$ of 11.5, F_1 of 44.4 and $F_{latency}$ of 32.7.

Loyola et al. proposed a methodology where it considers the decision of when to classify a user as depressed or not to be learned on its on [28]. The methodology trains two Support Vector Machines (SVMs), one to make class prediction and another to decide when to stop early but it the reason for it is hidden. The feature used here was bag-of-words and term frequency weighing scheme. Also using SVMs for this kind of problem is very costly as the document-term matrix has to be rebuilt form scratch every time new content is added. It has a F1 score of 68.23%.

Sayanta Paul, et al. had tried to focus on early detection of signs of depression or anorexia using posts or comments over social media [29]. They used Reddit posts. The authors used two different corpora, using the posts and comments over Reddit. The paper presents different machine learning techniques and analyze the performance for early risk detection of anorexia or depression. The techniques involve various classifiers and feature engineering schemes. The simple bag of words model has been used to perform ada boost, random forest, logistic regression and support vector machine classifiers to identify documents related to anorexia or depression using metamap, a tool to extract biomedical concepts. Therefore, the classifiers have been implemented using bag of words features and metamap features individually and subsequently combining these features.

Tyshchenko et al suggested categorizing the stop words and adding LIWC-like word categories as an extra feature to an already designed method (BOW+TFIDF+LIWC). In addition, he applied multiple feature combinations to increase the performance using Convolutional Neural Networks (CNN) which consist of neurons with learnable weights and disagree in terms of their layers. CNNs are very similar to simple feed-forward neural networks and state of the art method in the text and sentence classification tasks [30]. It got accuracy of 0.7857 by using CNN+GloVe model.

Maupom et al. implemented a system(uni-gram,bi-gram,tri-gram) based on the topic extraction algorithm and simple neural networks extracting 30 latent topics in an unsupervised manner. His result shows that the limited number of users greatly hindered the predicting power of the P MLP [31]. It got precision of 0.67, $ERDE_5$ of 9.81 and $ERDE_1$0 of 9.08.

Hongfei Lin et al use Support Vector Machines (SVMs) and Multiplayer Perceptron (MLP) to classify reddit, Facebook and Twitter users using feature such as bigram, LIWC and LDA [32]. Although it got an accuracy of 80% and 0.80 F1 scores, the downside of this paper is that it did not work on early detection of users so it will not be able to detect depressed users at the correct time.

Zinken et al. estimated the advancement of syndromes by using the psychological topicality of processing structures .They deduced that a written text might barely disagree in its word behavior but disagree in its processing structure, exceptionally in the making of relationships between the events and also establish that in the text written by depressed individuals there was a decreased use of complex syntax in comparison to non-depressed ones. [33]

Table 2.1 shows different approaches that had been done for detecting depression.

Table 2.1: Comparison of Traditional Machine Learning and Deep Learning Approaches

Paper ID	Year	Learning Model	Feature	Results
[18]	2011	SVM	TF-IDF	90% average F1 score
[19]	2013	NB (Naïve Bayes), DT (Decision Tree)	TF and IDF, Bag-of-Words, Linguistic rules	80% Precision
[20]	2013	SVM	Emotion, Time, N-Grams, Style, Engagement	
[21]	2014	Different traditional machine learning classifiers	Affective features, Mood tags, LIWC and topics	

[22]	2015	SVM	Bag-of-Words, Ratio of positive and Negative Words, Posting hour, Tweet frequency, Mention url	66% accuracy
[23]	2017	RNN, GRU	Bigram, Tigrams	53% average F1 Score and 12.29% ERDE50
[24]	2017	SVM, Naïve Bayes and RNN		
[25]	2017	Feed-forward multi-layer perceptrons and feed-forward multi-task models		85% recall
[26]	2017	SVM, NB (Naïve Bayes), ME (Max-imum Entropy)	Emotion, Sentiment	
[27]	2017	SVM, RNN	Depression lexicon and concept unique identifiers from Unified Medical Lan-guage System	
[11]	2018	SVM, GRU	Words, Depressive Words, Depressive Embed and MetaMap	15.5% ERDE 5, 11.5% ERDE 50,44.4% F1 and 32.7% F-Latency
[28]	2018	SVM	Bag-of-Words,TF	68.23% average F1 score
[29]	2018	Logistic Regression and SVM	Bag-of-Words,MetaMap	
[30]	2018	CNN+GloVe	Bag-of-Words+TF-IDF+LIWC	78% accuracy
[31]	2018	Dense Neural Network	Uni-Gram, Bi-Gram, Tri-Gram	67% precision, 9.81% ERDE5 and 9.08% ERDE10
[32]	2019	SVM,Multiplayer Perceptron (MLP)	Bigram, LIWC	80% accuracy and 80% F1 score

2.3 Summary

In this chapter we described about different papers we studied to know the current works that has been on Depression Detection. Descriptions about what features and methods were used by different authors to predict depression from social media.

Chapter 3

Background Study

3.1 Overview

In this chapter we have mentioned some topics that are essential to know before starting with depression detection using machine learning and deep learning techniques. Here, section 3.2 gives an idea about depression and how it is related to our everyday life. Section 3.3 describes about depression analysis. Section 3.4 explains natural language processing and how that performed. Section 3.5 describes machine learning and some of its method such as Naïve Bayes, Decision tree, Support vector machine (SVM). Section Section 3.6 focuses on deep learning methods such as Artificial Neural Network (ANN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and Long Short-term Memory (LSTM). Section 3.7 shows the comparison between traditional machine learning and deep learning. Section 3.8 describes about feature selection and some of the techniques of feature selection such as TF-IDF, Meta Features. Section 3.9 describes about vector representation of word using Embedding techniques and section 3.10 shows the comparion between them.

3.2 Depression

Depression is a disorder that can affect everything you do in your daily life. It is not something you can quickly recover from, like a cold or stomach bug. Many people with depression think they are just feeling sad, and that it will go away with time. For some people, it does just that. But for 15 million others, depression is a constant feeling that does not go away on its own. These people are the ones who may benefit from extra support and help. In fact, the World Health Organization (WHO) found that depression is the second greatest reason for disability in the world. Unfortunately, only 10% of these people ever receive any effective treatment [34].

Depression can impact every area of your life, including but not limited to how you sleep and eat, your education and career, your relationships, health, and concentration. Individuals suffering from depression often also have comorbid disorders, such as alcohol and drug abuse or other addictions. Depression doesn't just occur for an individual in vacuum; it can affect your friends, family, co-workers, and everyone around you. In addition, depression may impact how you perform at work or your levels of concentration, so it can negatively affect productivity. Leaving depression untreated can lead to many other complications in one's personal and professional life. This is why it is so important to seek out help, not just for relationships and work, but for your own sake [34].

3.3 Depression Analysis

Depression essentially relates to mental state of a person. Depression analysis from text refers to the practice of applying Natural Language Processing and Text Analysis techniques to identify and extract subjective information from a piece of text. A person's opinion or feelings are for the most part here. Which means to accurately analyze an individual's state from a piece of text can be extremely difficult. We are essentially looking to get an understanding of the mental state of a writer with respect to a topic in a piece of text and its polarity whether it represents depression or not.

3.4 Natural Language Processing (NLP)

Natural Language Processing, usually shortened as NLP, is a branch of artificial intelligence that deals with the interaction between computers and humans using the natural language. The ultimate objective of NLP is to read, decipher, understand, and make sense of the human languages in a manner that is valuable. Most NLP techniques rely on machine learning to derive meaning from human languages. Syntax analysis is the main technique used to complete Natural Language Processing tasks. Here is a description on how it can be used.

3.4.1 Syntax Analysis

Syntax refers to the arrangement of words in a sentence such that they make grammatical sense. In NLP, syntactic analysis is used to assess how the natural language aligns with the grammatical rules. Computer algorithms are used to apply grammatical rules to a group of words and derive meaning from them. Here are some syntax techniques that can be used.

- **Tokenization**- Tokenization is breaking up a whole sequence of strings into pieces such as words, keywords, symbols, phrases and other elements called tokens. Tokens are individual words, phrases or even whole sentences. Some characters are discarded during tokenization such as punctuation marks. The tokens become the input for another process like parsing and text mining. Example, Sentence: I am having a very good day. Word tokens: "I", "am", "having", "a", "very", "good", "day" [35].

- **Parts of Speech (POS) tagging**- Parts of speech tagging is process which identifies the parts of speech of a given word. This is commonly known as POS tagging. Parts of speech include noun, pronoun, adjectives, adverbs, conjunctions and their individual sub categories. A POS tagger does as below Example: Word: "Good", POS-Tag: "Adjective" Word: "I", POS-Tag: "Pronoun" [35].
- **Stemming**- Documents have different forms of a word because of grammatical reasons i.e., Playing, Played, Plays. As computer doesn't understand this difference, it treats them as different words while converting then in numerical values for analysis. In such cases stemming plays the major role of identifying the different form of base word. Example: – Playing – Played – Playable – Plays – Play [After Stemming] [35].

3.5 Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves. The process of learning begins with the observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly [36].

Computers were originally designed to follow algorithms. An algorithm is simply a series of steps coded in a computer language. Skilled computer programmers would liaise with process experts to map business operations into a flowchart diagram which could then be implemented as a computer program. A flowchart explicitly positions the tasks that should be performed, in the order that they need to be executed, together with any decisions that need to be made along the way. Flowcharts are great at modelling repetitive, predictable processes where decisions are made on unambiguous data. These systems are said to be deterministic.

But not all processes follow clear, unchanging rules, and most decisions in the real world do not lead to a single unambiguous answer. Machine Learning systems are probabilistic: tasks are executed and decisions are made on incomplete information and outcomes are assigned probabilities of being correct. Machine learning is suited to problems involving classification (dividing objects into two or more classes), regression (discovering relationships between variables) and clustering (grouping objects by similar characteristics) [36]. This leads to uses such as:

- Recognizing patterns
- Extracting insight
- Discovering anomalies
- Making predictions

3.5.1 Classification

In machine learning, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. Some examples of classifiers are Naïve Bayes, Decision Tree, K-Nearest Neighbor, Support Vector Machine etc.

- **Naive Bayes Classifier :** It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability. Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods. The formula it uses is:

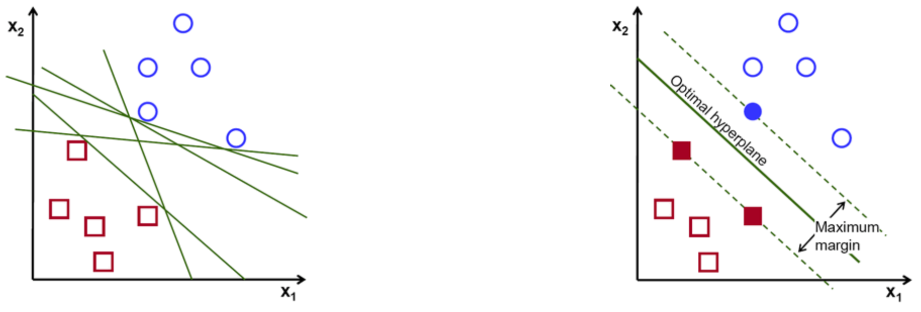
  $$P(C|X) = P(X|C)P(C)/P(X) \tag{3.1}$$

  P(C|X) is the posterior probability of class (Ci, target) given predictor (X, attributes). P(C) is the prior probability of class. P(X|C) is the likelihood which is the probability of predictor given class. P(X) is the prior probability of predictor.

- **Decision Tree Classifier:** Decision tree builds classification or regression models in the form of a tree structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches and a leaf node represents a classification or decision. The top-most decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

- **K-Nearest Neighbor:** The K-Nearest-Neighbors algorithm is a classification algorithm, and it is supervised: it takes a bunch of labelled points and uses them to learn how to label other points. To label a new point, it looks at the labelled points closest to that new point (those are its nearest neighbors), and has those neighbors vote, so whichever label the most of the neighbors have is the label for the new point (the "K" is the number of neighbors it checks). The distance it uses to check the nearest neighbors are Euclidian, Manhattan, Minkowski or Weighted.

- **Support Vector Machine:** The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N: the number of features) that distinctly classifies the data points.



(a) Possible Hyperplane (b) Optimal Hyperplane

Figure 3.1: Hyperplane [2]

We will experiment different classifier to see which performs better for our case. Then we will choose the best among them.

3.6 Deep Learning

Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones. Deep learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. The word 'deep' in 'deep learning' refers to the number of layers through which the data is transformed. Learning can be supervised, semi-supervised or unsupervised. In deep learning, each level learns to transform its input data into a slightly more abstract and composite representation. Deep learning is focused on training multiple layered neural network models using the back propagation algorithm [4].

3.6.1 Artificial Neural Network (ANN)

An artificial neuron network (ANN) is a computational model inspired by the structure and functions of biological neural networks that constitute human brain. Such models "learn" to perform tasks by considering examples, which does not require any task-specific rules to be programmed [4].

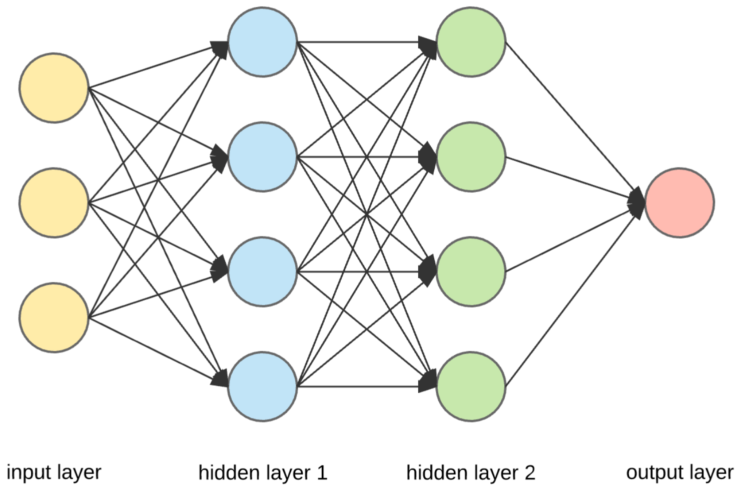


Figure 3.2: Artificial Neural Network [3]

3.6.2 Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) is a class of artificial neural network that involves directed cycles in memory and uses sequence data. RNN can use their internal memory to process sequences of input which makes them applicable for the tasks such as unsegmented, connected handwriting recognition, speech recognition.

RNN remembers the past and it's decisions are influenced by what it has learnt from the past. Traditional neural network also known as feed forward neural network, remembers things too, but they remember things they learnt during training. While RNNs learn similarly while training, in addition, they remember things learnt from prior inputs while generating outputs. It's part of the network.

In a feedforward neural network signal flows in only one direction, input to output, one layer at a time. In a RNN the output of a layer is added to the next input and fed back to the same layer, which is typically the one layer in the entire network. Unlike feed forward net the recurrent net can receive a sequence of values as input and it can also produce a sequence of values as output.

RNNs are trained using backpropagation through time, which reintroduces the vanishing gradient problem. The problem of vanishing gradient is exponentially worse for an RNN. The reason for this is that each time step is equivalent to the entire layer in a feedforward network. So, training an RNN for a 100 times steps is like training a 100 layers feedforward network. This leads to exponentially small gradients and a decay of information through time. The most popular way to address this problem is gating which takes the output of any time step and the next input, and performs a transformation before feeding the result back into the RNN.

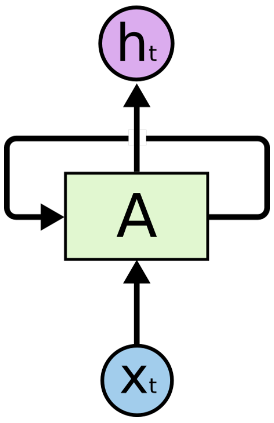


Figure 3.3: Recurrent Neural Network [4]

3.6.3 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is an artificial recurrent neural network architecture which was developed to deal with the exploding and vanishing gradient problems that can be encountered when training traditional RNN. [4]

The most common architecture of LSTM unit is composed of a cell (the memory part of the LSTM unit) and three "regulators" which are called gates, of the flow of information inside the LSTM unit: an input gate, an output gate and a forget gate. Some variations of the LSTM unit do not have one or more of these gates or maybe have other gates. For example, Gated Recurrent Units (GRUs) do not have an output gate.

Intuitively, the cell is responsible for keeping track of the dependencies between the elements in the input sequence. The input gate controls the extent to which a new value flows into the cell, the forget gate controls the extent to which a value remains in the cell and the output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit. The activation function of the LSTM gates is often the logistic function.

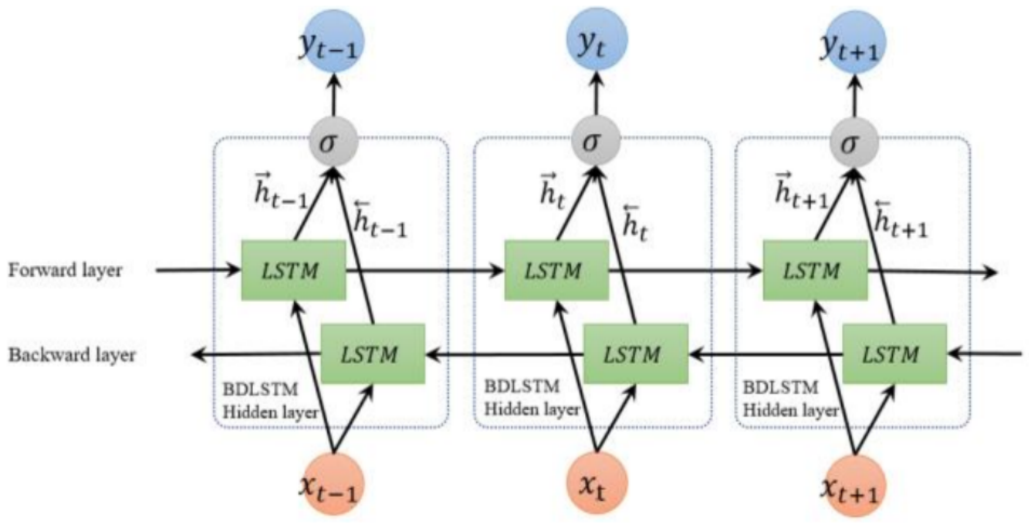


Figure 3.4: Long Short-Term Memory (LSTM) [5]

There are connections into and out of the LSTM gates, a few of which are recurrent. The weights of these connections, which need to be learned during training, determine how the gates operate.

3.6.4 Dense Neural Network

The name suggests that layers are fully connected (dense) by the neurons in a network layer. Each neuron in a layer receives an input from all the neurons present in the previous layer—thus, they're densely connected.

In other words, the dense layer is a fully connected layer, meaning all the neurons in a layer are connected to those in the next layer. [6]

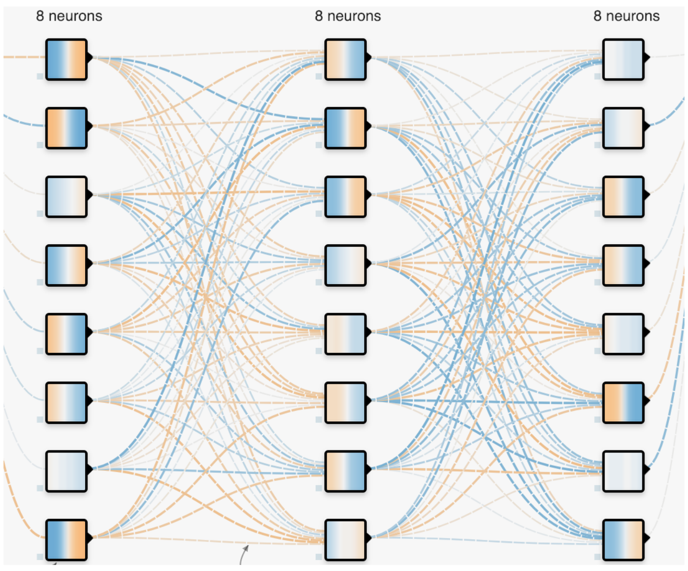


Figure 3.5: Dense Neural Network [6]

3.7 Traditional Machine Learning vs. Deep Learning

Over past several years deep learning has outperformed traditional machine learning for its superiority performance on wide variety of task including natural language, speech, vision, games. Still there are a few advantages of using machine learning in some specific situation.

Traditional machine leaning uses different types of automated algorithm that learn to model functions and predict future actions for data while deep learning use neural network that pass the data through many processing layer to interpret data features and relationship.

Data dependency is most important to estimate which approach is going to give higher accuracy, when the dataset is too large it is undoubtedly beneficial to use deep learning but in some cases huge data may not be available, for such case traditional approach gives better output.

There is no need to feature engineering for deep learning, it directly passes the data to neural net. But to get a decent accuracy feature must be selected carefully for machine learning.

Traditional machine learning works well in real time while for deep learning it is quite challenging. [4]

3.8 Feature Selection

Feature selection is one of the most important process in machine learning or deep learning. The better the feature learning process, they better the model will be. Some commonly used feature selection process is described here.

3.8.1 TF-IDF

TF-IDF stands for term frequency-inverse document frequency, and the TF-IDF weight is a weight which is generally used in text mining and information retrieval [67]. This weight is a statistical measure carrying the importance of a word in a collection of document or corpus. The importance increment of a word is proportional to the number of times the word appears in the document but is offset by the frequency of it in the corpus. Often time search engines use variations of TF-IDF weighting scheme as a central tool in scoring and ranking a document's relevance given by a user query. Summing the TF-IDF weight for each query term is one of the simplest ranking functions, there are many more sophisticated ranking functions that are variants of this simple model. TF-IDF can be successfully used for stop-words filtering in various subject fields including text summarization and classification.

- **TF: Term Frequency**, which computes how frequently a term occurs in a document. Since every document is of different length, it is expected that a term or word would be used much more times in long documents than the shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:

$$TF(t) = \frac{NumberOfTimesTermtAppearsInADocument}{TotalNumberOfTermsInTheDocument} \tag{3.2}$$

- **IDF: Inverse Document Frequency**, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:

$$IDF(t) = \log_e \frac{TotalNumberOfDocuments}{NumberOfDocumentsWithTermtInIt} \tag{3.3}$$

So, TF-IDF is the combination of both TF and IDF. We get the TF-IDF by multiplying both of them as TF-IDF returns only one value.

$$TF - IDF(t) = TF(t) \times IDF(t) \tag{3.4}$$

3.8.2 Lexicon Based Feature Selection

Lexicon is more likely the keyword based method. In Lexicon, the specific words are given that are more important than the other words in the sentence. For each sentence, those important words are figured out and they decide what will be the class for that sentense.

3.8.3 Meta Data Features

Meta Data Features are designed to extract general properties which is able to characterize datasets. The meta-feature values should provide relevant evidences about the performance of algorithms. Thus, these measures must be able to predict, with a low computational cost, the performance of the algorithms under evaluation. [37]

In case of text, Meta Features can be noun, pronoun, verb, length of text, lexicon based words and so on. What features will be taken is completely dependend on the application you are working on. For depression detection, depressive words, anti depressants , ' I ' and so on can be considered as meta features.

3.9 Embedding

Embedding mainly convert the high-dimensional vectors into low-dimensional space. A word embedding is a class of approaches for representing words and documents using a dense vector representation. It is an improvement over the traditional bag-of-word model encoding schemes where large sparse vectors were used to represent each word or to score each word within a vector to represent an entire vocabulary. These representations were sparse because the vocabularies were vast and a given word or document would be represented by a large vector comprised mostly of zero values. Instead, in an embedding, words are represented by dense vectors where a vector represents the projection of the word into a continuous vector space. The position of a word within the vector space is learned from text and is based on the words that surround the word when it is used. The position of a word in the learned vector space is referred to as its embedding. Two popular examples of methods of learning word embedding from text include Word2Vec, GloVe.

- **GloVe** [14] has been trained on 27 billion tweets of 50, 100 and 150 dimension. The similarity metrics used for nearest neighbor evaluations produce a single scalar that quantifies the relatedness of two words. This simplicity can be problematic since two given words almost always exhibit more intricate relationships than can be captured by a single number. For example, "man" may be regarded as similar to "woman" in that both words describe human beings; on the other hand, the two words are often considered opposites since they highlight a primary axis along which humans differ from one another.

  In order to capture in a quantitative way, the nuance necessary to distinguish man from woman, it is necessary for a model to associate more than a single number to the word pair. A natural and simple candidate for an enlarged set of discriminative numbers is the vector difference between the two word vectors. GloVe is designed in order that such vector differences capture as much as possible the meaning specified by the juxtaposition of two words. The underlying concept that distinguishes man from woman, i.e. sex or gender, may be equivalently specified by various other word pairs, such as king and queen or brother and sister. To state this observation mathematically, we might expect that the vector differences man - woman, king - queen, and brother - sister might all be roughly equal. This property and other interesting patterns can be observed in the above set of visualizations.

- **Word2Vec** [15] is a method to construct such an embedding. It can be obtained using two methods (both involving Neural Networks): Skip Gram and Common Bag Of Words (CBOW). CBOW method takes the context of each word as the input and tries to predict the word corresponding to the context. The Skip-gram model archi-

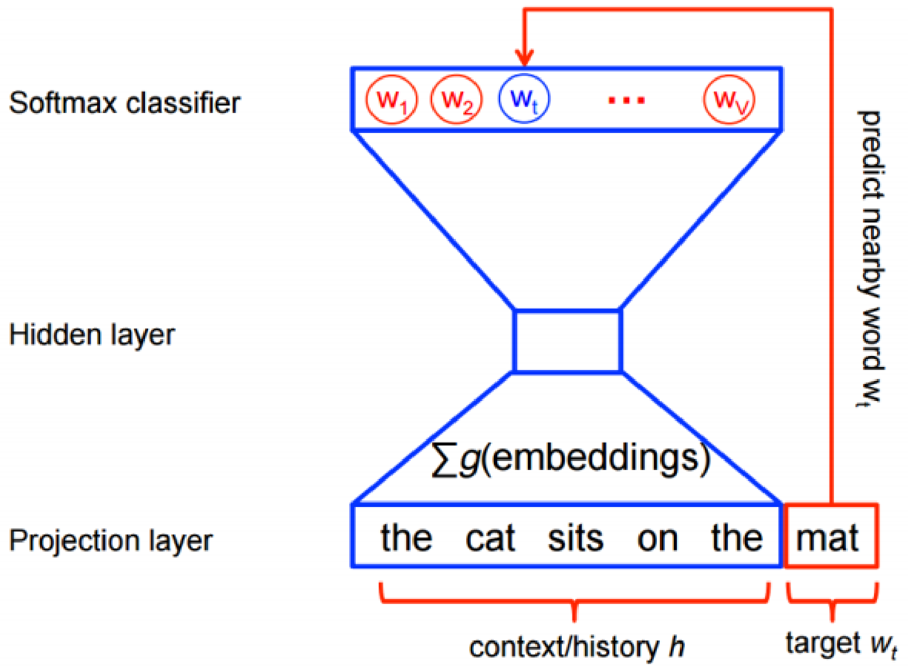


Figure 3.6: GloVe Model Architecture [7]

tecture usually tries to achieve the reverse of what the CBOW model does. It tries to predict the source context words (surrounding words) given a target word (the center word).Both have their own advantages and disadvantages. Skip Gram works well with small amount of data and is found to represent rare words well. On the other hand, CBOW is faster and has better representations for more frequent words.

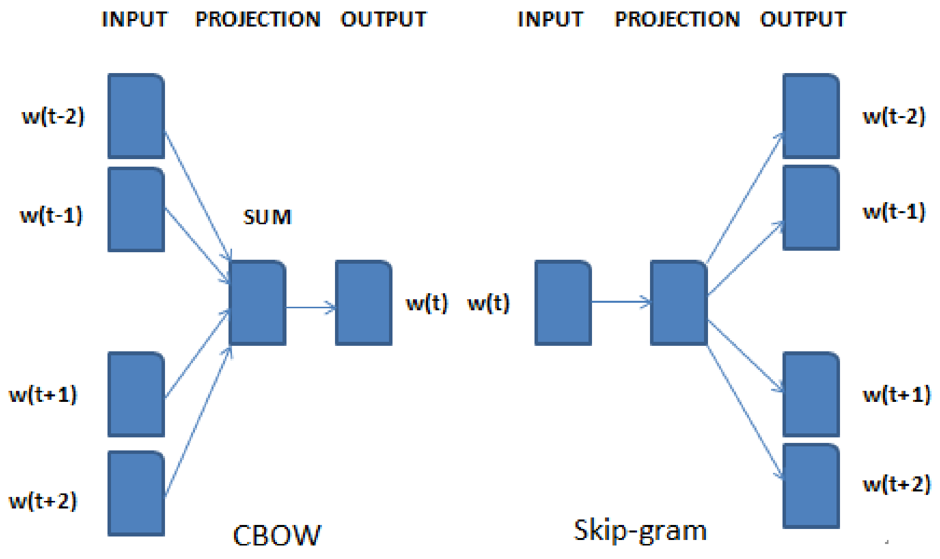


Figure 3.7: Word2Vec Model Architecture [8]

- **FastText** [16] is another method to construct such an embedding. It is an open-source, free, lightweight library that allows users to learn text representations and text classifiers. It works on standard, generic hardware. Models can later be reduced in size to even fit on mobile devices. The model allows to create an unsupervised learning or supervised learning algorithm for obtaining vector representations for words. Facebook makes available pretrained models for 294 languages. FastText uses a neural network for word embedding.

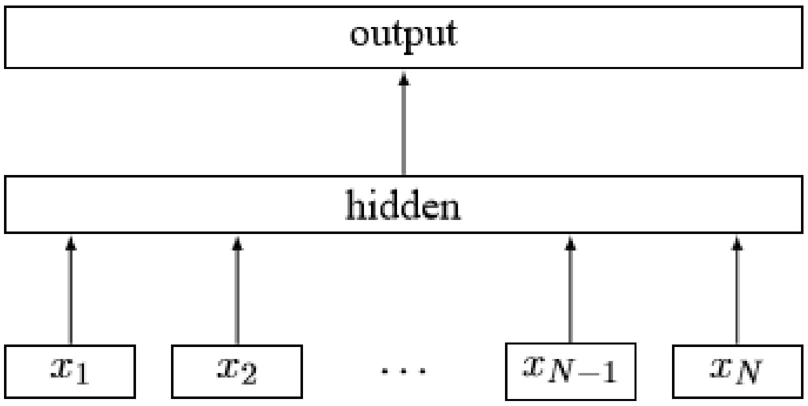


Figure 3.8: FastText Model Architecture [9]

Figure 3.8 shows the Model architecture of fastText for a sentense with N ngram features x_1,..,x_N. These features are embedded and averaged to form the hidden variable.

3.10 Comparing Different Embedding Techniques

Comparison Between Different Embedding Techniques like Word2Vec, GloVe, FastText is illustrated below.

Word2Vec takes texts as training data for a neural network. The resulting embedding captures whether words appear in similar contexts [38].It treats each word in corpus like an atomic entity and generates a vector for each word. In this sense Word2vec is very much like Glove - both treat words as the smallest unit to train on [39].

GloVe focuses on words co-occurrences over the whole corpus. Its embeddings relate to the probabilities that two words appear together [38]. GloVe stresses that the frequency of co-occurrences is vital information and should not be “wasted ”as additional training examples. It builds word embeddings in a way that a combination of word vectors relates directly to the probability of these words’ co-occurrence in the corpus.

FastText(which is essentially an extension of word2vec model) improves on Word2Vec by taking word parts into account, too. This trick enables training of embeddings on smaller datasets and generalization to unknown words [38]. It treats each word as composed of character ngrams. So the vector for a word is made of the sum of this character n grams. For example the word vector "apple" is a sum of the vectors of the n-grams "ap", "app", "appl", "apple" [39]. FastText works well with rare words. Even if a word wasn't seen during training, it can be broken down into n-grams to get its embeddings.

3.11 Summary

In this chapter we described the different topics we learned and used during our thesis on user based depression. Description about different topics of traditional machine learning and deep learning are given in this chapter.

Chapter 4

Proposed Methodology

4.1 Overview

We have tried to experiment with user based depression from reddit data. Below we are presenting our experimental methodology as flow chart diagrams. We have shown 2(two) flow chart diagrams, one for traditional machine learning approach and other one is for deep learning approach. We have described each process in details.

4.2 Proposed Model

As we have used both traditional machine learning and deep learning process in our experiments, we have two models. They are shown separately using flow charts. The details steps of the flow charts are also described in this chapter.

4.2.1 Traditional Machine Learning Methodology

Our Tradition Machine Learning Model contains both depression lexicon based approach and machine learning approach to classify depressed text in reddit. We have used a depression lexicon that contains common depression symptoms from the established clinical assessment questionnaire PHQ-9 [40]. We used it to extract depression related features from reddit posts. In machine learning approach we used we used the entire feature set. We made use of single words, group of words, and triplet of words as features.

Then we have used the features with Term Frequency–Inverse Document Frequency (TF-IDF) to give the depression features a better score after that we have used some different supervised algorithm for emotion classification. We have used Naïve Bayesian, Decision Tree and Support Vector Machine for depression classification, all of these are supervised machine learning algorithm. We have tried to experiment ourselves with detecting the emotion from a text document. In figure "Machine Learning Methodology Diagram" (fig. 4.1) we are presenting our experimental methodology as a flow chart diagram. We will later describe each process in details.

4.2.1.1 Dataset

A number of social media websites were considered as potential data sources for this task. Twitter was discarded because it provided little to no context about the user, is highly dynamic and did not allow them to collect more than 3200 tweets per user, which, in 140-character microblogs, represents only a small amount of text. MTV's A Thin Red Line (ATL), a platform designed to empower distressed teens to respond to issues ranging from sexting to cyberbullying, was also considered, but discarded as there were concerns about redistribution and problems regarding obtaining user history. Eventually, Reddit, a social media and news aggregation website, was selected because of its organization of contents among specific subreddits, and the ease of collecting data using the API provided by Reddit itself. For each user, the organizers collected the maximum number of submissions they could find and were allowed to download through the API (maximum 2000 posts and comments per user). Users with less than 10 submissions were discarded. After the data collection, the users were divided into two cohorts: an experimental depressed group and a control (non-depressed) group. For the depressed group, the organizers searched for phrases associated with self-declaration of depression, such as diagnosed with depression, and then manually examined the posts to filter down to just those redditors who explicitly said they were diagnosed with depression by a physician. These self-declaration posts were omitted from the dataset to avoid making the detection trivial. For the non-depressed group, organizers collected redditors who had participated in depression forums but had no declaration of depression, as well as redditors from other random subreddits. Their final collection contained 531,453 submissions from 892 unique users, of which 486 users were used as training data, and 401 were used as test data. The dataset was published for the pilot task Early Detection of Depression in CLEF eRisk 2017 [13]. For Machine Learning Methodology, from the above dataset we further reduced the dataset to 15000 depressive posts and 15000 non depressive post just for experimental purpose . We picked them randomly. Point to be noted that, our Machine Learning Methodology was just on text classification rather than user classification.

4.2.1.2 Preprocessing

Raw data contains a lot of useless value which can effect result. That is why preprocessing is needed. In our case, Data is in some XML files. First, we had parsed only text of users from XML file. Then the text had passed through some processing step as follows:

- URL Removing
- Punctuation Removing
- Lowecase Conversion
- Tokenizing
- Stopword Removing
- Lemmatizing

After these steps we had got the processed data which is ideal for classification. An example of preprocessing is given in table 4.1 .

4.2.1.3 Feature Selection

In machine learning and statistics, feature selection, also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. After preprocessing we will use the following feature selection.

4.2.1.3.1 Depression Lexicon Features This method is based on the depressive words in the reddit post. We have used a depression lexicon that contains common depression symptoms from the established clinical assessment questionnaire PHQ-9 [40]. We have mapped those words with our reddit post's words and retrieved the depressive words. Table 4.2 illustrates this process.

4.2.1.3.2 Unigram, Bigram, Trigram Features We used the entire feature set. We made use of single words, group of words, and triplet of words.

- Unigram: An n-gram consisting of a single item from a sequence. Here n=1.
- Bigram: Bigrams are a special case of the n-gram where n is 2.
- Trigram: Trigrams are a special case of the n-gram, where n is 3.

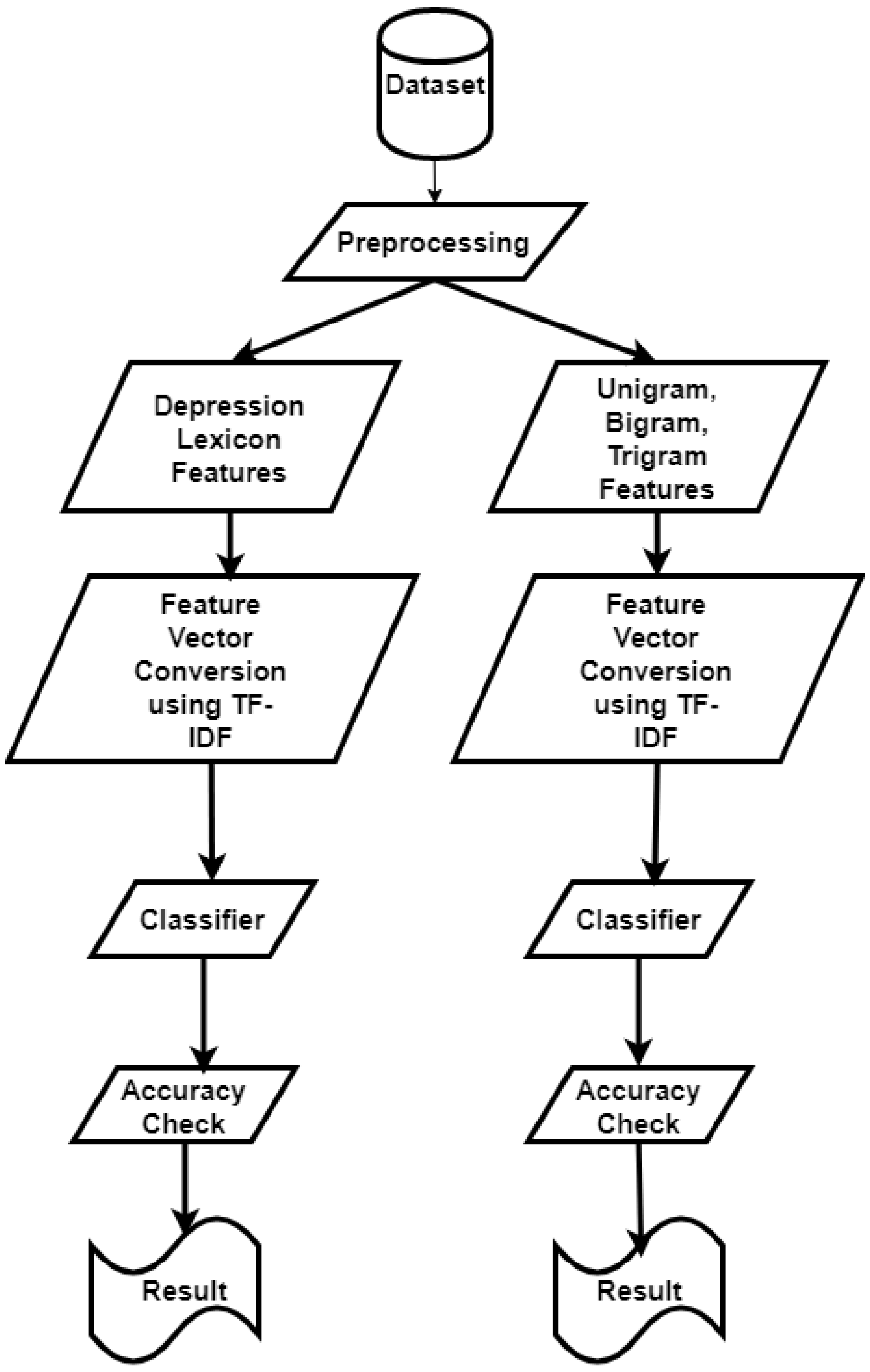

Dataset
Preprocessing
Depression Lexicon Features
Unigram, Bigram, Trigram Features
Feature Vector Conversion using TF-IDF
Feature Vector Conversion using TF-IDF
Classifier
Classifier
Accuracy Check
Accuracy Check
Result
Result

Figure 4.1: Machine Learning Methodology Diagram

Table 4.1: Preprocessing Steps

Actual Text	<DATE>2015-06-30 20:53:00 </DATE> <INFO>reddit post </INFO> <TEXT>Even in my blackest depressions, never regretted having been born. It is true that I had wanted to die, but that is peculiarly different from regretting having been born. https://www.reddit.com/r/help/comments/ 8r0h2l/this_is_not_an_it_help_desk/ </TEXT>
Parsed Text	Even in my blackest depressions, never regretted having been born. It is true that I had wanted to die, but that is peculiarly different from regretting having been born. https://www.reddit.com/r/help/comments/8r0h2l/this_ is_not_an_it_help_desk/
URL Removing	Even in my blackest depressions, never regretted having been born. It is true that I had wanted to die, but that is peculiarly different from regretting having been born.
Punctuation Removing	Even in my blackest depressions never regretted having been born It is true that I had wanted to die but that is peculiarly different from regretting having been born
Lowercase Conversion	even in my blackest depressions never regretted having been born it is true that i had wanted to die but that is peculiarly different from regretting having been born
Tokenizing	’even’, ’in’, ’my’, ’blackest’, ’depressions’, ’never’, ’regretted’, ’having’, ’been’, ’born’, ’it’, ’is’, ’true’, ’that’, ’i’, ’had’, ’wanted’, ’to’, ’die’, ’but’, ’that’, ’is’, ’peculiarly’, ’different’, ’from’, ’regretting’, ’having’, ’been’, ’born’, ’goto’
StopWord Removing	’even’, ’blackest’, ’depressions’, ’never’, ’regretted’, ’born’, ’true’, ’wanted’, ’die’, ’peculiarly’, ’different’, ’regretting’, ’born’, ’goto’
Lemmatizing	’even’, ’blackest’, ’depression’, ’never’, ’regret’, ’born’, ’true’, ’want’, ’die’, ’peculiar’, ’different’, ’regret’, ’born’

Table 4.2: Depressive Lexicon Features

Processed Text	’even’, ’blackest’, ’depression’, ’never’, ’regret’, ’born’, ’true’, ’want’, ’die’, ’peculiar’, ’different’, ’regret’, ’born’
Depressive Words	Depression, die, regret

4.2.1.4 Feature Vector Conversion

For creating the feature vector from the 2 features set we have used two well-known techniques. These are term frequency (TF) and inverse document frequency (TF-IDF).

TF-IDF Features: TF-IDF stands for term frequency-inverse document frequency, and the TF-IDF weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the TF-IDF weighting scheme are often used by search engines as a central tool in scoring and ranking document's relevance given a user query. One of the simplest ranking functions is computed by summing TF-IDF for each query term; many more sophisticated ranking functions are variants of this simple model. TF-IDF can be successfully used for stop-words filtering in various subject fields including text summarization and classification.

4.2.1.5 Machine Learning Classifier

We have used Naïve Bayesian, Decision Tree and Support Vector Machine in our model. Bayesian Classifiers build a probabilistic model based on the word features in different classes. For our depression classification problem we have taken Multinomial Naïve Bayes Classifier. We have trained our model on MultinomialNB provided by the sk-learn package. In Naive Bayes, texts are classified based on posterior probabilities generated based on the presence of different classes of words in texts. This assumptions makes the computations resources needed for a naïve bayes classifier is far more efficient than non naïve bayes approaches which is exponential complexity [41]. Naïve Bayes has been widely used for classifying text because it is simple and fast. SVMs achieve high performance in text categorization since they accept high dimensional feature spaces and sparse feature vectors. Also, text classification using SVMs is very robust to outliers and does not require any parameter tuning. It finds a maximum margin separating hyper plane between two classes of data. For multi-class classification SVM maximize the margin for one vs all classes of data. For our classification problem we have used Linear SVM of sk-learn package. It has been shown that linear kernels based SVM performs a lot better on non-linear SVM in terms of text classification [41]. Decision trees are slow and sometimes suffer from over-fitting. However, its accuracy competes with well-known text classification algorithms such as SVM. For our model we took sk-learn DicisionTreeClassifier. We defined the criterion as entropy as we want the function to measure the quality of a split in text for the information gain. Information gain measures how much organized the input features became after we divide them up using a given feature. Also we have given a static random state value for our classifier. As for text classification a decision tree takes the features as input values. The decision nodes checks the feature values and leaf nodes which assign one of the classes from our multi class emotion. To choose a class for our input text the model start with the initial node as root node which contains a condition on the input features, it than selects a branch based on that feature which leads to a new condition and it makes a new decision based on it. The flow continues until it arrives at a leaf node which provides an class for the input value [42].

4.2.1.6 Result Analysis

We have evaluate the model on 30000 samples. Upon preprocessing some data samples are removed from the orginal dataset. We take accuracy, precision, recall and f1-score for each class. The experimental results are discussed in chapter 5.

4.2.2 Deep Learning Methodology

Different deep learning methodologies have been used by different researchers. We proposed a model that is different and unique in structure. We propose a model that contains two different features concatenated. We used Word2Vec [15] embedding concatenated with Meta Data Features(described in 4.2.2.2) and also FastText [16] [43] embedding concatenated with Meta Data Features with BiLSTM. These procedures are described accordingly below. We also experimented with Meta Data as the only feature with Dense Neural Network. We also experimented with trainable embedding layer and Long Short Term Memory (LSTM), Word2Vec Embedding and Long Short Term Memory (LSTM) with non trainable embedding layer and Glove embedding and Long Short Term Memory (LSTM) with non trainable embedding layer. Our proposed model consists of embedding and Meta Data Features with Bidirectional Long Short Term Memory (LSTM), and the other experimented models are described in the next sections. We took different approaches, experimented on them to check the effect on the results. We have briefly described the different parts of our proposed model in the next section.

4.2.2.1 Embedding

A word embedding is a class of approaches for representing words and documents using a dense vector representation. It is an improvement over more the traditional bag-of-word model encoding schemes where large sparse vectors were used to represent each word or to score each word within a vector to represent an entire vocabulary. These representations were sparse because the vocabularies were vast and a given word or document would be represented by a large vector comprised mostly of zero values. Instead, in an embedding, words are represented by dense vectors where a vector represents the projection of the word into a continuous vector space. The position of a word within the vector space is learned from the text and is based on the words that surround the word when it is used. The position of a word in the learned vector space is referred to as its embedding. Three popular examples of methods of learning word embeddings from the text include Word2Vec [15], GloVe [14] and FastText [16] [43]. In addition to these carefully designed methods, a word embedding can be learned as part of a deep learning model. This can be a slower approach but tailors the model to a specific training dataset. In our experiment we have all the available embedding in different models.

We trained Word2Vec, Glove and FastText with our own data. In the case of concatenation with MetaData Features we used Word2Vec and FastText as it is trained with a lot of data from different social media platform. The result are later compared in the section below.

4.2.2.2 Meta Data Features

Meta Data is "data that provides information about other data". In other words, it is "data about data". For example we can say antidepressant drug is a Meta Data. It illustrates the count of mention of antidepressant drugs in user's post. Many distinct types of metadata exist, including descriptive metadata, structural metadata, administrative metadata, reference metadata and numerical metadata [17]. We used numerical metadata. Moreover depressed users tend to talk about themselves a lot so, they use phrases like 'I' a lot, talk about antidepressants, diagnosis. They use a lot of personal and possessive pronouns. This was found out statistically by authors of [44] by finding out the most common words from depressive posts. Theses are called hand crafted meta features. We used 10 Meta Data features in our different model. 'I' in a sentence, Antidepressant Drugs, Diagnosis, Therapist, Depressive Words, Verbs in a sentence, Personal Pronoun, Possessive Pronoun, Length of Posts and for non depressive posts mention of happiness. For W2V with metadata model, for Meta Data features such as depression, therapist, diagnosis and happy we found all the similar words using W2V(Word2Vec) model and used them as Meta Data features. We counted the number of occurrences of each Meta Data in a depressive post. Function 1 demonstrates how we can get count based feature.Then we normalized them using min max scalar [45] to bring them into a range of 0 to 1 and then used them as features.

4.2.2.3 Bidirectional LSTM

Recurrent Neural Networks are used for modeling sequence data. By adding a loop in the traditional feed-forward neural network, it can use previous information by passing prior information forward. Multiple copies of the same network, each passing information to a

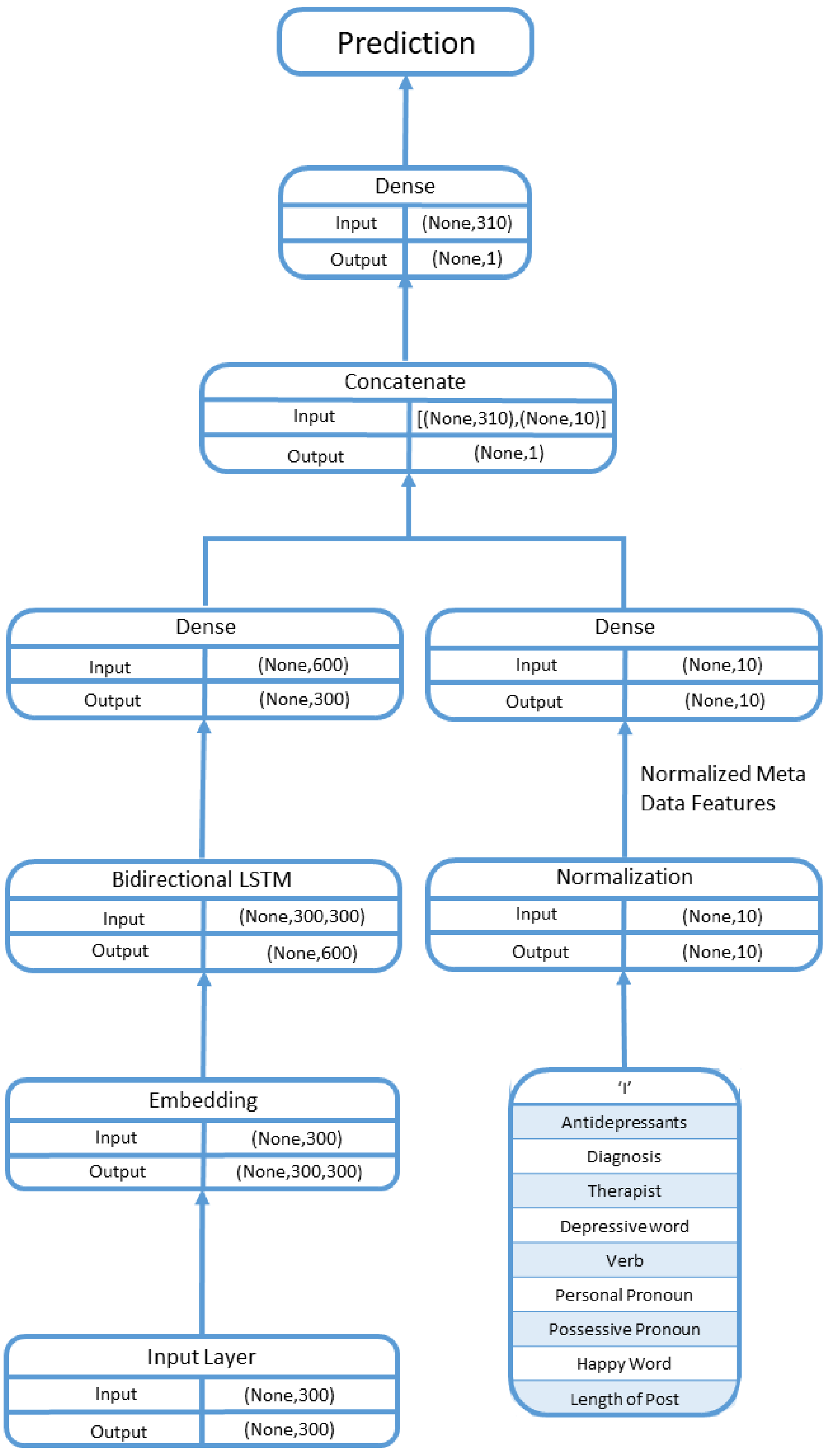


Figure 4.2: Deep Learning Methodology Diagram

Function 1: getCountBasedFeature(post,dictionary)

```
Input: A Post of user, Dictionary
Output: Normalized Feature
1 $tokens \leftarrow tokenize(post)$;
2 $tokenCount \leftarrow 0$;
3 for each token in tokens do
4     if token in dictionary then
5         $tokenCount \leftarrow tokenCount + 1$;
6 return $tokenCount$;
```

successor can be thought of as a recurrent neural network. RNN's suffer from short term memory because of vanishing gradient problem. So RNN's may leave out important information from earlier if a paragraph of text is processed to do predictions. LSTM is a specialized Recurrent Neural Network that is created to mitigate the short-term memory problem of RNN. LSTM's function just like RNN's, but they're capable of learning long-term dependencies using mechanisms called "gates". These gates are different tensor operations that can learn what information to add or remove to the hidden state. In Bidirectional LSTM the concept of LSTM remains the same but we use two LSTM layers where one starts from the first symbol to last and another one does the reverse process. Bidirectional recurrent neural networks add a hidden layer that passes information in a backward direction to more flexible process such information. The figure below illustrates the architecture of a bidirectional recurrent neural network with a single hidden layer.

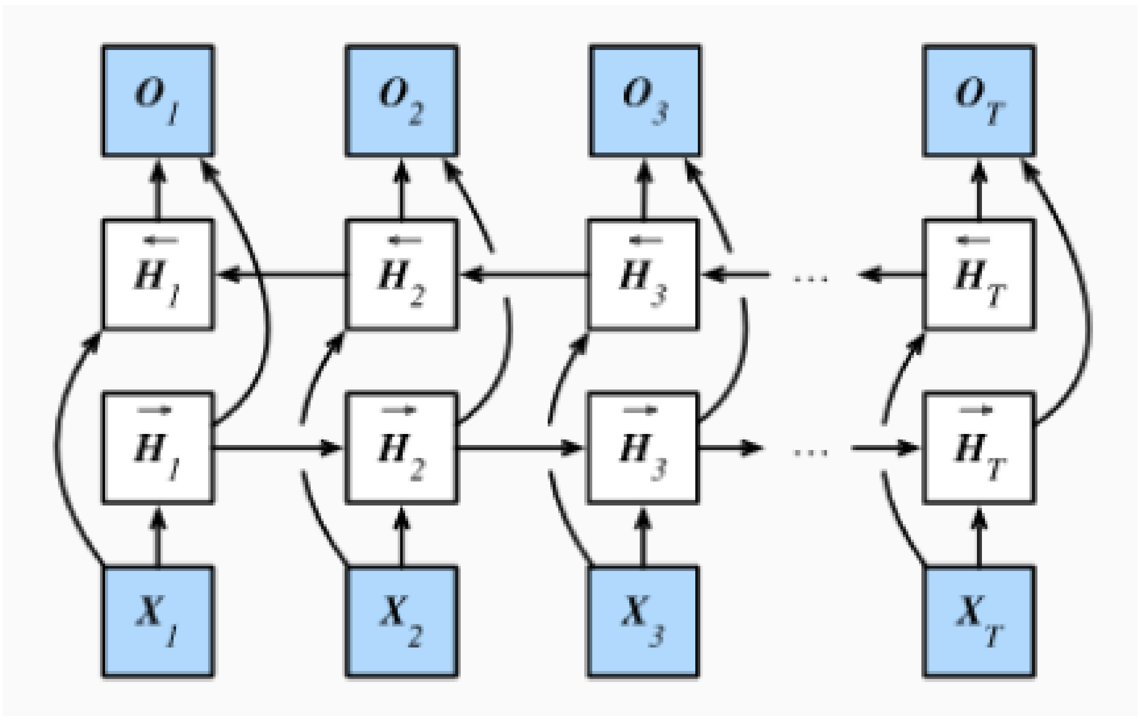


Figure 4.3: Bidirectional RNN [10]

4.2.2.4 Dense Layer

Dense Layer is a fully connected layer. A dense layer is just a regular layer of neurons in a neural network. Each neuron receives input from all the neurons in the previous layer, thus densely connected. We used this layer with normalized Meta Data Feature vector as input to make the network more expressive and accurate. Input and Output dimension of this layer was (None,10). Relu activation function is used here. We also used this layer with Bidirectional LSTM features as input. Input and Output dimension of this layer were (None,600) and (None,300) respectively. Again relu activation function is used here. And Lastly we used this layer with the concatenated features to produce output. Sigmoid activation function is used here. We have experimented with different activation functions and output unit with small portion of data to check which one gives us better result. The optimal one is mentioned here.

4.2.2.5 Concatenation

In this stage, we concatenated both the feature vectors of two Dense Layers, one which came form Meta Data Features and another from LSTM and Embedding layer. For simulation where only one of Meta Data Features or Embedding layer has been used, this concatenation operation was not needed.

4.2.2.6 Prediction using Risk Window

In preliminary analysis of the models of other papers, it was found it was common for a model to make occasional mistakes. But it was recalled that in early depression prediction, the first “+” or “-” prediction is considered final, so occasional mistakes will force a early detection model to abort entirely, even if it has seen only a small number of posts so far. This can have a significant impact on their performance. Thus a technique was introduced, which can apply generally to any model, that trades off between latency and precision [11]. If the model makes a prediction that the user is depressed after post p (flag being raised), it only confirms that prediction if the model continues to make the same (depressed) prediction for the next n posts (we refer to this as the risk window), or, if the user has fewer than n posts remaining, continues to make the same (depressed) prediction for all of the remaining posts. Figure 4.4 demonstrates the process with a risk window of size 10 [11].

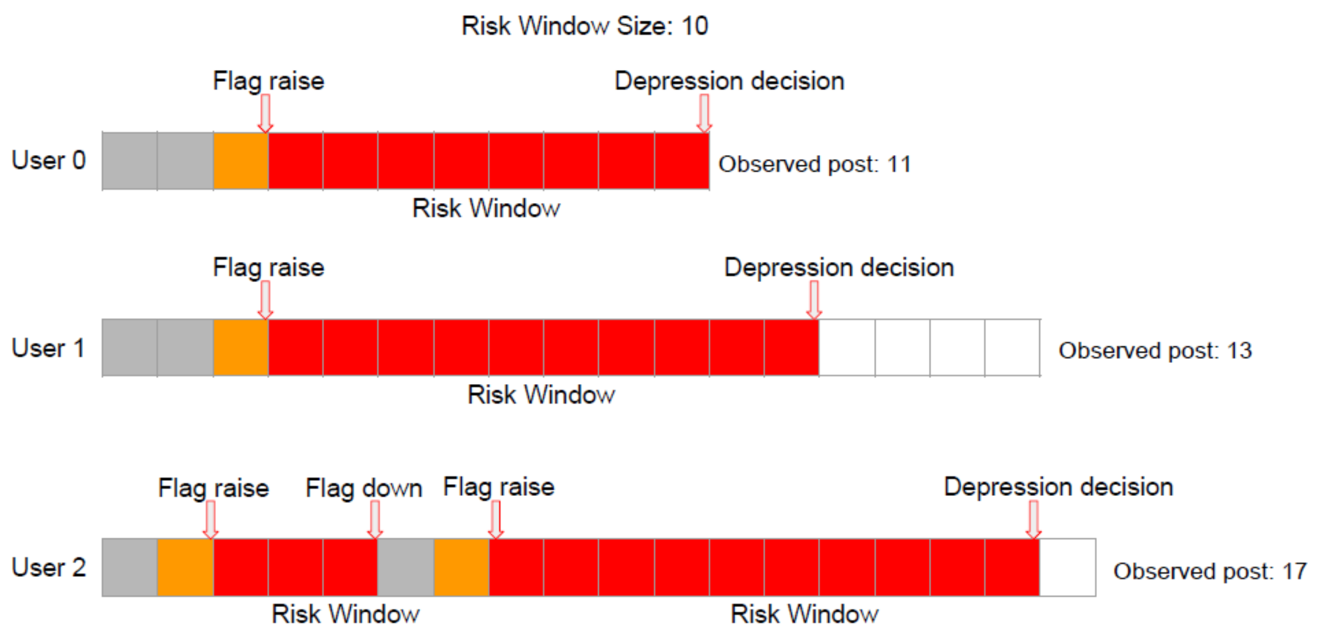


Figure 4.4: Example of post-by-post depression prediction with a risk window of size 10. Each block represents 1 post: gray is observed, orange is where the flag was raised, red is in the risk window, and white is unobserved. User 0 is an example where there are fewer remaining posts than the risk window, and user 2 is an example of restarting after a broken streak. [11]

We evaluated the model on the test reddit dataset. We took Precision, Recall, F1score, $F_{latency}$ for each model. We then compared the result with GRU: Depword and Metamap model [11]. The table 4.3 demonstrates the combination. We have conducted an experiment based on these combinations and evaluated the results in the next chapter. Function 2 illustrates how we had classified user according to risk window.

Table 4.3: Different Combination of our Proposed Model

Model Name	Actual Operation
Model-1	LSTM:Trainable Embedding
Model-2	LSTM:Word2Vec Embedding
Model-3	LSTM:GloVe Embedding
Model-4	DNN:MetaData
Model-5	BiLSTM:Word2Vec Embedding+MetaData
Model-6	BiLSTM:FasText Embedding+MetaData

```
Function 2: ClassifyUser($S_{user}, riskWindowSize, model$)
  Input: A set of user with their posts, size of the risk window and classification
         model
  Output: Predict their state( Having depression or not )
  for each user in $S_{user}$ do
      $riskFlag \leftarrow false$;
      $riskCounter \leftarrow 0$;
      for each post in $P_{user}$ do
          $prediction \leftarrow model.predict(post)$;
          if $prediction = 1$ then
              $riskFlag \leftarrow true$;
          else
              $riskFlag \leftarrow false$;
          if $riskFlag = true$ then
              $riskCounter \leftarrow riskCounter + 1$;
          else
              $riskCounter \leftarrow 0$;
          if $riskCounter = riskWindowSize$ then
              break;
      if $riskCounter = riskWindowSize$ then
          print('This user is depressed');
      else
          print('This user is not depressed');
```

4.3 Summary

In this chapter, we have proposed 2 methodologies the first one is the traditional machine learning approach using depression lexicon. Another one is using deep learning to learn the to evaluate users using Risk Window. The results and simulations along with a baseline model has been compared with our work in chapter 5.

Chapter 5

Experiment Results Evaluation

5.1 Overview

In this chapter, we will show the statistical result of our proposed model. We will represent our result with different visualization tools (like table, graph) to make it easier to understand.

5.2 Basic Terminology

In this section we are going to see some basic terminology that are used to represent result of the model.

- **Confusion Matrix:** A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. Table 5.1 demonstrates the confusion matrix.

Table 5.1: Confusion Matrix

Actual class\Predicted class	C1	C1
C1	True Positives (TP)	False Negatives (FN)
C1	False Positives (FP)	True Negatives (TN)

- **True Positives (TP)**: These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if actual class value indicates that this passenger survived and predicted class tells you the same thing.

- **True Negatives (TN)**: These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing.

- **False Positives (FP)**: When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

- **False Negatives (FN)**: When actual class is yes but predicted class in no. E.g. if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

- **Accuracy**: Percentage of test set tuples that are correctly classified.

$$Accuracy = \frac{(TP + TN)}{All} \tag{5.1}$$

- **Precision**: What % of tuples that the classifier labeled as positive are actually positive.

$$Precision = \frac{TP}{(TP + FP)} \tag{5.2}$$

- **Recall**: What % of positive tuples did the classifier label as positive.

$$Recall = \frac{TP}{(TP + FN)} \tag{5.3}$$

- **F measure (F1 or F-score)**: Harmonic mean of precision and recall.

$$F1\ Score = \frac{2 * (Recall * Precision)}{(Recall + Precision)} \tag{5.4}$$

- **Weighted Average Precision, Recall and F-measure**: To get the weighted average of Precision, Recall and F-measure, first we need to calculate Precision, Recall and F-measure of each class, then weight by the number of instances of each class. That will give us the weighted Precision, Recall and F-measure. For example:

$$Weighted_{Precision} = \frac{(Precision_{class1} * Sample_{Class1}) + (Precision_{class2} * Sample_{Class2})}{(Sample_{Class1} + Sample_{Class2})} \tag{5.5}$$

  Same applied for $Weighted_{Recall}$ and $Weighted_{F1}$.

- **Early Risk Detection Error (ERDE)**: Standard classification measures such as F1-measure(F1), Precision (π) and Recall (ρ) does not take time into account. For that reason, in the pilot task, the measure proposed in paper [46] was also used, called

Early Risk Detection Error (ERDE) measure, which is defined by:

$$ERDE_0(d,k) = \begin{cases} c_{fp} & \text{if d = p AND truth = n} \\ c_{fn} & \text{if d = n AND truth = p} \\ lc_0(k) * c_{tp} & \text{if d = p AND truth = p} \\ 0 & \text{if d = n AND truth = n} \end{cases} \tag{5.6}$$

Where the sigmoid latency cost function, lco(k) is defined by:

$$lc_o(k) = 1 - \frac{1}{1 + e^{k-o}} \tag{5.7}$$

The delay is measured by counting the number (k) of distinct textual items seen before making the binary decision (d) which could be positive (p) or negative (n). The o parameter serves as the "deadline" for decision making, i.e. if a correct positive decision is made in time k > o, it will be taken by ERDEo as if it were incorrect (false positive). Additionally, in the pilot task, it was also set $c_{fn} = c_{tp} = 1$ and $c_{fp} = 52/401 = 0.129$. Note that c_{fp} was calculated by the number of depressed subjects divided by the total subjects in the test set. One problem with this measure is that it does not tell how many post should one expect system to take to predict depression.

- **Latency and Latency-weighted F1**: As an alternative to ERDE, a simple, interpretable way is also available of measuring how long it takes a system to predict a depressed user. The latency of a system is defined to be the median number of posts that the system observes before making a prediction on a depressed user [11]. Formally:

$$latency(U, sys) = \underset{u \in U \wedge ref(u)=+}{\text{median}} time(sys, u) \tag{5.8}$$

where, as above, U is the set of users, ref (u) is the reference label ('+' or '-') assigned to the user, and time(sys,u) is the time in number of posts observed for the system's earliest non-'?' prediction. Latency directly answers our earlier question: how many posts should one expect system to take to predict depression? Latency, a measure of speed, is coupled with measures of accuracy, like precision and recall, to give a complete picture of a system's performance. To produce a single overall measure that combines latency and accuracy, another metric was introduced called latency-weighted F1. Latency-weighted F1, or Flatency, is defined as the product of a model's F1-measure (the harmonic mean of precision and recall) and the median of a set of penalties in the range [0, 1), which are determined by the model's time to predict each user. The penalty is 0 if a prediction is made after exactly 1 post is observed, and approaches 1 as the number of observed posts increases. Formally, we define:

$$P_{latency}(u,sys) = -1 + \frac{2}{1 + e^{-p(time(u,sys)-1)}} \tag{5.9}$$

$$F_{latency}(U,sys) = F_1(U,sys)(1 - \underset{u \in U \wedge ref(u)=+}{\text{median}} P_{latency}(u,sys)) \tag{5.10}$$

where F1(U,sys) is the F-measure(F1-Score) of the system. $F_{latency}$ has a parameter p that defines how quickly the penalty should increase. That parameter should be set in such a way that the latency penalty is 0.5 at the median number of posts of a user. For that reason p is set as 0.0078 after doing analysis.

5.3 Result of Traditional Machine Learning Approaches

In this section, we are going to show the result of traditional machine learning approaches. These approaches are not used to classify user rather used to classify posts of the user. In below, we have shown our result of different classifiers with different techniques. We split dataset as 75% for train set and 25% test set. Description of dataset is given in section 4.2.1.1 . In case of accuracy, precision and recall,f1 score we used weighted average values for all the results.

5.3.1 Result by taking matched word with lexicon as feature

Table 5.2 shows the accuracy, precision, recall and F1 Score for taking matched word with lexicon as feature.

Table 5.2: Result obtained from classifier by using depression lexicon feature

Classifier	$\text{Accuracy}_{weighted}$	$\text{Precision}_{weighted}$	$\text{Recall}_{weighted}$	$\text{F1 Score}_{weighted}$
Naive Bayes	0.643083	0.65	0.61	0.63
Decision Tree	0.583346	0.57	0.53	0.59
KNN	0.530095	0.51	0.55	0.49
SVM	0.644139	0.65	0.61	0.67

Here we can see that SVM gives us 64% accuracy, which is best for this technique.

5.3.2 Result by taking Unigram as feature

Table 5.3 shows the accuracy, precision, recall and F1 Score for taking unigram of word.

Here we can also see that SVM returns quite good result. It reaches 69% accuracy.

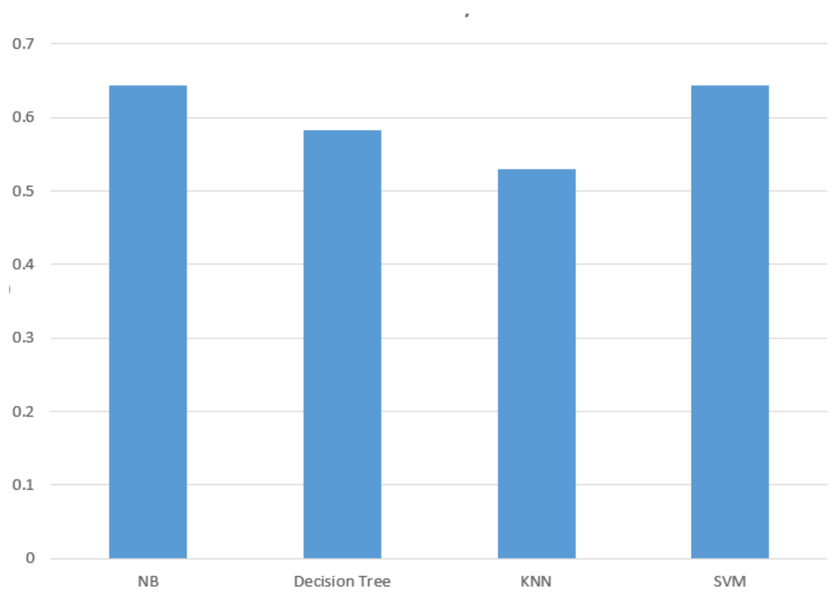


Figure 5.1: Accuracy comparison of classifiers using depression lexicon feature

Table 5.3: Result obtained from classifier by using unigram feature

Classifier	$\text{Accuracy}_{weighted}$	$\text{Precision}_{weighted}$	$\text{Recall}_{weighted}$	$\text{F1 Score}_{weighted}$
Naive Bayes	0.618492	0.66	0.62	0.59
Decision Tree	0.616670	0.64	0.60	0.67
KNN	0.524105	0.56	0.50	0.59
SVM	0.693915	0.70	0.66	0.63

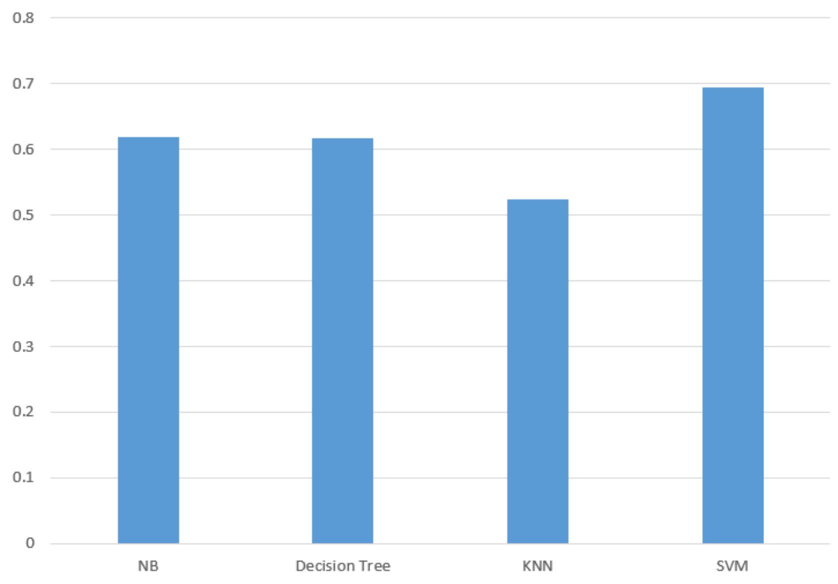


Figure 5.2: Accuracy comparison of classifiers using unigram feature

5.3.3 Result by taking Bigram as feature

Table 5.4 shows the accuracy, precision, recall and F1 Score for taking unigram of word.

Table 5.4: Result obtained from classifier by using bigram feature

Classifier	$\text{Accuracy}_{weighted}$	$\text{Precision}_{weighted}$	$\text{Recall}_{weighted}$	$\text{F1 Score}_{weighted}$
Naive Bayes	0.611421	0.65	0.61	0.59
Decision Tree	0.611742	0.61	0.58	0.63
KNN	0.513070	0.58	0.51	0.54
SVM	0.513070	0.69	0.66	0.68

In this case we can see Decision Tree gives us 61% accuracy which is best for this technique.

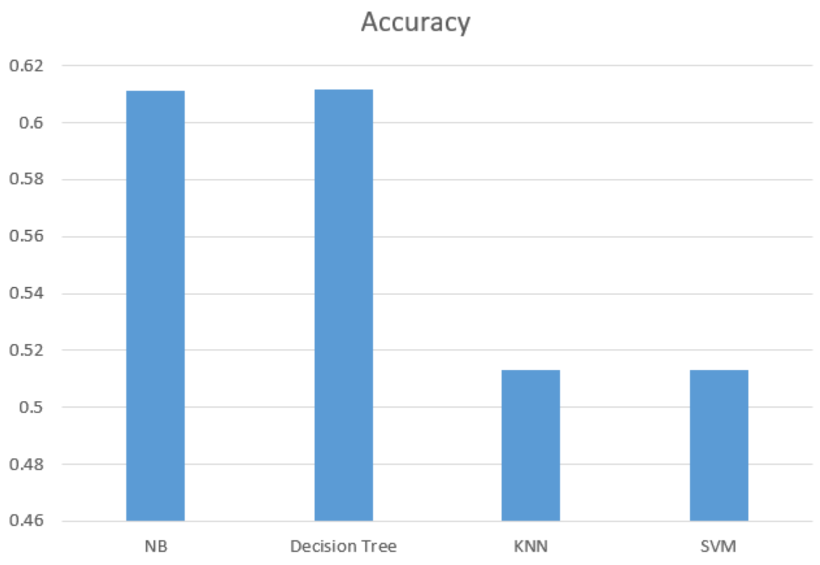


Figure 5.3: Accuracy comparison of classifiers using bigram feature

5.3.4 Result by taking Trigram as feature

Table 5.5 shows the accuracy, precision, recall and F1 Score for taking trigram of word.

Table 5.5: Result obtained from classifier by using trigram feature

Classifier	$\text{Accuracy}_{weighted}$	$\text{Precision}_{weighted}$	$\text{Recall}_{weighted}$	$\text{F1 Score}_{weighted}$
Naive Bayes	0.599421	0.66	0.60	0.56
Decision Tree	0.612171	0.61	0.58	0.64
KNN	0.512801	0.57	0.51	0.53
SVM	0.685987	0.69	0.65	0.67

Here also SVM gives us better result with 68% accuracy.

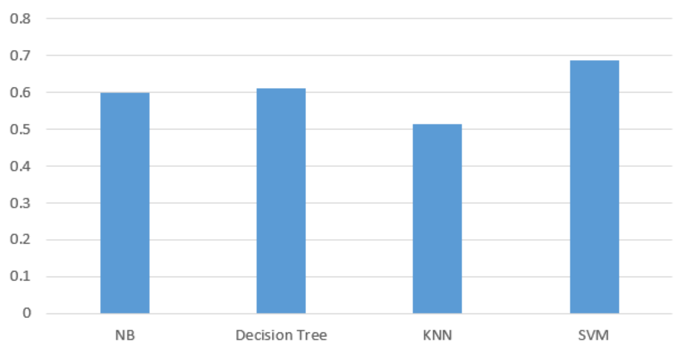


Figure 5.4: Accuracy comparison of classifiers using trigram feature

5.3.5 Result Analysis for Traditional Machine Learning Approaches

From the above result, we found the best result with unigram feature using 75%-25% train test split using SVM classifier. No user based prediction was done here. Only text classification was done. User based classification with risk window should be done with traditional machine learning techniques to get better result.

5.4 Result of Deep Learning Approaches

In this section, we are going to show the result of deep learning approaches. These approaches are used to classify user by predicting the polarity of their post sequentially. Different risk window is used to further find the best result and stop classifier from making any mistake. In case of precision,recall and f1 score we used weighted average values for all the results.

5.4.1 LSTM: Trainable Embedding Layer

The table 5.6 shows the result of LSTM with Trainable Embedding Layer. It's F1 score is comparatively low for all Risk Window so it does not classify users correctly and $F_{latency}$ is low and $ERDE_5$ and $ERDE_{50}$ is comparatively high, which means it's speed of detection is comparatively low.

Table 5.6: Result obtained by using LSTM: Trainable Embedding Layer

Risk Window	**Precision**$_{weighted}$	**Recall**$_{weighted}$	**F1**$_{weighted}$	**F**$_{Latency}$	**ERDE**$_5$	**ERDE**$_{50}$
4	0.81	0.27	0.29	0.14	0.20	0.11
5	0.80	0.34	0.39	0.14	0.20	0.13
6	0.80	0.42	0.41	0.15	0.20	0.14

7	0.80	0.49	0.45	0.16	0.19	0.14
8	0.78	0.57	0.53	0.14	0.18	0.14
9	0.78	0.59	0.58	0.15	0.16	0.14
10	0.78	0.60	0.65	0.17	0.15	0.14
11	0.78	0.62	0.62	0.20	0.14	0.13
12	0.78	0.65	0.60	0.20	0.14	0.13
15	0.79	0.61	0.57	0.21	0.17	0.18
23	0.81	0.49	0.52	0.19	0.16	0.17

5.4.2 LSTM: Word2Vec Embedding

The table 5.7 shows the result of Word2Vec Embedding and LSTM. It's F1 score is comparatively low for all Risk Window so it does not classify users correctly and $F_{latency}$ is low and $ERDE_5$ and $ERDE_{50}$ is comparatively high, which means it's speed of detection is comparatively low but better than LSTM with Trainable Embedding Layer model.

Table 5.7: Result obtained by using LSTM: Word2Vec Embedding

Risk Window	**Precision**$_{weighted}$	**Recall**$_{weighted}$	**F1**$_{weighted}$	**F**$_{Latency}$	**ERDE**$_5$	**ERDE**$_{50}$
4	0.83	0.26	0.23	0.22	0.21	0.11
5	0.81	0.32	0.29	0.19	0.21	0.12
6	0.79	0.36	0.35	0.18	0.21	0.14
7	0.79	0.44	0.39	0.19	0.20	0.14
8	0.79	0.51	0.42	0.20	0.18	0.14
9	0.78	0.58	0.59	0.20	0.17	0.14
10	0.80	0.60	0.62	0.21	0.16	0.13
11	0.79	0.61	0.61	0.19	0.15	0.13
12	0.80	0.63	0.60	0.19	0.14	0.13
15	0.75	0.63	0.53	0.20	0.14	0.12
23	0.71	0.67	0.43	0.20	0.13	0.12

5.4.3 LSTM: GloVe Embedding

The table 5.8 shows the result of Glove Embedding and LSTM. It's F1 score is comparatively low for all Risk Window so it does not classify users correctly and $F_{latency}$ is low and $ERDE_5$ and $ERDE_{50}$ is comparatively high, which means it's speed of detection is comparatively low but better than LSTM with Trainable Embedding Layer model and Word2Vec Embedding with LSTM model.

Table 5.8: Result obtained by using LSTM: GloVe Embedding

Risk Window	**Precision**$_{weighted}$	**Recall**$_{weighted}$	**F1**$_{weighted}$	**F**$_{Latency}$	**ERDE**$_5$	**ERDE**$_{50}$
4	0.86	0.33	0.31	0.11	0.19	0.10
5	0.83	0.40	0.35	0.12	0.20	0.11
6	0.81	0.50	0.45	0.11	0.19	0.13
7	0.80	0.59	0.49	0.14	0.17	0.13
8	0.80	0.60	0.55	0.15	0.16	0.13
9	0.79	0.61	0.63	0.15	0.15	0.13
10	0.79	0.59	0.63	0.15	0.14	0.13
11	0.80	0.63	0.70	0.16	0.14	0.13
12	0.81	0.64	0.73	0.17	0.13	0.13
15	0.76	0.67	0.70	0.18	0.14	0.13
23	0.77	0.68	0.65	0.19	0.13	0.13

5.4.4 DNN: Meta Data

The table 5.5 shows the result of Meta Data and Dense Neural Network. It's F1 score is comparatively low for all Risk Window so it does not classify users correctly and F$_{latency}$ is low and ERDE$_5$ and ERDE$_{50}$ is comparatively high, which means it's speed of detection is comparatively low.

Table 5.9: Result obtained by using DNN: MetaData

Risk Window	**Precision**$_{weighted}$	**Recall**$_{weighted}$	**F1**$_{weighted}$	**F**$_{Latency}$	**ERDE**$_5$	**ERDE**$_{50}$
4	0.84	0.44	0.45	0.14	0.18	0.12
5	0.82	0.59	0.48	0.15	0.17	0.12
6	0.79	0.67	0.55	0.15	0.16	0.13
7	0.80	0.79	0.70	0.15	0.14	0.12
8	0.80	0.82	0.75	0.16	0.14	0.13
9	0.80	0.84	0.78	0.15	0.13	0.13
10	0.83	0.87	0.79	0.16	0.13	0.12
11	0.83	0.87	0.70	0.13	0.13	0.12
12	0.83	0.87	0.62	0.12	0.13	0.13
15	0.76	0.79	0.58	0.11	0.14	0.11
23	0.72	0.65	0.55	0.14	0.15	0.14

5.4.5 BiLSTM: Word2VecEmbedding+MetaData

The table 5.10 shows the result of BiLSTM with Word2VecEmbedding and MetaData. We got the best result with this model with the highest F1 Score, $F_{Latency}$ and the lowest $ERDE_5$ and $ERDE_{50}$. Which means this model is the most consistent at predicting users correctly at a reasonable speed.

Table 5.10: Result obtained by using BiLSTM: Word2VecEmbedding+MetaData

Risk Window	**Precision**$_{weighted}$	**Recall**$_{weighted}$	**F1**$_{weighted}$	**F**$_{Latency}$	**ERDE**$_5$	**ERDE**$_{50}$
4	0.86	0.21	0.18	0.17	0.18	0.10
5	0.87	0.25	0.24	0.22	0.20	0.09
6	0.87	0.28	0.29	0.26	0.21	0.09
7	0.88	0.31	0.34	0.30	0.21	0.09
8	0.86	0.35	0.40	0.34	0.21	0.09
9	0.85	0.40	0.46	0.38	0.20	0.10
10	0.84	0.43	0.50	0.40	0.20	0.10
11	0.83	0.45	0.52	0.39	0.20	0.12
12	0.83	0.50	0.57	0.39	0.19	0.12
15	0.82	0.65	0.70	0.35	0.17	0.13
23	0.79	0.80	0.79	0.21	0.14	0.13

5.4.6 BiLSTM: FastText Embedding+Metamap

The table 5.11 shows the result of BiLSTM with FastTextEmbedding and MetaData. We got the second best result with this model with the second highest F1 Score, $F_{Latency}$ and the lowest $ERDE_5$ and $ERDE_{50}$. Which means this model is the also consistent at predicting users correctly at a reasonable speed.

Table 5.11: Result obtained by using BiLSTM: FastText Embedding+Metamap

Risk Window	**Precision**$_{weighted}$	**Recall**$_{weighted}$	**F1**$_{weighted}$	**F**$_{Latency}$	**ERDE**$_5$	**ERDE**$_{50}$
4	0.89	0.26	0.25	0.23	0.19	0.09
5	0.84	0.30	0.33	0.29	0.20	0.10
6	0.84	0.38	0.44	0.36	0.20	0.10
7	0.85	0.45	0.52	0.37	0.19	0.11
8	0.84	0.52	0.59	0.36	0.19	0.11
9	0.84	0.57	0.64	0.36	0.18	0.12
10	0.83	0.63	0.69	0.34	0.17	0.13

11	0.82	0.67	0.72	0.25	0.16	0.13
12	0.81	0.72	0.76	0.23	0.15	0.13
15	0.80	0.57	0.64	0.33	0.18	0.13
23	0.79	0.76	0.77	0.26	0.15	0.13

5.4.7 Comparison with baseline work

As the baseline work (GRU:DepWords+MetaMap) [11] had the best result for risk window 23, we are comparing the result for risk window 23. Our Model 5 got the best result as it has a better performance in correctly classifying users although at a slower rate. Table 5.12 demonstrates the result comparison between different our approaches and baseline work.

Table 5.12: Result Comparison

Model	Method	Risk Window Size	F1 Score	$F_{Latency}$	$ERDE_5$	$ERDE_{50}$
Baseline Model [11]	**GRU: DepWords+ MetaMap**	**23**	**0.65**	**0.52**	**0.15**	**0.11**
Model-1	LSTM: Trainable Embedding Layer		0.52	0.19	0.16	0.17
Model-2	LSTM: Word2Vec		0.43	0.20	0.13	0.12
Model-3	LSTM: Glove Embedding		0.65	0.19	0.13	0.13
Model-4	DNN:MetaData		0.55	0.14	0.15	0.14
Model-5	**Bi-LSTM: Word2Vec+ MetaData**		**0.79**	**0.21**	**0.14**	**0.13**
Model-6	Bi-LSTM: FastText+ MetaData		0.77	0.26	0.15	0.13

In figure 5.5 we have shown an overall comparison of the F1 Score, $F_{latency}$, $ERDE_5$ and $ERDE_{50}$ of our deep learning approaches. The details of the notations are given in table 4.3. We can conclude that Model-5 has the best performance.

5.5 Summary

From the above experiments, we can conclude that our proposed model has slightly performed better in terms of classifying the user correctly than the recent work that has been done on user based depression. We have tried to solve the limitation of F1 score in our deep learning model. We have tried to solve our limitations of traditional machine learning of only classifying text, in our deep learning methodology, our model learns from the sequence

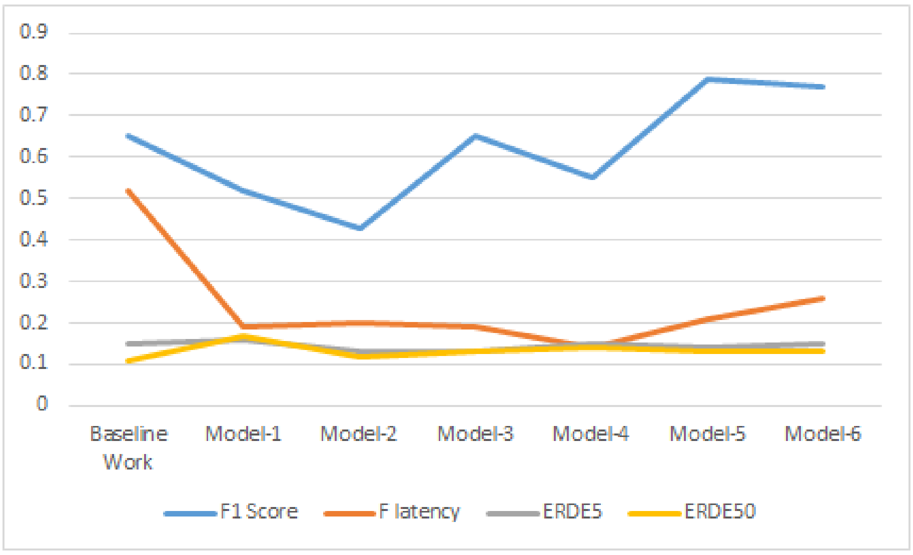


Figure 5.5: Result Comparison of Different models of Deep Learning for Risk Window 23

of text when in machine learning we have only worked on text classification. We can also conclude adding more features in our model can solve the problem of classifying users more quickly.

Chapter 6

Conclusion and Future Works

6.1 Conclusion

Depression detection is one of the toughest problems to solve. Yet we have tried several approaches to detect depression from text. Our main focus was to reduce the amount of time to predict the state of the user, not only to correctly classify them. In our proposed deep learning model it was compared with a baseline work which was conducted using our prepared dataset. We can say upon the observation that our model performs better than the baseline work. We introduced a Hybrid Model with W2Vec Embedding and MetaData features with BiLSTM that performs better than the baseline work.

6.2 Future Works

We are currently working on attention based model to improve performance. Moreover our machine learning approaches are able to classify text rather than user. We are also working on this. We used 10 Metadata feature which can be increased by taking more features like LIWC, MetaMap, Valid Depression Lexicon etc. We are planning to run our model on other dataset as well.

References

[1] "Suicidile statistics." https://en.wikipedia.org/wiki/Suicide_in_the_United_States. Accessed: 2019-09-28.

[2] "Towards Data Science." https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. Accessed: 2019-6-17.

[3] "Applied deep learning." https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6. Accessed: 2019-12-28.

[4] Y. B. I. Goodfellow and A. Courville, *Deep Learning*. The MIT Press, 2016.

[5] "Deep stacked bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction." https://www.semanticscholar.org/paper/Deep-Stacked-Bidirectional. Accessed: 2019-12-30.

[6] "Dense neural network." https://heartbeat.fritz.ai/classification-with-tensorflow-and-dense-neural-networks-8299327a818a. Accessed: 2019-09-28.

[7] "Approximating the softmax." http://ruder.io/word-embeddings-softmax/. Accessed: 2019-12-30.

[8] "Neural word embeddings." https://pathmind.com/wiki/word2vec. Accessed: 2019-01-10.

[9] "Fasttext architecture." http://llcao.net/cu-deeplearning17/pp/class7_FastText.pdf. Accessed: 2019-01-10.

[10] "Bidirectional recurrent neural networks." https://www.d2l.ai/chapter_recurrent-neural-networks/bi-rnn.html. Accessed: 2019-05-30.

[11] F. Sadeque, D. Xu, and S. Bethard, *Measuring Latency of Depression Detection in Social Media*. WSDM'18, February 5-9, 2018, Marine Del Rey, CA, USA, 2018.

[12] "Businessman accused of rapekills." https://punchng.com/businessman-accused-of-rape-kills-self-after-posting-suicide-note-on-tweet. Accessed: 2019-10-30.

[13] D. Losada, F. Crestani, and J. Parapar, *CLEF 2017 eRisk Overview: Early Risk Prediction on the Internet: Experimental Foundations.In Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum*. 2017.

[14] "Glove: Global vectors for word representation." https://nlp.stanford.edu/projects/glove/. Accessed: 2019-12-30.

[15] "Introduction to word embedding and word2vec." https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa. Accessed: 2019-12-30.

[16] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, *Enriching Word Vectors with Subword Information*. 2016.

[17] Directorate and O. Statistics., *OECD Glossary of Statistical Terms - Reference metadata Definition*. 24 May 2018.

[18] G. Dulac-Arnold, L. Denoyer, and P. Gallinari, *Text Classification : A Sequential Reading Approach*. European Conference on Information Retrieval, 2011.

[19] X. Wang, C. Zhang, Y. Ji, L. Sun, L. Wu, and Z. Bao, *A depression detection model based on sentiment analysis in micro-blog social network*. Trends and applications in knowledge discovery and data mining (PAKDD), 2013.

[20] M. D. Choudhury, S. Counts, and E. Horvitz, *Social Media as a Measurement Tool of Depression in Populations*. WebSci, 2013.

[21] T. Nguyen, D. Phung, S. V. B. Dao, and M. Berk, *Affective and content analysis of online depression communities*. IEEE Trans. Affect. Comput., vol. 5, no. 3, pp. 217-226, 2014.

[22] S. Tsugawa, Y. Kikuchi, F. Kishino, K. Nakajima, Y. Itoh, and H. Ohsaki, *Recognizing Depression From Twitter Activity*. Understanding Health through Online Behavior, 2015.

[23] H. Almeida, A. Briand, and M.-J. Meurs, *Detecting Early Risk of Depression from Social Media User-generated Content*. In Proceedings Conference and Labs of the Evaluation Forum CLEF. American Psychiatric Association (2013). Diagnostic and statistical manual of mental disorders (DSM-5 R). American Psychiatric Pub., 2017.

[24] A. MM and A. HF, *Predicting depression levels using social media posts*. 2017 IEEE 13th international Symposium on Autonomous decentralized system (ISADS), 2017.

[25] A. Benton, M. Mitchell, and D. Hovy, *Multi-Task Learning for Mental Health using Social Media Text*. [Online]. Available: https://arxiv.org/abs/1712.03538, 2017.

[26] Hassan, A. U., Hussain, J., Hussain, M., Sadiq, M., Lee, and S., *Sentiment analysis of social networking sites (SNS) data using machine learning approach for the measurement of depression*. 2017 International Conference on Information and Communication Technology Convergence (ICTC), 2017.

[27] F. Sadeque, D. Xu, and S. Bethard, *UArizona at the CLEF eRisk 2017 Pilot Task: Linear and Recurrent Models for Early Depression Detection*. CEUR Workshop Proc. Author manuscript; PMC 2017, 2017.

[28] J. M. Loyola, M. L. Errecalde, H. J. Escalante, and M. M. y Gomez, *Learning When to Classify for Early Text Classification*. Revised Selected Papers. Communications in Computer and Information Science (CCIS), Springer, 790, 24-34., 2018.

[29] S. Paul, J. S. Kalyani, and T. Basu, *Early Detection of Signs of Anorexia and Depression Over Social Media using Effective Machine Learning Frameworks*. CLEF 2018, 2018.

[30] Y. Tyshchenko, *Depression and anxiety detection from blog posts data*. 2018.

[31] D. Maupomés and M. Meurs, *Using topic extraction on social media content for the early detection of depression*. 2018.

[32] M. M. Tadesse, H. Lin, B. Xu, and L. Yang, *Detection of Depression-Related Posts in Reddit Social Media Forum*. 2019 IEEE Access, 2019.

[33] J. C. W. L. B. J. Zinken, K. Zinken and T. Skinner, *Analysis of syntax and word use to predict successful participation in guided self-help for anxiety and depression*. 2010.

[34] "Anxiety and depression association of america." https://adaa.org/learn-from-us/from-the-experts/blog-posts/consumer/depression-and-daily-life. Accessed: 2019-12-28.

[35] E. K. S. Bird and E. Loper, *Natural Language Processing with Python*. O'Reilly Media, 2009.

[36] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, Inc., first ed., 1997.

[37] "Meta features." https://cran.r-project.org/web/packages/mfe/vignettes/mfe-vignette.html. Accessed: 2019-12-30.

[38] "Comparing different embedding techniques." https://cai.tools.sap/blog/glove-and-fasttext-two-popular-word-vector-models-in-nlp/. Accessed: 2020-01-10.

[39] "Difference between word2vec and fasttext." `https://www.quora.com/What-is-the-main-difference-between-word2vec-and-fastText`. Accessed: 2020-01-10.

[40] K. Kroenke, R. L. Spitzer, and J. B. Williams, *The phq-9*. Journal of general internal medicine, 2016.

[41] Y. Yang and X. Liu, *A re-examination of text categorization methods*. Sigir, 1999.

[42] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O'Reilly Media, Inc., 2009.

[43] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, *Bag of Tricks for Efficient Text Classification*. 2016.

[44] Trotzek, M., Koitka, S., Friedrich, and C.M., *Linguistic Metadata Augmented Classifiers at the CLEF 2017 Task for Early Detection of Depression.* Working Notes Conference and Labs of the Evaluation Forum CLEF 2017, Dublin, Ireland, 2017.

[45] "Sklearn minmax scaler." `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html`. Accessed: 2019-01-10.

[46] D. E. Losada and F. Crestani, *A Test Collection for Research on Depression and Language Use.* International Conference of the Cross-Language Evaluation Forum for European Languages. Springer International Publishing., 2016.