Public Sentiment Analysis Based on Social Media Reactions for Bangla Natural Language

Md. Tazimul Hoque
Student Id: 012161021
Department of Computer Science and Engineering
United International University

A thesis submitted for the degree of
M.Sc. in Computer Science & Engineering
June 2020

Approval Certificate

This thesis titled **"Public sentiment analysis based on social media reactions for Bangla natural language"** submitted by Md. Tazimul Hoque, Student ID: 012161021, has been accepted as Satisfactory in fulfillment of the requirement for the degree of Master of Science in Computer Science and Engineering.

Board of Examiners

1. _____

Dr. Mohammad Nurul Huda **Supervisor**
Professor & Director - MSCSE
Department of Computer Science & Engineering (CSE)
United International University (UIU)
United City, Madani Avenue, Badda, Dhaka 1212, Bangladesh.

2. _____

Dr. Md. Saddam Hossain Mukta **Head Examiner**
Assistant Professor
Department of Computer Science & Engineering (CSE)
United International University (UIU)
United City, Madani Avenue, Badda, Dhaka 1212, Bangladesh.

3. _____

Dr. Swakkhar Shatabda **Examiner-I**
Associate Professor & Undergraduate Program Coordinator
Department of Computer Science & Engineering (CSE)
United International University (UIU)
United City, Madani Avenue, Badda, Dhaka 1212, Bangladesh.

4. _____

Rubaiya Rahtin Khan **Examiner-II**
Assistant Professor
Department of Computer Science & Engineering (CSE)
United International University (UIU)
United City, Madani Avenue, Badda, Dhaka 1212, Bangladesh.

5. _____

Dr. Salekul Islam **Ex-Officio**
Professor & Head of the Dept.
Department of Computer Science & Engineering (CSE)
United International University (UIU)
United City, Madani Avenue, Badda, Dhaka 1212, Bangladesh.

Declaration

I, Md. Tazimul Hoque, declare that this thesis titled, **"Public sentiment analysis based on social media reactions for Bangla natural language"** and the research work presented in it are my own. I confirm that:

- This research work was completed while in candidature for a M.Sc. degree at United International University.
- Where any portion of this research work has been submitted previously for any degree or any other qualification at United International University or any other institution, this has been clearly mentioned.
- Where I have discussed any published work of other researchers, this is properly attributed in my writings.
- Where I have quoted from the work of others, the source is always given as reference. With the exception of such quotations, this research work is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date: 25 June, 2020

Md. Tazimul Hoque
Department of Computer Science and Engineering
Masters of Science in Computer Science & Engineering
Student ID: 012161021
United International University (UIU)
United City, Madani Avenue, Badda, Dhaka 1212, Bangladesh.

Abstract

Representing text documents as vector or in numerical format has been a revolution in natural language processing. It represents similar parts of text in such a way that they are very close to each other, making it very easy to classify or find similarities among them. These vectors also represent the way we use the words or parts of documents as well which helps finding similarity even between pair of words. While *word2vec* is such a technique that represents each word as a vector, *doc2vec* takes it to another level by representing a whole sentence or document as a vector. Being able to represent an entire document as a vector allows comparing a substantial number of words or sentences at a time which can save computational power as well as bandwidth. This relatively newer *doc2vec* technology has not yet been implemented for Bengali sentiment analysis and its feasibility is also unknown. In this study, we have trained *doc2vec* and *word2vec* models using a corpus constructed with 10500 Bengali documents. The corpus consists of three types of data differentiated by their polarity i.e. *positive*, *negative* and *neutral*. Later, we have employed several machine learning algorithms for comparing the accuracy of classification. To evaluate machine learning classifiers performance, we've applied k-fold cross validation technique. In k-fold cross validation we've used document vectors directly obtained from *doc2vec* model, and TF-IDF averaged document vectors gained from *word2vec* model.

Published Papers

Work relating to the research presented in this thesis has been published by the author in the following peer-reviewed conference:

1. Hoque, M. T., Islam, A., Ahmed, E., Mamun, K. A., and Huda, M. N. (2019, February). Analyzing Performance of Different Machine Learning Approaches With Doc2vec for Classifying Sentiment of Bengali Natural Language. In 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE) (pp. 1-5). IEEE.

Acknowledgements

Firstly I'm grateful and expressing my gratitude to Almighty Allah for giving me the strength to complete this research work successfully.

My research work titled **"Public sentiment analysis based on social media reactions for Bangla natural language"** has been completed to fulfill the requirement of MS CSE program. I'm thankful to the people from whom I received guidance, cooperation and suggestions throughout the journey.

I would like to express my gratitude to my thesis supervisor, **Dr. Mohammad Nurul Huda**, Professor & Director - MSCSE, Dept. of CSE, United International University, for his supervision, continuous support, encouragement and giving me the opportunity to do this research work with him.

Finally, I'm very much thankful to my parents for their encouragement, continuous support and endless love.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This chapter represents an overview of introductory aspects of our research work in sentiment analysis. It includes current problem statements, motivation to work in this topic, aim and objectives of our work and about the contributions we made by this research. Organization of the thesis section gives an brief outline for remaining chapters of the book.

1.1 Motivation

In recent years, various social media platforms e.g. Facebook, YouTube, Twitter play a vital role in day to day life due to their ease-of-access, portability, and affordability [6, 7]. According to Statistic, around 2.46 billion people are actively using social media worldwide as of 2017 and it is expected to reach 3.02 billion in 2021 where Facebook has remained the most popular one as of April, 2018 [8]. Another survey conducted in September 2018 by StatCounter says that 89.04% of social media users interact using Facebook in Bangladesh [9]. A very large number of data has been comprised over the Internet as a result of enormous dealing with social media platforms which conveys a significant contribution in Sentiment analysis (SA) [6]. To be specific, analyzing the reactions by users accumulated from social media contents and posts lead to categorize them into several labels i.e. sad, angry, love.

Sentiment analysis is also known as opinion mining, mood or emotion analysis which is a well-known part of Natural Language processing (NLP). The year 2001 or around can be marked as the beginning of the research awareness in the field of SA and opinion mining [10]. Research papers mentioning "sentiment analysis" focus specifically on the application of text classification according to their polarity positive (good), negative (bad) or neutral. But now-a-days SA expresses broadly to mean the computational treatment of public opinion or review in textual format, processing natural language data,

computational linguistics and bio-metrics to systematically extract, identify, quantify and study effective states with subjective information [11]. In addition, recent advents in machine learning research, particularly deep learning based methods e.g. recurrent neural network (RNN), avail the opportunities to infer decisions by training a model in SA. Moreover, the latest key technique titled as "*doc2vec*" developed by Google Inc. [12] in which usually a document is represented by a vector, can be an emerging tactics for classifying emotions or opinions from social media reactions and posts. Although a lot of research work has been conducted in the area of SA and they are mainly based on the social media posts written in English, still these areas are yet to be explored for the social media posts in Bengali language.

1.2 Aim and Objectives

The main goal of this experiment is to create and transform data corpus to suitable document embedding model representing numeric vector, hence analysing different machine learning techniques to evaluate the performance and accuracy of the classifiers in the context of Bengali sentiment analysis. Objective of our study can be pointed as -

- Create a standard Bengali sentiment classification corpus.
- Categorizing documents according to selective human sentiments.
- Construct document embedding model representing numeric vectors to work with machine learning algorithms.
- Analysing performance of deep learning and traditional machine learning approaches for Bengali sentiment analysis.

1.3 Contribution

Our work includes system literature review process and we followed standard steps for searching, screening, raw data-extraction, model generation, experimenting with different ML classifiers and reporting accordingly. At initial step, we searched for relevant papers, research reports, journals, presentations that were broadly concerned with sentiment analysis or opinion mining. Relevant research articles were searched into IEEE Explorer, ACM Portal, Springer Linker, Science Direct and Google Search Engine. Systematic search strategy applied to achieve consistent best search results. Search keywords and phrases were selected according to our desired research interest.

This research aims to analyze public sentiments composed in Bengali on any topic and then categorize them into three particular classes i.e. *positive*, *negative* and *neutral sentiment*. For this we are considering Facebook post reactions – *Love, Wow, Sad, Angry,*

and *Haha* which represent different states of emotion. Here, *Love* and *Wow* reactions are considered as *positive sentiment* whilst *Sad* and *Angry* reactions are considered as *negative sentiment*. Facebook added these new reactions feature allowing users to react along with *Like* in a post. We have employed different machine learning methods i.e. Logistic Regression (LR), Support Vector Machine (SVM), Decision Tree (DT), K-Neighbors, Linear Discriminant Analysis (LDA), Gaussian Naive Bayes (GaussianNB), Sequential Model (SM), Long Short-term Memory (LSTM), and Bidirectional Long Short-term Memory (BLSTM) to build classification models so that they can classify the sentiments from users' reactions in different posts published in Bengali.

Our work can be defined into the following phases:

- Corpus construction phase: This phase includes corpus collection, filtering, labeling raw corpus according to sentiment score.
- Model generation phase: This phase focus on choosing and constructing suitable data model representing numeric vectors to work with ML classifiers.
- Experiment phase: This last phase applies different ML classifiers with selected data models to classify sentiment into classes. We briefly discussed about the performance and resulted accuracy of the employed ML classifiers in this step.

1.4 Organization of the Thesis

This experimental thesis work is distributed into rest of the chapters following the below sequence:

- **Chapter 2** presents literature review and background study of machine learning tools and technology we used for this experiment.
- **Chapter 3** describes about our proposed work flow for this experiment.
- **Chapter 4** represents details about our experimental work & discussion about result and performance of employed document embedding systems & machine learning classifiers.
- **Chapter 5** presents conclusion, limitation and future work of this thesis.

Chapter 2

Background Materials

2.1 Literature Review

Previously accomplished research works related to sentiment analysis or opinion mining are discussed in this section.

2.1.1 Non-Bengali Languages

Many research works are accomplished by measuring the overall polarity of a document or sentence to determine if it is a positive or negative review [13, 14, 15]. Turney et al. used simple unsupervised learning algorithm which finds average semantic orientation of the phrases form the review containing adjectives or adverbs [13]. In system [14] Dave et al. trained a classifier using a self-tagged corpus of reviews form web sites. Pang et al. applied machine learning approach for textual data categorization to identify the subjective portions of any document [15]. Phrase-level sentiment analysis is discussed in [16] which can identify contextual sentiment polarity for a given large subset of sentiment expression. In their work they explained that contextual polarity of a phrase may be different from the polarities of the words appear in that phrase. Some popular approaches of sentiment analysis – subjective lexicon, using N-Gram modeling, machine learning are discussed in [17]. Using deep learning model, Ouyang et al. proposed a framework "*word2vec* + Convolutional Neural Network (CNN)" [18] for classifying sentiment of movie reviews into fives labels: positive, somewhat positive, negative, somewhat negative and neural. They achieved 45.4% accuracy.

2.1.2 Bengali Language

Though a lot of works have been explored considering the research works for Bengali in this ground, very few experiments have been investigated in recent years. Chowdhury

et al. worked on sentiment analysis in Bengali microblog posts using SVM and Maximum Entropy (MaxEnt) classification techniques [19]. They collected 1,300 tweets using Twitter API and split the dataset as 1,000 tweets for training and 300 tweets for testing. They identified the overall polarity of a sentence as either negative or positive. Their achieved accuracy is 93% for SVM using unigrams with emoticons as features. Das et al. developed a phrase level polarity classification system using SVM [20]. They constructed a Bengali News corpus containing 3,435 distinct word-forms. It can categorize opinion phrase as either positive or negative. Their evaluated result have a precision of 70.04% and a recall of 63.02%. Amin et al. used "*word2vec*" model for vector representation of Bengali words [21]. They achieved 75.5% of accuracy using "*word2vec*" word co-occurrence score with the words sentiment polarity score. They collected 16,000 Bengali single line and multiline comments from blog posts and tagged them as positive or negative comment by a survey. Hassan et al. used deep recurrent model Long Short Term Memory (LSTM), with two loss functions – binary cross-entropy and categorical cross-entropy for Bengali sentiment analysis [22]. They used 10,000 Bengali and Romanized Bengali text samples which were divided into three categories - Positive, Negative and Ambiguous. They achieved 70% accuracy with Bengali dataset and using Bengali and Romanized Bengali dataset the accuracy score was 55%.

2.2 Natural Language Processing

Natural Language Processing (NLP), is a branch of artificial intelligence (AI). Using natural language, NLP deals with the interaction between computers and human. The main purpose of NLP is to read, understand and make sense of the human languages to represent it in a valuable manner. It uses machine leaning techniques to derive meaning from human languages. Few useful applications derived from NLP :

- Search engine, spell checker, keyword search, questions answering system
- Speech recognition applications, intelligent personal assistants
- Chat bots for customer support, device controlling, ordering goods.
- Recommendation system based on human behavior
- Human sentiment analysis
- Financial risks or fraud detection
- Market prediction based on information retrieved from websites such as products, price location, dates etc.
- Spam detection and data filtering applications

2.3 Sentiment Analysis

A popular topic of Natural Language Processing (NLP) is Sentiment Analysis (SA) which is also recognized as Opinion Mining. Sentiment Analysis identifies and extracts information like opinion, subjectivity, polarity from textual data. Here polarity can be defined as a measurement unit of sentiment or emotion.

Using sentiment analysis, unstructured data can be extract and transformed into structured information of public opinions about news, products, services, brands, politics or any topic that people can express opinions about. This information can be valuable for commercial applications like -

- Market analysis
- Product reviews and feedback
- Movie review
- Public relations
- Customer service

2.3.1 Different Levels of Sentiment Analysis

Sentiment analysis can be applied in different levels e.g. document, sentence and entity level. Each of this level represents its own characteristics regarding opinion mining.

2.3.1.1 Document level

Document level opinion categorization refers to the overall sentiment classification of the full document. For example, given an elaborate movie review, a sentiment analysis system identifies whether user review expresses an overall positive or negative sentiment about that movie. This level of sentiment analysis considers that each document represents opinion on a single entity. Document level classification doesn't evaluate or compare sentiments in its entity level.

2.3.1.2 Sentence level

The task of sentence level sentiment analysis is to determine whether each sentence expresses a single unit of opinion like - positive, negative or neutral. Here neutral generally means no opinion/sentiment at all. Sentence level classification is very closely related with subjectivity classification, that distinguishes sentences which express factual

information. It is noted that subjectivity is not equivalent to opinion as many objective sentences may express opinion.

2.3.1.3 Entity level

Document and sentence level sentiment analyses can't identify what exactly a person liked and did not like. Entity level sentiment analysis looks directly at option itself instead of looking at language construction like - documents, paragraphs, sentences, clauses or phrases. Here the main idea is an opinion consists of a sentiment (emotion) and a target (of that sentiment). Opinion without any target being identified as of limited use. The importance of opinion targets also helps us to understand sentiment classification problem more better. For an example, "I enjoyed the food, though the restaurant environment was not that good." - clearly this sentence has a positive opinion, but we cannot say that the statement is entirely positive. The sentence tells positive opinion about the food, but expresses negative opinion about the service of that restaurant. So the purpose of entity level sentiment analysis is clearly to discover sentiments on entities.

2.4 Corpus Construction

A well constructed data corpus has great impact on machine learning approaches, resulted better performance with high accuracy. In our experiment we constructed a raw corpus using social media as our primary data source. Steps for a corpus construction are discussed below.

2.4.1 Scripting

Scripting refers to the process by which raw data is collected from different websites. We used *python* language to write script which collected necessary data from online pages periodically.

2.4.2 Prepossessing

Prepossessing is an important part of corpus construction. It includes filtering data to reduce noise from data set. Prepossessing helps to make a solid corpus with relevant data only. The filtering rules are defined as per research requirement. We applied the following rules -

- Removing Hyper links.

- Removing Special characters
- Checking for only Bengali Phonetics
- Checking duplicate data entry

2.4.3 Data Set Labeling

We applied supervised machine learning approaches that required labeled data. Labelling a data set refers to the process which maps a single unit data to some predefined class. This mapping can be one-to-one or one-to-many. One-to-one represents a single unit of data to only one class where one-to-many represents an unit of data can belongs to multiple class.

2.5 Data Model Construction

In natural language processing one of the key ideas is how efficiently texts can be converted into numeric vectors so that it can be fed into different machine leaning techniques to perform training and classification. As our corpus contains raw textual data, we had to prepare a suitable data model to work with machine learning algorithms. We looked for different word embedding systems which can represent textual data to numerical vectors.

2.5.1 Word Embedding Techniques

Word embedding system, representations of words as vectors in a predefined vector space, learned by exploiting large amounts of text. This learning technique for texts can represent words having same meaning, using a similar representation. All the words are mapped into one vector and then the vector values are learned by the system in a way that creates a neural network.

2.5.1.1 Word2Vec

One of the latest key techniques for word embedding is *word2vec* developed by Google [23]. *word2vec* consists of two different methods: Continuous Bag of Words (CBOW) and Skip-gram.

- **Skip-Gram :** Skip-gram predicts a window of words given a single word. Let's consider a sentence "He is a very good boy" and a window size of six. Now if we

consider the word “good” as input then Skip-gram should predict: “he”,“is”, “a”, “very”, “boy”.

- **Continuous Bag of Words (CBOW) :** In converse, CBOW can predict a word given surrounding words. The center word vector is generated by the sum of context words.

For their classification algorithm both methods use artificial neural networks. Initially each word is assigned to a random N- dimensional vector. During the training period, using the CBOW or Skip-gram methods, the algorithm learns the optimal vector for each word.

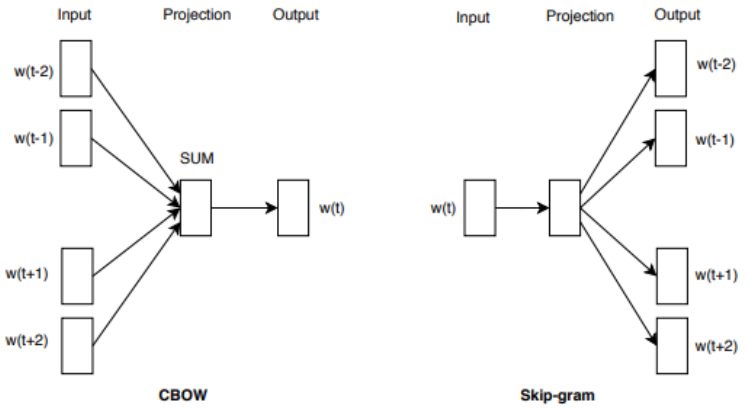

FIGURE 2.1: Architecture of CBOW and Skip-gram [1]

It takes text corpus as input, and provides a set of vectors as output. The output vectors are feature vectors for the words containing in that corpus. *word2vec* itself is not a deep neural network. It converts text into a numerical form which deep neural networks can understand and process further. It can group the vector of similar words together which can detect similarities mathematically. By providing enough data, usage and contexts, *word2vec* can provide highly accurate output for any input words meaning or similarity / association with other words based on past appearance. Thus this prediction can be used to cluster documents and classify them by topic or label. Those clusters can be used for the fundamental basis of sentiment analysis, search engine, document classification and other diverse fields of scientific research.

2.5.1.2 Sentence2Vec

A new model for sentence embedding called *sentence2vec* [24] is introduced which uses unsupervised learning of sentence embedding using compositional n-gram features. To train distributed representations of sentences, *sentence2vec* presents an efficient unsupervised objective. It is an extension of word2vec (CBOW) to sentences. The average of

the source word embedding of its constituent words is defined as sentence embedding. This model is augmented by learning source embedding for not only uni-grams but also n-grams of words present in each sentence, and averaging the n-gram embedding along with the words.

2.5.1.3 Doc2Vec

Goal of *doc2vec* [25] is to make numeric vector representation of documents, regardless of each documents length. This technique is an adaptation of *word2vec*. For *doc2vec* training it requires a set of documents. For each word a word vector W is prepared and for each document a document vector D is prepared. *doc2vec* model by itself is an unsupervised method. *doc2vec* also comes with two different approaches used to built the vector model - Paragraph Vector Distributed Memory (PVDM) and Paragraph Vector Distributed Bag of Words (PV-DBOW).

- **PV-DM :** PV-DM predicts the center word from the set of context words in given document and a document id. PV-DM approach should perform consistently better than the following PV-DBOW approach mentioned in [12].

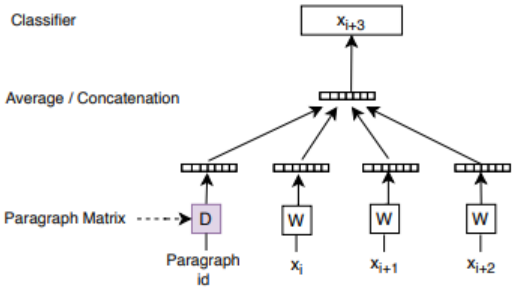

FIGURE 2.2: PV-DM [2]

- **PV-DBOW :** PV-DBOW determines the context probability with given paragraph or document, but ignores context words from input document.

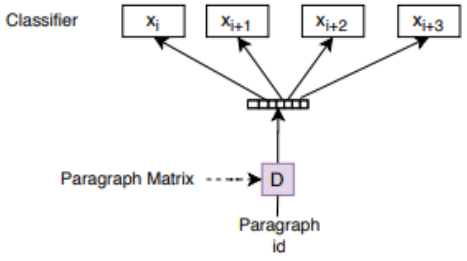

FIGURE 2.3: PV-DBOW [2]

Tags can be assigned for each documents in *doc2vec* and we can easily get their representation as vectors. Figure -2.4 shows *doc2vec* representation with tag vector.

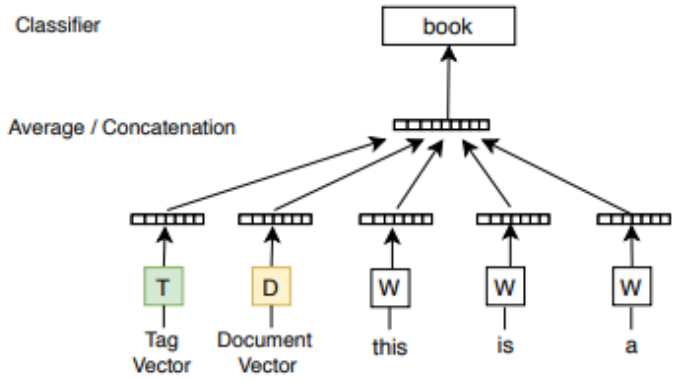

FIGURE 2.4: *doc2vec* model with tag vector [3]

2.6 Types of Machine Learning Algorithms

Machine learning algorithms can gain information from data and improve their outcome from experience without human mediation. Different machine learning techniques are described below.

2.6.1 Supervised Machine Learning

Supervised learning algorithm operates under direct supervision which is - a set of labeled data corpus is fed into the system with some strict rule to operate. After analyzes training data-set, supervised learning algorithm produces an inferred function, which can be used for mapping new inputs.

2.6.2 Unsupervised Machine Learning

In unsupervised machine learning, system is trained with unlabeled data. The system will be able to classify new inputs after it learns patterns from data. It is particularly useful in cases where we don't know what to look for in data-set. Two main methods are employed in unsupervised learning - principal component and cluster analysis.

2.6.3 Semi-supervised Machine Learning

Semi supervised learning technique uses unlabeled data for training, usually mixing a small amount of labeled data with a large set of unlabeled data. This learning approach

falls between supervised learning (trained with labeled data) and unsupervised learning (trained with un-labeled data).

2.6.4 Reinforcement Machine Learning

Reinforcement learning is a type of machine learning, hence it's a branch of artificial intelligence. In an interactive fashion, reinforcement technique continuously learns from the environment. In this way, the system learns from its experiences of the environment until it explores the full range of possible states.

2.7 Machine Learning Tools for Classification

We have used supervised machine learning techniques in this experiment. Our employed machine learning classifiers can be divided as deep learning based ML classifiers and regular ML classifiers.

2.7.1 Regular Machine Learning Classifiers

Regular machine learning techniques use algorithms to process data, learn from it, hence make decisions based on what has been learned. We experimented with regular machine learning classifiers e.g. LR, LDA, SVM, K-Neighbors, DT and GaussianNB.

2.7.1.1 Logistic Regression (LR)

Logistic Regression [26] is known as direct probability model in statistics, developed by statistician D. R. Cox in 1958. It's a predictive analysis like all regression analyses. A binary response can be determined using binary logistic model based on one or more predictor data features. This ability makes LR a probabilistic classification model in the field of machine learning.

In optimization problem, binary-class L2 penalized LR minimizes cost function represented in eq-2.1.

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} log(exp(-y_i(X_i^T w + c) + 1) \tag{2.1}$$

2.7.1.2 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis(LDA) is also known as Normal Discriminant Analysis or Discriminant Function Analysis. For supervised classification problems, LDA is commonly used to reduce dimensionality. It is employed for modeling differences in groups like separating two or more classes. To project the features in higher dimension space into a lower dimension space, LDA is mostly used technique.

Equation for LDA can be derived using simple probabilistic models. Here for each class k, conditional data distribution is $P(X|y = k)$. Using Bayes formula, we can obtain the prediction:

$$P(y = k|X) = \frac{P(X|y = k)P(y = k)}{P(X)} = \frac{P(X|y = k)P(y = k)}{\sum_l P(X|y = l) \cdot P(y = l)} \tag{2.2}$$

and we choose the class k which maximizes conditional probability. $P(X|y)$ is modeled as a multivariate Gaussian distribution for linear and quadratic discriminant analysis with density:

$$P(X|y = k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(X - \mu_k)^t \Sigma_k^{-1}(X - \mu_k)\right) \tag{2.3}$$

where number of features is d.

If we want to use this model as classifier, from the training data we need to estimate - class priors $P(y = k)$, class means μ_k and the co-variance matrices.

For LDA, Gaussians for each class are considered to share same co-variance matrix : $\Sigma_k = \Sigma$, for all k. It leads us to linear decision surfaces, which can be determined by comparing log-probability ratios $\log[P(y = k|X)/P(y = l|X)]$:

$$\begin{aligned} \log\left(\frac{P(y = k|X)}{P(y = l|X)}\right) = \log\left(\frac{P(X|y = k)P(y = k)}{P(X|y = l)P(y = l)}\right) = 0 \Leftrightarrow \\ (\mu_k - \mu_l)^t \Sigma^{-1} X = \frac{1}{2}(\mu_k^t \Sigma^{-1} \mu_k - \mu_l^t \Sigma^{-1} \mu_l) - \log \frac{P(y = k)}{P(y = l)} \end{aligned} \tag{2.4}$$

2.7.1.3 Support Vector Machine (SVM)

Support Vector Machine (SVM) [27] is a supervised learning technique which can be applied to any classification or regression task. SVM is an extension to nonlinear model of the generalized portrait algorithm developed by Vladimir Vapnik. SVM algorithm operates based on the statistical learning approach and the Vapnik Chervonenkis dimension developed by Vladimir Vapnik and Alexey Chervonenkis.

A hyper-plane or set of hyper-planes is constructed by SVM in a higher dimension space, which can be applied in classification problems. Using hyper-plane a good separation can be achieved which has the largest distance to the nearest training data points of any class, since in general the larger the margin the lower the generalization error of the classifier. Let's consider training vectors $x_i \in \mathbb{R}^p, i = 1, ..., n$, in 2 classes and a vector $y \in 1, -1^n$, SVM solves the mathematical eq-2.5

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i \tag{2.5}$$

Here $y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i$ and $\zeta_i \geq 0, i = 1, ..., n$. Its dual equation is represented by eq- 2.6.

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha \tag{2.6}$$

Here $y^T \alpha = 0$ and $0 \leq \alpha_i \leq C, i = 1, ..., n$, where e represents the vector of all ones, $C > 0$ is the upper bound, $\mathbb{Q}$ is n by n positive semi-definite matrix $\mathbb{Q}_{ij} \equiv K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is the kernel. Using function ϕ, training vectors are implicitly mapped into a higher dimensional space. To make decision, eq-2.7 is used.

$$\text{sgn}(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + \rho) \tag{2.7}$$

2.7.1.4 K-Nearest Neighbors

K-Nearest Neighbors(KNN) is one of the most basic yet essential classification algorithm in the field of machine learning. It follows supervised learning technique. It has been applied mostly in data mining fields, pattern recognition, intrusion detection. In real-life scenarios it is widely disposable since it is non-parametric, that means about the distribution of data, it does not make any underlying assumptions.

KNN calculates the distance between data points. For this, simple Euclidean Distance formula can be used:

$$d(p, q) = d(q, p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{2.8}$$

2.7.1.5 Decision Tree (DT)

Decision Tree is known as non-parametric supervised learning technique, mostly used in data classification and regression. Its target is to build a model which can assume a

target variable's value by learning simple decision rules obtained from the data features. The deeper the decision tree goes, more complex decision rules generate and the model becomes more fitter.

Provided training vectors $x_i \in R^n$, i=1,2,....,l and a label vector y (where $y \in R^l$), a decision tree recursively partitions the space in a way that data information with same labels are grouped together.

Let's consider data at node m is represented by Q. For each candidate split $\theta = (j, t_m)$ consisting of a feature j and threshold t_m, partition the data into $Q_{left}(\theta)$ and $Q_{right}(\theta)$ subsets

$$\begin{aligned} Q_{left}(\theta) &= (x, y)|x_j <= t_m \\ Q_{right}(\theta) &= Q \setminus Q_{left}(\theta) \end{aligned} \tag{2.9}$$

Using an impurity function $H()$ we can determine the impurity at m, the selection depends on the task being performed (either classification or regression)

$$G(Q, \theta) = \frac{n_{left}}{N_m} H(Q_{left}(\theta)) + \frac{n_{right}}{N_m} H(Q_{right}(\theta)) \tag{2.10}$$

To minimise impurity

$$\theta^* = \text{argmin}_\theta \, G(Q, \theta) \tag{2.11}$$

Recurse for subsets $Q_{left}(\theta^*)$ and $Q_{right}(\theta^*)$ until the maximum allowable depth is reached, $N_m < \min_{samples}$ or $N_m = 1$.

2.7.1.6 Gaussian Naive Bayes (GaussianNB)

Gaussian Naive Bayes (GaussianNB), is a special branch of Naive Bayes, mostly used for features having continuous value. It's considered that all the features have normal distribution.

The following mathematical relation comes from Bayes' theorem, where class variable y and dependent feature vector x_1 through x_n, :

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)} \tag{2.12}$$

Using the naive conditional independence assumption

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y), \tag{2.13}$$

this relationship is simplified for all i as

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)} \tag{2.14}$$

Since $P(x_1, \ldots, x_n)$ is constant given the input, following classification rule can be used:

$$\begin{aligned} P(y \mid x_1, \ldots, x_n) &\propto P(y)\prod_{i=1}^{n} P(x_i \mid y) \\ &\Downarrow \\ \hat{y} &= \arg\max_y P(y)\prod_{i=1}^{n} P(x_i \mid y), \end{aligned} \tag{2.15}$$

The likelihood of the features in GaussianNB is assumed to be Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \tag{2.16}$$

Maximum likelihood is used to estimate σ_y and μ_y parameters.

2.7.2 Deep Learning Classifiers

Deep learning is a most popular part of machine learning in artificial intelligence based on neural network. It is also known as deep neural network or deep structured learning. We experimented with deep learning based classifier e.g. LSTM, BLSTM & SM.

2.7.2.1 Long Short-term Memory (LSTM)

Long short-term memory networks [28] mostly called "LSTMs", are special kind of recurrent neural network (RNN) architecture. It can remember values over arbitrary intervals. LSTM is designed to avoid the long-term dependency problem of RNN. It addresses vanishing/exploding gradient problem to allow learning of long-term dependencies. Figure -2.5 represents LSTM cell [4] design.

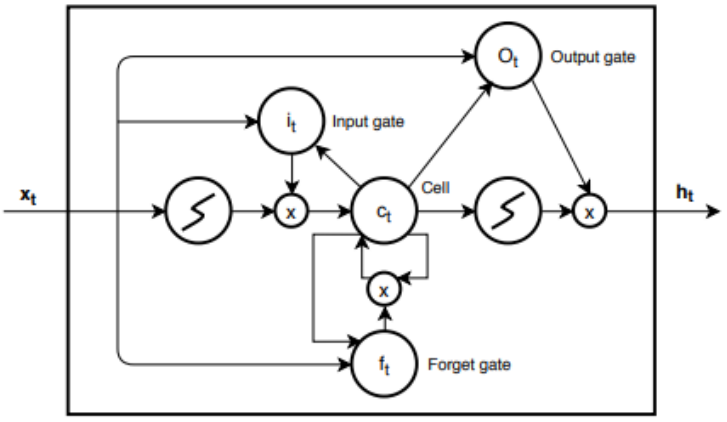

FIGURE 2.5: LSTM block containing input, output and forget gates [4]

2.7.2.2 Bidirectional Long Short-term Memory (BLSTM)

An extension of traditional LSTM is called Bidirectional LSTM which can significantly improve model performance in sequence classification problems. In the input sequence, BLSTM trains two LSTM instead of one. First recurrent layer is duplicated in the network to create two layers side-by-side. After that it provides the input sequence as-is as input to the first layer and a reversed copy of the input sequence is provided to the second layer. This action serves additional context to the network and result outcome in faster and even fuller learning on the problem. Bidirectional networks are faster and more effective in any sequence classification problem than unidirectional ones. Figure -2.6 shows BLSTM classifier design [5].

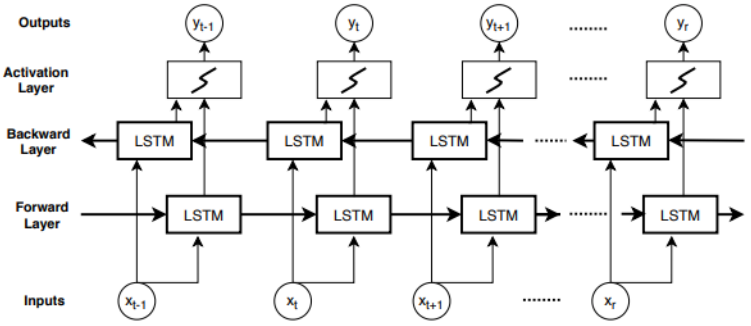

FIGURE 2.6: BLSTM classifier design [5]

2.7.2.3 Sequential Model (SM)

Deep learning based Python library Keras [29] focuses on the creation of models as a sequence of layers. Sequential class contains very simple model which is actually a linear stack of Layers. Using Sequential class constructor we can easily define all of the layers it requires, hence the model is ready to use. Sequential model requires prior knowledge

about its input shape. For this purpose, in the first layer of a Sequential model, its input shape information is served. Using compile method, learning process is configured before training a model. Input data and labels are represented using Numpy arrays to train Keras models. Usually fit function is used to train a model.

2.8 Performance Evaluation

To evaluate overall performance of our employed machine learning classifiers, we've used k-fold cross validation with evaluation scores like accuracy, f1-score, precision and recall. For multi-class classification, we've applied *macro* average method to calculate precision, recall and f1-score. Accuracy is most used when all the classes are equally important. On the other hand, F1-score gives a better measure of the incorrectly classified cases than the accuracy metric. We need the precision and recall to calculate the F1-score. Using confusion matrix, average performance of the model can be determined.

2.8.1 Confusion Matrix

Confusion matrix (CM) [30] represents information of actual and predicted classifications done by a classifier. Generally performance of any ML classifier is measured using the numeric information presented in confusion matrix. Confusion matrix for a two class classifier is represented in Table-2.1.

Table 2.1: Confusion Matrix for Binary Class Classifier

		Predicted	
		Negative	Positive
Actual	Negative	tn	fp
	Positive	fn	tp

The entities of confusion matrix represented in Table-2.1 have the following meaning:

- tn represents accurate predictions for negative entity
- fp represents wrong predictions for positive entity
- fn represents wrong predictions for negative entity
- tp represents accurate predictions for positive entity

2.8.2 Precision

Precision (P) [31] is ratio of correctly identified positive cases. It can be calculated using the eq-2.17.

$$P = \frac{tp}{tp + fp} \tag{2.17}$$

2.8.3 Recall

Recall (R) [31] is the ratio of positive observations that were correctly identified. It can be represented using the eq-2.18

$$R = \frac{tp}{tp + fn} \tag{2.18}$$

2.8.4 F1-Score

F1-Score (F_1) which is also known as balanced F-score or traditional F-measure [31], is harmonic mean of precision and recall. We can calculate it using eq-2.19.

$$F_1 = 2\frac{P * R}{P + R} \tag{2.19}$$

For F1 score :

- best_score = 1
- worst_score = 0

2.8.5 Accuracy

Accuracy (A) is the ratio of correctly identified observation to the total observations hence it's the most intuitive performance measure matrix. Accuracy calculation is shown in eq-2.20

$$A = \frac{tp + tn}{tp + tn + fp + fn} \tag{2.20}$$

2.8.6 Macro Average for Precision, Recall and F1-score

This method provides average values for independently calculated precision and recall for each class. Then f1-score is determined using harmonic mean of macro averaged precision and recall scores. Macro average is suitable when we have balanced data-set in each class.

Let's consider for class A, B and C, we've corresponding precision values P_a , P_b , P_c and recall values R_a, R_b, R_c.

We can calculate macro average precision (P) using eq-2.21.

$$P = \frac{P_a + P_b + P_c}{3} \tag{2.21}$$

eq-2.22 represents calculation for macro average recall (R).

$$R = \frac{R_a + R_b + R_c}{3} \tag{2.22}$$

2.8.7 k-Fold Cross Validation

Cross validation [32] is a well know approach to evaluate performance of a ML classifier model. It is also known as re-sampling procedure for a model containing limited data. In this approach, a portion of data is kept aside which won't be used while training, later that data sample will be used for testing the ML classifier.

k-Fold cross validation procedure has one parameter called k, which represents total number of groups that a given data-set is to be split into. We can use a specific value for the parameter k and then use this number in place of k to refer the cross validation. The working procedure of k-fold cross validation is, it takes a group from k split and hold it as test data-set. Remaining (k-1) groups are used as training data-set. After completing the training step, evaluation score is retrained for the ML classifier using the test data-set. This procedure continues by shifting test data-set group. Finally the performance of the ML classifier is evaluated based on the evaluation scores from each step.

We can observe k-fold procedure with an example. Let's consider a data-set of 6 observations and we'll split it into 3 groups. That means k=3, and we can refer it to 3-fold cross validation.

Data-set = [1, 2, 3, 4, 5, 6]

Splitting this data-set into 3 groups:

Group1 = [1, 3]

Group2 = [4, 5]

Group3 = [6, 2]

Using 3-fold cross validation, we'll have 3 data-sets to train and test any ML classifier.

Model1: Train data-set Group1 + Group2, Test data-set Group3

Model2: Train data-set Group2 + Group3, Test data-set Group1

Model3: Train data-set Group3 + Group1, Test data-set Group2

The evaluation scores (accuracy, f1, precision, recall) for ML classifier can be retained for each model and then we can use those scores to analyze that ML classifier performance on the given data-set.

2.9 Summary

We begin this chapter with a discussion of related works accomplished previously for sentiment analysis. Both Bengali and non-Bengali works were elaborately discussed. Then we explained different level of sentiment analysis, importance of well structured corpus for ML applications, preparing data model to work with ML classifiers. After that we focused on different types of machine learning and our employed machine learning classifiers. Last of all we finished this chapter with explaining cross validation, performance matrix and their importance to evaluate machine learning classifiers.

Chapter 3

Proposed Method

We are going to present our proposed architecture for this research work in this chapter. It covers the data collection planning, data filtering based on our need, data labeling to model generation, training and testing approaches for selected ML classifiers.

3.1 Overview of proposed system

As we aimed to work with sentiment analysis, we had to narrow our research interest in this field to be more specific about what we like to do. We were interested to work for Bengali sentiment analysis. For that purpose initially we looked for related works accomplished in this sector in past few years and then planned for our own work. For sentiment classification, we finalized to use supervised machine learning technique which requires labeled data. Machine learning techniques in classification problem require a good collection of data set. The data set is required to train and test the performance of ML classifiers. So our first challenge was to look for a data source which will provide Bengali textual data. Considering all those needs, Facebook was a good candidate for our primary data source. In the Figure-3.1 we presented an overview diagram of our research work. Our proposed approach can be divided into sub parts - data collection, filtering, labeling, data model generation, train ML classifiers, test and evaluate performance of ML classifiers.

3.2 Corpus Creation

Our corpus creation planning can be divided into three steps - data collection, data filtering and data labeling. These three steps are described below -

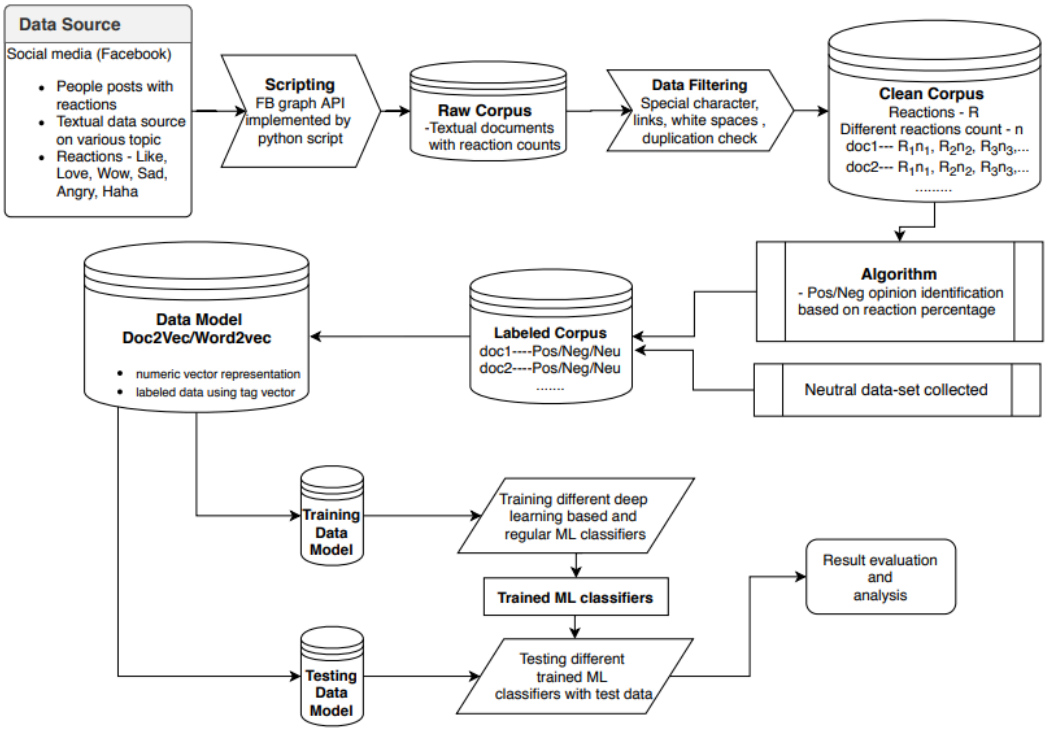

FIGURE 3.1: Proposed architecture of our research work

3.2.1 Data Collection

Our primary data source was Facebook from where we have collected textual data with user reaction counts. It has been done using Facebook graph API implemented by python script. A set of neutral Bengali sentence list is collected manually for further experiment.

3.2.2 Data Filtering

Data filtering process is required to reduce noise from data and also to filter anything on demand. We reduced noise from our collected data set by filtering hyperlinks, special characters, checking duplicate data. As we aimed to work for Bengali sentiment analysis, we filtered any characters except Bengali phonetics.

3.2.3 Data Labeling

For sentiment classification we've considered positive, negative and neutral polarity. Each of the documents in our data set contains user reaction counts. We have developed an algorithm-1 to prepare labeled data (positive and negative) using the reaction counts. We also checked if a document is polarized or not using this algorithm-2. Thus we have constructed our labeled corpus.

3.3 Data Model Selection

To work with ML classifiers, we needed to select a word embedding system which represents textual data as numeric vectors. We explored latest word embedding technologies and found *word2vec*, *sentence2vec* and *doc2vec* quite useful and interesting. We've prepared *doc2vec* and TF-IDF averaged document vector model using *word2vec* from our textual data-set to work with further steps.

3.4 Choosing Machine Learning Classifiers

We have selected Python based deep learning library Keras [29] to implement it's own Sequential Model (SM) API. Then we enhanced our experiment by adding LSTM cell and Bidirectional LSTM layer with SM. BLSTM was chosen to implement for it's well-known performance in sequence classification problem.

Among other traditional ML classifiers, we chose to train and test the performance of LR, LDA, SVM, K-Neighbors, DT and GaussianNB. These traditional classifiers have been implemented using Python based machine learning library scikit-learn [33].

3.5 Result and Performance Evaluation

To evaluate each ML classifiers performance, we chose to use k-fold cross validation with performance evaluation scores - accuracy and F1 score. Using only accuracy matrix wont provide any good result for imbalanced number of data in each classes. But our final data-set contains equal number of documents for positive, negative and neutral sentiments. So using accuracy matrix along with F1 score provided much better insight for result and performance analysis in our experiment.

3.6 Summary

We have discussed about our research planning and presented it step by step. Also a diagram presenting the total work flow is shown in Figure-3.1. This chapter represents different approaches used in each steps of this research work to achieve the best outcome.

Chapter 4

Experimental Analysis

4.1 Experiments

In this chapter, we have discussed about our experimental setup. Section-4.1.1 describes about our corpus collection, filtering process and data-set labeling. Section-4.1.2 represents how we constructed TF-IDF averaged *word2vec* and *doc2vec* models from labeled data-set and finally Section-4.1.3 describes about the parameters we have used to train and test our model for different machine learning algorithms.

4.1.1 Corpus Construction

Aim of this study is to analyze public sentiment on any topic from Bengali text and then categorize it based on sentiment polarity. We have considered positive, negative and neutral sentiment in this work. To construct a corpus for Bengali sentiment analysis, different sources have been considered, among which Facebook post data seems most promising for SA as they represent the most natural form of language. In Facebook posts, people react with different reactions i.e. “Like”, “Love”, “Wow”, “Sad”, “Angry”, and “Haha”, each of which represent different states of emotion. Our aim is to classify these emotions into positive, negative or neutral class. Users react with “Like” more than other reactions as it is easy to perform although it does not represent a specific sentiment polarity that can be classified as positive or negative [34]. Correlation among “Like” and other reactions can be expressed as-

- Strongly positive correlation with “Love” and “Wow”.
- Weakly positive correlation with “Sad” and “Angry”.

Although “Like” reaction is the most common, we have considered this as low-effort data from users and ignored it while classifying the sentiment polarity of a post. Furthermore,

we have observed that people reacted “Haha” reaction in any funny, sarcastic posts more than any other reactions. Therefore, we can’t polarize post sentiment in either positive or negative category based on “Haha” reaction.

We have used **Facebook Graph API** [35] implemented by our own Python script to collect data regularly from some popular Bengali Facebook public pages. We have collected 6244 Facebook posts which were pre-processed afterwards to validate them as proper text data. The pre-processing stage includes the filtering of any kind of hyperlink, special characters, duplicate post and non Bengali phonetics. This filtering shrunk the volume of our data set to 4317 posts. We stored this data into database which contains following columns - page type, page post text, and reaction counts of - “like”, “love”, “wow”, “sad”, “angry” and “haha”. Fig. 4.1 demonstrates the total flow of data collection and corpus preparation from Facebook posts.

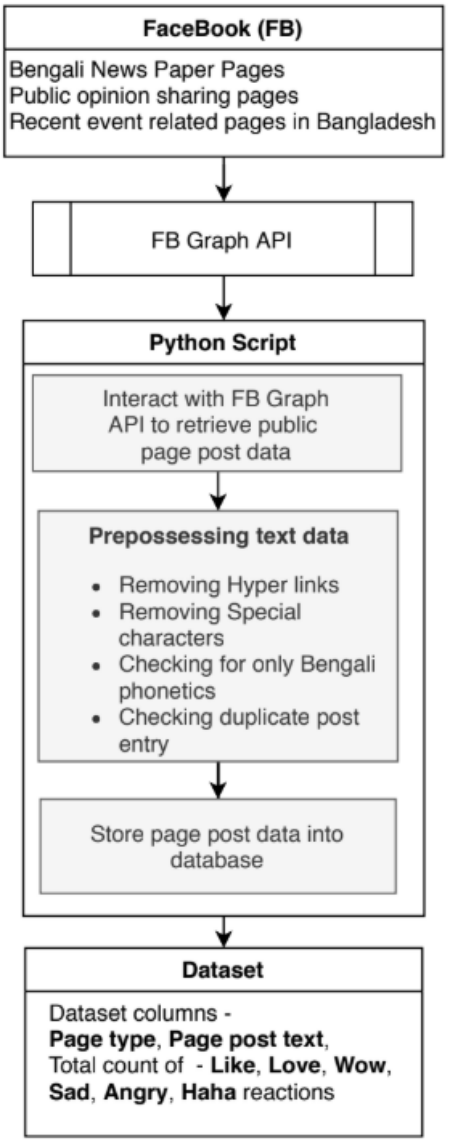

FIGURE 4.1: Flow of data collection and corpus preparation from Facebook post.

To prepare positive and negative post documents from this database we had to categorize multiple reactions into either positive or negative. Here, “Love” and “Wow” reactions represent positive polarity and on the other hand “Sad” and “Angry” reactions represent negative polarity. We considered total count of “Love” and “Wow” reactions as summation of total positive reactions, and total count of “Sad” and “Angry” reactions as summation of total negative reactions. Comparing the total numbers of positive and negative reactions of a post, we categorized it accordingly. This process is summarized in Algorithm 1. Here, we have not categorize a post’s sentiment if-

- the total number of “Haha” reaction is greater than positive or negative reactions.
- the total number of positive and negative reactions are same or both are zero.

The procedure to determine whether a post is categorized or not is shown in Algorithm 2. After this procedure we have 3,193 posts where majority reaction counts are -

- Love: 1,162
- Wow: 529
- Sad: 1,007
- Angry: 495.

So, finally we have 1,691 posts with positive polarity and 1,502 posts with negative polarity. To keep equal polarity data, we finally stored 1,500 posts per sentiment polarity (positive and negative).

We’ve manually constructed a set of neutral documents containing 3500 Bengali sentences. Socian Ltd. [36] provided a public corpus containing 4,000 labeled Bengali sentences according their sentiment polarity, either positive or negative which contains equal distribution of labeled data. They have collected this corpus from different social media platforms, news paper sites and blogs. We included this data set with our prepared corpus. This way finally we managed to prepare a corpus of 10500 posts, 3500 documents for each sentiment polarity - positive, negative and neutral.

4.1.2 Model Generation

In this section we’ll describe about the procedure we used to generate numeric vector models from out textual data-set. For this purpose we’ve chosen to work with TF-IDF averaged *word2vec* and *doc2vec* models.

Algorithm 1 Preparing Positive/Negative Documents from Facebook Page Posts

```
1: procedure ParsePosts(posts)
2:     for each post do
3:         positive ← count(Love) + count(Wow)
4:         negative ← count(Sad) + count(Angry)
5:         if Categorizable() = false then
6:             skip to the next post
7:         else if positive > negative then
8:             save post text into positive.txt
9:         else
10:            save post text into negative.txt
11:        end if
12:    end for
13: end procedure
```

Algorithm 2 Checking a post is either categorizable or not

```
1: procedure Categorizable()
2:     if count(Haha) > positive or negative then
3:         return false
4:     else if positive = negative then
5:         return false
6:     else if positive = negative = 0 then
7:         return false
8:     else
9:         return true
10:    end if
11: end procedure
```

4.1.2.1 TF-IDF Averaged Word2vec Model

Word2vec model represents numeric vector for the words in a document. Here each word is represented as a vector where similar words have closer values to it. First we constructed gensim *word2vec* [23] model using our prepared corpus.

Parameters used to train gensim word2vec model-

- **size :** 100 ; word vectors dimensionality
- **window :** 25 ; max distance between focus and predicted word in sentence
- **min_count :** 1 ; word frequency to ignore below this
- **workers :** 20 ; worker threads used for training
- **alpha :** 0.03 ; initial learning rate
- **min_alpha :** 0.02 ; min learning rate over training progress
- **training iteration :** 60

After training the word2vec model, it contains a vocabulary of 23574 unique words. We used min_count 1 to keep all the words in vocabulary. We had to find a way to get document vector from word vectors. Following approaches are consider for this purpose-

- **Word2vec vectors average:** It's a simple approach to take the average of all word vectors from a document to represent document vector.
- **Word2vec vectors average using TF-IDF:** This is the best approach to find document vector from word vectors. Firstly word vectors are multiplied with their corresponding TF-IDF scores and then the average vector represents document vector.

We chose to work with TF-IDF approach, hence mentioning the outcome document vector model as TF-IDF averaged *word2vec* model. TF-IDF means "Term Frequency - Inverse Data Frequency". Here TF provides the frequency of a word in each document in a data-set. It can be represented by the ratio of a word appearance in a document with the total number of words in that document. IDF is used to calculate the weight of rare words across all documents in a corpus. Lets consider a term t from a document d in a document set. Then the formula to find TF-IDF score for that term -

$$TFIDF(t,d) = TF(t,d) * IDF(t) \tag{4.1}$$

where the IDF is calculated as -

$$IDF(t) = log[\frac{n}{DF(t)}] + 1 \tag{4.2}$$

Here n is the number of documents in data-set. DF(t) is the document frequency of t, the document frequency is the number of documents in the document set that contain the term t. The effect of adding "1" to the IDF in the equation above is that terms with zero IDF, i.e., terms that occur in all documents in a training set, will not be entirely ignored.

We chose to implement TD-IDF averaged document vector from word vectors. If we consider a document D and n number of word vectors from that document as W1, W2, ..., Wn , then the document vector Dvec using TD-IDF averaged -

$$Dvec = \frac{W1 * TFIDF(W1,D) + W2 * TFIDF(W2,D) + ... + Wn * TFIDF(Wn,D)]}{n} \tag{4.3}$$

Some example sentences from our data-set are presented with their corresponding TF-IDF averaged document vector using *word2vec* model.

- **Positive Bengali Sentence:**

  স্টিভের উদ্ভাবন প্রযুক্তি খাতে এনেছে দারুণ পরিবর্তন

  Corresponding TF-IDF averaged document vector (100 dimension) -

  [0.12677471 0.19230783 0.49904703 ..... -0.28776385 0.31691263 0.02206575]

- **Negative Bengali Sentence:**

  সবাই এখন মুখুশ ধারী আসলে কেও মানবতার জন্য কাজ করে না

  Corresponding TF-IDF averaged document vector (100 dimension) -

  [0.01814755 0.11038729 -0.71053634 ..... -0.4587077 -0.05324249 1.46053586]

- **Neutral Bengali Sentence:**

  আমি কলেজ থেকে স্নাতক পাশ করার পরে বাড়িতে ফিরে যাই এবং তিন বছর আমার পিতামাতার সঙ্গে বসবাস করি

  Corresponding TF-IDF averaged document vector (100 dimension) -

  [0.47712728 1.72919988 0.46817737 ..... 0.01164649 0.28329555 -0.46544328]

4.1.2.2 Doc2vec Model

Creating numerical representation of any document is the goal of *doc2vec* [25]. Here each document or sentence is represented as a vector where similar documents have closer values. We've used our corpus to train *doc2vec* model. All the labeled sentences from our corpus were fed into *doc2vec* model to build its vocabulary. Here each labeled sentence contains a list of Bengali words and a label either "Positive", "Negative" or "Neutral" based on its sentiment polarity. An example of labeled sentences used to train *doc2vec* is -

[['word1', 'word2', 'word3',..., 'last word'], ['label']]

For each document we've used the polarity label with a unique identifier while training the *doc2vec* model, so that later we can observe any document's numeric vector representation. For unique identifier, we used document index. So a positive labeled sentence representation looks like -

[['word1', 'word2', 'word3',..., 'last word'], ['POS_unique_index']]

A negative labeled sentence representation -

[['word1', 'word2', 'word3',..., 'last word'], ['NEG_unique_index']]

And a neutral labeled sentence representation -

[['word1', 'word2', 'word3',..., 'last word'], ['NEU_unique_index']]

Parameters used to train gensim *doc2vec* model-

- **vector_size :** 100 ; feature vector dimension
- **dbow_words :** 1 ; to train word vectors
- **dm :** 0 ; training algorithm PV-DBOW
- **epochs :** 60 ; training epochs over data-set
- **window :** 25 ; max distance between focus and predicted word in document
- **min_count :** 2 ; word frequency to ignore below this
- **workers :** 20 ; worker threads used for training
- **alpha :** 0.03 ; initial learning rate
- **min_alpha :** 0.02 ; min learning rate over training progress

After training the *doc2vec* model, it contains 10500 documents vector and has a vocabulary of 10170 unique words. Using min_count 2 eliminates unimportant words while training the model. Below we are representing some example sentences from our data-set and their corresponding document vector from *doc2vec*.

- **Positive Bengali Sentence:**

  তিনি লেখক হিসেবে পুরো দেশে বিখ্যাত হয়ে ওঠেন

  Corresponding document vector (100 dimension) -

  [0.13282606 -0.19305475 0.3983899 ..... -0.09714048 -0.32621175 -0.2620505]

- **Negative Bengali Sentence:**

  রোহিঙ্গা সমস্যা বাংলাদেশের নিরাপত্তার জন্য ভয়াবহ হুমকি

  Corresponding document vector (100 dimension) -

  [-0.4323732 0.54408896 0.46205854 ..... -0.7783456 0.6751657 -1.1729606]

- **Neutral Bengali Sentence:**

  আপনি কি নিশ্চিত যে আপনি আপনার চাকরি ছেড়ে দিতে চান

  Corresponding document vector (100 dimension) -

  [0.12949668 -0.12175082 -0.27730727 ..... -0.5717205 0.4326126 -0.91951835]

4.1.3 Classifier Design

For sentiment classification using our prepared numeric vector models, we used machine learning approaches i.e. LR, SVM, DT, K-Neighbors, LDA, GaussianNB, SM, LSTM and BLSTM. Keras [29] API has been applied to train and test SM and LSTM, BLSTM deep learning classifiers. Using scikit-learn [33] API, other classifiers - LR, LDA, SVM, K-Neighbors, DT and GaussianNB were implemented. Chosen parameters for each classifier are described below:

- **LSTM classifier:**
  - **Input:** Input constructed with three different layers -
    - First Layer: LSTM cell including 64 hidden nodes and activation "$relu$".
    - Middle Layer: Dropout rate 0.25.
    - Final Layer: 3D Dense layer with activation function "$softmax$" and $kernel\_initializer$="$glorot\_uniform$".
  - **Compilation:** $optimizer$="$adam$", $loss$="$categorical\_crossentropy$" and $matrices$= ["$accuracy$"]
- **BLSTM classifier:**
  - **Input:** Three different layers used to construct input -
    - First Layer: Bidirection layer including LSTM cell with 64 hidden nodes and activation "$relu$".
    - Middle Layer: Dropout rate 0.25.
    - Final Layer: 3D Dense layer with activation function "$softmax$" and $kernel\_initializer$="$glorot\_uniform$".
  - **Compilation:** $optimizer$="$adam$", $loss$="$categorical\_crossentropy$" and $matrices$= ["$accuracy$"]
- **SM classifier:**
  - **Input:** Input contained three different layers -
    - First Layer: Dense layer with $batch\_size$=64, $input\_dim$=100 and activation "$relu$".
    - Middle Layer: Dropout rate 0.25.
    - Final Layer: 3D Dense layer with activation function "$sigmoid$".
  - **Compilation:** $optimizer$="$rmsprop$", $loss$="$categorical\_crossentropy$" and $matrices$= ["$accuracy$"]
- **LR classifier:**
  - $penalty$="$l2''$

- Tolerance for stopping criteria, tol=0.0001
- Max iteration, max_iter=100
- Inverse of regularization strength, C=1.0

- **LDA classifier:**
  - Solver, $solver$=‘‘svd”; Singular value decomposition (svd) is mostly recommended for any data-set having large number of features.
  - Rank estimation threshold in SVD solver, tol=0.0001
  - Shrinkage parameter, $shrinkage$=$None$

- **SVM classifier:**
  - Penalty parameter, C=1.0
  - Tolerance for stopping criterion, tol=0.0001
  - Kernel type, $kernel$=‘‘rbf”
  - Size of the kernel cache (in MB) $cache\_size$=200
  - Max iteration, max_iter=1000

- **K-Neighbors classifier:**
  - Number of neighbors, $n\_neighbors$=5
  - Weight function for prediction, $weights$=‘‘$uniform$”
  - Algorithm for computing NN, $algorithm$=‘‘$auto$”; Most appropriate algorithm is determined by $auto$ function based on the parameters value passed into fit method.
  - Distance metric used for the tree, $metric$=‘‘$minkowski$”
  - Minkowski metric power parameter, p=2; p=2 is completely equivalent to using $euclidean\_distance$.

- **DT classifier:**
  - Split quality measuring function, $criterion$=‘‘$gini''$; supported Gini impurity criteria.
  - Strategic parameter used to select split at each node, $splitter$=‘‘$best''$
  - $random\_state$=$None$

- **GaussianNB classifier:**
  - Prior probability of selective classes, $priors$=$None$
  - $var\_smoothing$=$1e-09$; For calculation stability, part of largest variance from all features is added to variances.

4.1.4 Summary

In this section, we explained about different components of our experimental setup. We discussed about choosing social media Facebook as our primary data source to construct the corpus. Numeric document vector model generation using TF-IDF averaged *word2vec* and *doc2vec* is discussed. Parameter choosing for our employed machine learning classifiers are also focused in this chapter.

4.2 Result and Analysis

To evaluate the effectiveness of our employed ML classifiers, we've applied k-fold cross validation technique and retained performance evaluation scores - accuracy, F1 score, precision and recall from each cross validation steps. For train and test ML classifiers, we've used document vectors gained from *doc2vec* and TF-IDF averaged document vectors from *word2vec*. To refer the computational efficiency of classification, we will use the term performance in this work.

4.2.1 k-Fold Cross Validation

We have applied 10-fold cross validation to justify the performance of selected ML classifiers using our prepared vector models from *doc2vec* and *word2vec*. Both of our vector models contain 10500 document vectors representing the full data-set. Below steps we've followed for applying k-fold cross validation to the ML classifiers for both vector models.

- We've initialized document vectors into python numpy [37] array and named it data array. This array size is 10500 and each item in this array contains 100 dimension vector shape. Corresponding labels for each vectors are loaded into another numpy array named label array.
- We've used same shuffled index for both data and label array using numpy random permutation [38] so that they are randomly distributed over the data-set.
- Using sklearn cross validation score API [39], we provided the ML classifier, data and label array, k-fold as 10, and defined the performance evaluation scoring parameter which we want to retain. Then this API provides expected evaluation scores for 10-fold cross validation and we stored it for our result analysis.

Using the above steps we retained performance evaluation scores for our employed ML classifiers. In following sub sections, we represent 10-fold cross validation results achieved for *word2vec* and *doc2vec* models. All the results are calculated in percentage.

4.2.1.1 10-Fold Cross Validation - TF-IDF Averaged Word2vec

Table-4.1 represents 10-fold accuracy scores with a mean column for TF-IDF averaged document vectors using *word2vec* model.

TABLE 4.1: 10-fold accuracy scores for TF-IDF averaged document vectors (Word2vec)

Classifier	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	Mean
BLSTM	77.81	78.76	76.95	76.95	73.71	79.43	78.19	76.57	76.48	78.57	77.34
LSTM	76.29	78.48	76.38	74.38	73.14	77.24	78.95	77.71	76.19	78.38	76.71
SM	75.9	78.95	74.1	72.38	73.81	76.29	77.62	75.33	75.33	76.67	75.64
SVM	74.19	75.43	70.76	71.81	72.29	72.67	76.38	73.24	71.71	72.48	73.1
LR	71.62	73.24	71.9	69.24	70.19	71.43	73.33	72.29	70.95	73.33	71.75
LDA	71.05	73.14	72.57	69.33	69.71	71.33	72.67	72.48	70.76	73.62	71.67
K-Neighbors	68.38	69.24	67.52	68.1	67.52	70.29	70.67	67.71	67.9	68.95	68.63
GaussianNB	58.38	62.19	64.95	59.24	61.62	63.33	63.33	62.1	60.1	60.57	61.58
DT	59.52	57.52	60.1	56	55.43	58.57	60.48	56.95	57.62	56.76	57.9

Table-4.2 represents all performance evaluation parameter's (accuracy, F1 score, precision, recall) 10-fold mean values for TF-IDF averaged document vectors using *word2vec* model. From this table we can observe that BLSTM has acquired highest accuracy of 77.34%. On other hand K-Neighbors, GaussianNB and DT have provided lowest accuracy below than 70%.

TABLE 4.2: 10-fold mean performance scores for TF-IDF averaged document vectors (Word2vec)

Classifier	Accuracy	F-1 Score	Precision	Recall
BLSTM	77.34	77.19	77.05	77.02
LSTM	76.71	76.19	77	76.51
SM	75.64	74.93	75.14	74.85
SVM	73.1	72.99	74.13	73.1
LR	71.75	71.79	71.93	71.75
LDA	71.67	71.8	72.19	71.67
K-Neighbors	68.63	68.48	69.09	68.63
GaussianNB	61.58	61.42	64.07	61.58
DT	57.9	57.76	57.42	57.51

4.2.1.2 10-Fold Cross Validation - Doc2vec

Table-4.3 represents 10-fold accuracy scores with a mean column for *doc2vec* document vectors.

Table 4.3: 10-fold accuracy scores for *doc2vec* document vectors

Classifier	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	Mean
BLSTM	74.57	77.05	73.71	74	76.67	74.38	74.29	77.05	78.19	75.71	75.56
LSTM	73.14	75.9	74.48	74.76	75.43	73.62	74.57	76.48	77.14	74.19	74.97
SM	73.14	75.24	74.38	72	73.9	71.14	72.38	74.76	76.1	72.48	73.55
SVM	72.38	72.48	72.1	71.52	71.81	72.29	71.24	73.9	75.43	73.52	72.67
LDA	70.38	71.05	71.14	70.48	73.05	69.9	70	72.1	71.52	70	70.96
LR	70.1	70.67	70.95	70.57	72.86	69.33	69.71	73.05	71.62	69.71	70.86
GaussianNB	64.57	64.1	66.19	65.05	65.9	66.1	65.33	66.57	66.29	63.9	65.4
K-Neighbors	58.29	57.62	58.19	58.57	58.57	58.67	57.62	59.9	58.76	57.9	58.41
DT	53.62	49.71	51.43	49.24	50	48.38	51.33	51.52	52.76	52.19	51.02

Table-4.4 represents all performance evaluation parameter's (accuracy, F1 score, precision, recall) 10-fold mean values for *doc2vec* document vectors. Here BLSTM has acquired highest accuracy of 75.56% and DT obtained lowest accuracy of 51.02%.

Table 4.4: 10-fold mean performance scores for *doc2vec* document vectors

Classifier	Accuracy	F-1 Score	Precision	Recall
BLSTM	75.56	75.77	75.42	75.74
LSTM	74.97	74.66	74.61	75.19
SM	73.55	73.61	73.58	73.7
SVM	72.67	72.44	72.76	72.67
LDA	70.96	70.77	70.79	70.96
LR	70.86	70.7	70.7	70.86
GaussianNB	65.4	64.9	65.5	65.4
K-Neighbors	58.41	56.19	64.59	58.41
DT	51.02	51.05	50.8	50.9

4.2.2 Doc2vec vs TF-IDF Averaged Word2vec

Document vectorization technique of *doc2vec* is an adaptation of *word2vec*. At first *doc2vec* creates vocabulary by extracting unique words from the provided documents data-set, therefor words are unique across all documents. For generating vector model, *doc2vec* offers two approaches - PV-DM and PV-DBOW. We've built our *doc2vec* model using PV-DBOW. In training phase, it doesn't consider word ordering information in a document. Also term frequency of a word is ignored in training phase. In simple it determines context probability for a given paragraph or document by sampling list of words from it. So we can see that term ordering and rareness of a term are not considered while creating document vector using *doc2vec*. It leads to a problem, common words will appear more often and other words containing more information about the document

will be less frequent, and thus the output document vector will be less informative for topic classification.

Now we'll discuss how TF-IDF vectorization overcomes these drawbacks. TF-IDF considers the term frequency of a word and makes a balance with its inverse document frequency. That means mostly common words across all documents will gain low scores. Rare words representing the topic of a document will gain higher scores and they will have more impact on the output document vector.

According to this research [40], performance of *doc2vec* is not remarkable for short length documents. *doc2vec* model is more suitable for very large corpus. On the contrary, TF-IDF is preferable for short text fragment and small or medium size corpus. As our corpus size is small and it contains mostly short length documents, TF-IDF seems the most suitable solution. Our result comparison also indicates that TF-IDF averaged *word2vec* provides more classification accuracy than *doc2vec*.

Table-4.5 represents a comparison of 10-fold mean accuracy scores achieved from TF-IDF averaged document vectors using *word2vec* and document vectors from *doc2vec*.

From this table our observation is, almost all ML classifiers performed slightly better with TF-IDF averaged document vectors than *doc2vec* document vectors. K-Neighbors and DT perform much better with TF-IDF averaged *word2vec*. Only GaussianNB has achieved better result with *doc2vec* model.

TABLE 4.5: Comparison of 10-fold mean accuracy scores gained for TF-IDF averaged *word2vec* and *doc2vec* models

Classifier	Word2vec Accuracy	Doc2vec Accuracy
BLSTM	77.34	75.56
LSTM	76.71	74.97
SM	75.64	73.55
SVM	73.1	72.67
LR	71.75	70.86
LDA	71.67	70.96
K-Neighbors	68.63	58.41
GaussianNB	61.58	65.4
DT	57.9	51.02

4.2.3 Discussion

In our study, we have applied 10-fold cross validation with most common machine learning performance matrices i.e. accuracy, precision, recall, F1 score for the evaluation of engaged ML classifiers. Obtained 10-fold mean performance scores for *word2vec* is represented in TABLE 4.2 and *doc2vec* is represented in TABLE 4.4. Results are sorted

decreasingly based on the classification accuracy achieved by the employed classifiers. According to the data available in TABLE 4.2 for TF-IDF averaged document vectors using *word2vec*, BLSTM has the best performance as it has gained an accuracy of 77.34% whilst DT has attained lowest accuracy which is 57.9% for the corpus we have built in this study. And in TABLE 4.4 for *doc2vec* document vectors, BLSTM has achieved highest accuracy of 75.56% whilst DT performs very poor with an accuracy of 51.02%. Classifiers result accuracy comparison for TF-IDF averaged *word2vec* and *doc2vec* is represented in TABLE 4.5. This table shows that almost all ML classifiers perform slightly better with document vectors constructed using TF-IDF score from *word2vec* model. Only GaussianNB has achieved better result with *doc2vec* document vectors comparing its result with TF-IDF averaged *word2vec*.

The *word*2*vec* algorithm makes distributed semantic representation of words. This idea can be extended for sentences and documents. Instead of learning feature representations for words, system can learn it for sentences or documents. *sentence*2*vec* represents mathematical average of all the word vector representations in a sentence. *doc*2*vec* extends the idea of *sentence*2*vec* or rather *word*2*vec* because sentences can also be considered as documents. For our experiment we required document vectors as our corpus contains documents as a single unit of labeled data and we aimed to classify it. *doc2vec* model gives document vector for each documents we provided while training the model. *word2vec* model only provides word vectors from a document. To make document vector using *word2vec* model, we applied TF-IDF averaged document vector which is mostly used in document classification and data analysis problems using word embedding technologies.

We observed that performance of deep learning approaches are better than regular ML classifiers using document vectors obtained from both *word*2*vec* and *doc*2*vec* model. BLSTM, LSTM and SM classifier are deep learning based approaches we used in this experiment. BLSTM used Sequential model with bidirectional LSTM cell which increases performance of classifier. In sequence classification problem, using the input sequence in first layer and a reverse copy in the second layer provide more context to the classifier network. This improves the learning process and provides faster result.

Among other traditional machine learning approaches, SVM, LR and LDA performs better than K-Neighbors, DT and GaussianNB using both *doc2vec* and *word2vec* model. Naive Bayes (NB) classifier works fine with numerical and textual data. But it has a major limitation. When features are highly correlated, it performs very poorly. It also fails to consider word occurrence frequency in feature vector regarding text classification problem. In our experiment GaussianNB has achieved accuracy of 61.58% with TF-IDF averaged document vector using *word2vec*. But it has achieved good result using *doc2vec* which is 65.4%. Nearest Neighbor classifier is known as effective and non-parametric in nature. But it takes very long time for classification. SVM offers an advantage which is, in over fitting problem, it tends to be fairly robust and can scale up to considerable

dimensionality. SVM achieved good result among other traditional ML classifier using both *doc2vec* and *word2vec* model. With TD-IDF averaged document vector, SVM achieved 73.1% accuracy which is pretty good.

Using a suitable pre-processing, K-Neighbors can achieve very good results. Its performance scales up well with the number of data set, which is not the case for SVM. SVM uses more parameters than LR and DT classifiers as per analysis. It can achieve highest classification precision most of the time. But SVM is very time consuming as it uses more parameters which requires more computation time. LDA is popular for multi-class classification, because it provides low-dimensional views of the data. It should be applied when training sample is small, to avoid high variance problem. Compared to SVM and LDA, LR is computationally efficient.

Deep learning classifiers performance is better that traditional machine learning when the scale of data increases. But with a small data-set, deep learning algorithms don't perform very well. The reason is deep learning approaches need a large amount of data to learn from it in context of classification. High-end machines are suitable for deep learning experiment contrary to traditional machine learning approaches.

4.2.4 Summary

In this section we've shown our experimental result analysis using 10-fold cross validation. Performance evaluation parameters - accuracy, F1-score, precision and recall have been retained from each k-fold validation step and displayed in tabular manner for relevant vector models. Comparison of *doc2vec* and TF-IDF averaged *word2vec* model is discussed. We also analyzed different ML classifiers performance with document vectors obtained from TF-IDF averaged *word2vec* and *doc2vec* models.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Word embedding technologies perform better with large data-set in context of natural language processing. One of the main focus of this research was to identify relevant data source and retrieve categorical data from it which can be applied to supervised machine learning and document embedding problems. Social media platforms are great source of data if we can apply proper filtering and extract valuable information from it. In our experiment, classification accuracy for different classifiers shows that word embedding technologies have enough potential if implemented properly.

5.2 Limitations

Classifying human sentiment has many limitations. First we want to discuss some limitations that depend on mainly human interactions and their believe, which also varies with time and place.

- **Person's perspective:** As we are taking about opinions, it is the nature of human to have different perspective about anything. It's difficult to mine and categorize large amount of data sample when attempting to analyze opinion or sentiment from it.
- **Time and Place:** An opinion may have different meaning and sentiment based on time and place. A demand of one country's people may not have any positive impact on other countries.
- **Group and Organizational Impact:** Religion and politics also have impact on human sentiment. Based on peoples believe/group/organization, their sentiment on a topic can vary.

Other limitations are related to data collection, filtering and system design.

- **Data Source Availability:** Availability of Bengali corpus for sentiment analysis is not that high. For this study we did not found any standard classified Bengali text corpus. That's why we have to create our own corpus.
- **Noisy Data:** We have created our own corpus by parsing social media content and that contains lots of noisy data. We removed those noisy data sometimes manually and sometimes pragmatically based on predefined filtering rules.
- **Bengali Phonetics based Filtering:** In this experiment we only worked with texts containing Bengali phonetics, which filtered out Romanized Bengali texts. But people often use Romanized text to write their thoughts.

5.3 Future Work

Although our corpus currently constructed with the polarity of sentiment, it is a definite possibility that multi-class model can be prepared given enough time and larger volumes of data.

- **Identify Different Human Emotions:** A textual data can represent very specific state of emotions like - happiness, sadness, fear, anger, surprise etc. In future, we can work with these multiple class classification instead of just identifying sentiment polarity.
- **Scoring Multiple Emotions:** Representing a document with percentage of emotions can be an excellent improvement.
- **Romanized Bengali Texts Classification:** We pre-processed dataset to get the text containing only Bengali phonetics which filtered out Romanized Bengali texts. This narrowed down our dataset and also the scope to work with Latin letters used to write Bengali sentences (Romanized Bengali text).
- **Work with Emoticons:** Filtering special characters removed any kind of emoticons used in the textual post, but emoticon plays a vital role in sentiment expression. We are intending to work with emoticons in our next research work involving sentiment analysis.

Bibliography

[1] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[2] Andrew M Dai, Christopher Olah, and Quoc V Le. Document embedding with paragraph vectors. *arXiv preprint arXiv:1507.07998*, 2015.

[3] A gentle introduction to Doc2Vec. `https://medium.com/wisio/a-gentle-introduction-to-doc2vec-db3e8c0cce5e`, 2020. (Visited on 02/03/2020).

[4] Wikipedia - LSTM. `https://en.wikipedia.org/wiki/Long_short-term_memory`, 2018. (Visited on 10/27/2018).

[5] Deep Dive into Bidirectional LSTM. `https://www.i2tutorials.com/technology/deep-dive-into-bidirectional-lstm/`, 2020. (Visited on 02/03/2020).

[6] Erik Cambria. Affective computing and sentiment analysis. *IEEE Intelligent Systems*, 31(2):102–107, 2016.

[7] Rui Gaspar, Cláudia Pedro, Panos Panagiotopoulos, and Beate Seibt. Beyond positive or negative: Qualitative sentiment analysis of social media reactions to unexpected stressful events. *Computers in Human Behavior*, 56:179–191, 2016.

[8] Social Media Statistics & Facts. `https://www.statista.com/topics/1164/social-networks/`, 2018. (Visited on 10/27/2018).

[9] Social Media Stats Bangladesh. `http://gs.statcounter.com/social-media-stats/all/bangladesh`, 2018. (Visited on 10/27/2018).

[10] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135, 2008.

[11] Sentiment analysis. `https://en.wikipedia.org/wiki/Sentiment_analysis`, 2018. (Visited on 10/27/2018).

[12] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

[13] Peter D. Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, pages 417–424, 2002. ISSN 0738467X.

[14] Kushal Dave, Steve Lawrence, and David M Pennock. Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. In *Proceedings of the 12th international conference on World Wide Web (WWW '03)*, pages 519–528, 2003.

[15] Bo Pang, Lillian Lee, Z. A. Bán, Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Proceedings of the Conference on Empirical Methods in Natural Language Processing. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 48(1):49–55, 2002.

[16] T. Wilson, J. Wiebe, and P. Hoffman. Recognizing contextual polarity in phrase level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354, 2005.

[17] Amandeep Kaur and Vishal Gupta. A Survey on Sentiment Analysis and Opinion Mining Techniques. *Journal of Emerging Technologies in Web Intelligence*, 5(4): 367–371, 2013.

[18] Xi Ouyang, Pan Zhou, Cheng Hua Li, and Lijun Liu. Sentiment analysis using convolutional neural network. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 2359–2364, 2015.

[19] Shaika Chowdhury and Wasifa Chowdhury. Performing sentiment analysis in Bangla microblog posts. In *2014 International Conference on Informatics, Electronics and Vision, ICIEV 2014*, 2014.

[20] Amitava Das and Sivaji Bandyopadhyay. Opinion-polarity identification in bengali. In *International Conference on Computer Processing of Oriental Languages*, pages 169–182, 2010.

[21] Md Al-Amin, Md Saiful Islam, and Shapan Das Uzzal. Sentiment analysis of Bengali comments with Word2Vec and sentiment information of words. In *ECCE 2017 - International Conference on Electrical, Computer and Communication Engineering*, pages 186–190, 2017.

[22] Asif Hassan, Mohammad Rashedul Amin, Abul Kalam Al Azad, and Nabeel Mohammed. Sentiment analysis on bangla and romanized bangla text using deep recurrent models. In *IWCI 2016 - 2016 International Workshop on Computational Intelligence*, pages 51–56, 2017.

[23] Word2Vec. https://code.google.com/archive/p/word2vec/, 2018. (Visited on 10/27/2018).

[24] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*, 2017.

[25] Doc2vec paragraph embeddings. https://radimrehurek.com/gensim/models/doc2vec.html, 2018. (Visited on 10/27/2018).

[26] Alexander Genkin, David D Lewis, and David Madigan. Large-scale bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304, 2007.

[27] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[28] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[29] Keras: The Python Deep Learning library. https://keras.io, 2018. (Visited on 10/27/2018).

[30] Foster Provost and Ron Kohavi. On applied research in machine learning. In *Machine learning*, pages 127–132, 1998.

[31] David Martin Ward Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *International Journal of Machine Learning Technology*, 2(1):37–63, 2011.

[32] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[33] Scikit-learn - Machine Learning in Python. https://scikit-learn.org, 2018. (Visited on 10/27/2018).

[34] Facebook Reactions. http://minimaxir.com/2016/06/interactive-reactions/, 2018. (Visited on 10/27/2018).

[35] Facebook Graph API. https://developers.facebook.com/docs/graph-api/, 2018. (Visited on 10/27/2018).

[36] Socian Bangla Sentiment Dataset. https://github.com/socianltd/socian-bangla-sentiment-dataset-labeled/, 2018. (Visited on 10/27/2018).

[37] NumPy. https://numpy.org/, 2020. (Visited on 02/03/2020).

[38] Numpy Random Permutation. https://numpy.org/devdocs/reference/random/generated/numpy.random.permutation.html, 2020. (Visited on 02/03/2020).

[39] Sklearn Cross Validation Score. `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html`, 2020. (Visited on 02/03/2020).

[40] Cedric De Boom, Steven Van Canneyt, Steven Bohez, Thomas Demeester, and Bart Dhoedt. Learning semantic similarity for very short texts. In *2015 ieee international conference on data mining workshop (icdmw)*, pages 1229–1234. IEEE, 2015.

Appendix A

My Publications

1. Hoque, M. T., Rifat-Ut-Tauwab, M., Kabir, M. F., Sarker, F., Huda, M. N., and Abdullah-Al-Mamun, K. (2016, May). Automated Bangla sign language translation system: Prospects, limitations and applications. In 2016 5th International Conference on Informatics, Electronics and Vision (ICIEV) (pp. 856-862). IEEE.

2. Hoque, M. T., Islam, A., Ahmed, E., Mamun, K. A., and Huda, M. N. (2019, February). Analyzing Performance of Different Machine Learning Approaches With Doc2vec for Classifying Sentiment of Bengali Natural Language. In 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE) (pp. 1-5). IEEE.