Identifying Emotions for Code Mixed Hindi–English Tweets

MSc Research Project
Data Analytics

Sanket Sonu
Student ID: x19206071

School of Computing
National College of Ireland

Supervisor: Dr. Rejwanul Haque

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Sanket Sonu
Student ID:	x19206071
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Dr. Rejwanul Haque
Submission Due Date:	16/12/2021
Project Title:	Identifying Emotions for Code Mixed Hindi–English Tweets
Word Count:	8213
Page Count:	20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Sanket Sonu
Date:	15th December 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Identifying Emotions for Code Mixed Hindi–English Tweets

Sanket Sonu
x19206071

Abstract

Social media is getting bigger day by day. Billions of people use social media on daily basis. Billions of posts are posted every day on Twitter, Facebook, Instagram, etc. which have billions of comments on them. These posts and comments show the emotion of the user. Many companies use these data to find hidden insights about their products by analysing the emotions of the user. Detecting emotions out of monolingual texts such as English texts is easy to process because of a wide variety of pre-trained models introduced by Google, Facebook, etc. However, when trying to detect emotions for Code Mixed Hindi–English texts are complex, and not much research has been proposed. These bilingual Code Mixed Hindi–English texts are a mixture of 2 languages such as English and Hindi, nowadays user also uses English alphabets to write Hindi words. There is no spelling checker or supported library for processing transliterate Hindi words, which results in less accuracy by any machine or deep learning models. This project is using Twitter's data that has been extracted using the official Tweepy API released by Twitter. The research paper will use the different supervised machine and deep learning models for predicting 7 emotions which are *'Happy', 'Sad', 'Angry', 'Fear', 'Disgust', 'Surprise', or 'No emotions'*. This research will use the various supervised machine and deep learning models such as SVC, Multinomial Naive Bayes, Logistic Regression, Random Forest, CNN, and LSTM. This study will also propose a few easy and effective methods to clean, and pre-process Code Mixed Hindi–English texts for corpus creation which will provide the effective result when machine and deep learning models are trained using this corpus. The SVC model performed best by providing 73.75% accuracy.

Key Words: Emotion detection, SVC, Logistic Regression, Naive Bayes, Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM).

1 Introduction

1.1 Background and Motivation

The world is transforming their living style and people love to share their daily day-to-day lifestyles using social media posts and status. Micro-blogging websites like Twitter and Facebook are famous platforms where normal user and celebrities post their thoughts by posting Tweets and commenting on someone's posts. These posts and comments show the emotion of the user. Nowadays, people are more tend to use Twitter and Facebook for daily news and updates rather than watching television or reading newspapers. People use Twitter to directly give reviews, feedback, or complaint against any products on Twitter. Also, the current ruling government keeps track of tweets posted about them, which shows normal public emotions for the current government, which can be used in a positive way to improve and overcome current problems which are faced by normal people in daily life. This can help the current ruling government to win the next election and can also give opponents an opportunity to tackle the problem and create new rules which can make a majority of the population happy, safe and comfortable. This was not possible before social media because the majority of the population cannot interact with the government one by one and share the problem. People also tweet about their favourite celebrities, sportsperson, politicians, etc. These tweets show emotions such as *'Happy', 'Sad', 'Angry', 'Fear', 'Disgust', 'Surprise', or 'No emotions'.*

There is much research that shows good results when detecting emotions out of monolingual texts such as English. Salam and Gupta (2018) and Tiwari and Sinha (2020) shows how they performed emotion detection from monolingual texts. Users posts tweets which has words with spelling mistakes or shorthands or acronyms. There are a few python libraries that detect all these words and correct them in proper English words for example *'govt'* can be corrected as *'government'*, *'happyy'* can be corrected as *'happy'*. All those words are in English vocabulary, which makes the pre-processing and model training easy and smooth. There are many pre-trained models such as BERT and ALBERT from Google, fastText and RoBERTa from Facebook, codeBERT from Microsoft, GPT from OpenAI, and many more, which are pre-trained using English newspaper data, Wikipedia data, and many English articles available on the internet. These famous models support many other famous languages as well such as Spanish, German, French, and a few more. However, Hindi is not supported by any of these famous pre-trained models.

India is the 2nd largest country in terms of population, i.e, 1.35 billion and there are more than 1600 native languages, where more than 50% of the population have Hindi as their native language. Most Indian use Code Mixed Hindi–English texts for their social media. Hindi native speakers use transliterate words for social media such as Hindi texts can be written using English alphabets. India has a very huge crowd who use social media and simple common Hindi words can be written in multiple ways such as 'No' which is an English word that means 'Ni' in Hindi, and this Hindi words can be written in multiple ways such as 'nahi', 'ny', 'nehi', 'nae', 'nhi', 'nahy', etc. However, there are no libraries and models to detect these words which can correct them into 1 word. This makes the pre-processing and training of the model very complex because of manual interaction with the data. Below are a few examples of tweets used in this project:

Tweet 1: yaar india kahi t20 world cup se bahr na ho jae mjhe bht darr lag rha hai
Translation: Man India should not be out of the t20 World Cup I am very scared

Tweet 2: agle mahine se ipl start hone wala hai mast maaza aaega
Translation: IPL is going to start from next month it will be fun

Tweet 3: kya faltu government hai sharam v ni aati
Translation: what a useless government shame on them

These 3 tweets show how users transliterate Hindi texts using English words. Tweet 1 shows the 'Fear' emotion, tweet 2 shows the 'Happy' emotion and tweet 3 shows the 'Disgusting' emotion. This research uses more than 9165 Code Mixed Hindi–English tweets. All these tweets are annotated manually during corpus creation. All tweets are divided into 7 emotions as *'Happy', 'Sad', 'Angry', 'Fear', 'Disgust', 'Surprise', or 'No emotions'*(confusing tweets or tweets which don't show any emotions).

1.2 Research Question and Objectives

'There is no publicly good quality data and pre-trained models for Code Mixed Hindi–English data, also it is expensive to collect, pre-process and run various machine and deep learning models. How efficiently can machine and deep learning models predict emotions from Code Mixed Hindi–English tweets?'

The main objective of this research is to find a good supervised machine or deep learning models, which can predict emotions for Code Mixed Hindi–English data and classify them into 7 emotions such as *'Happy', 'Sad', 'Angry', 'Fear', 'Disgust', 'Surprise', or 'No emotions'.* Along with this, the paper will also discuss a few easy and necessary pre-processing steps which can be done using code and by manual interaction to the dataset. This will help in corpus creation which can be used by machine and deep learning models to predict emotions, these manual interaction and pre-processing will increase the performance of the model.

1.3 Structure of the paper

This research paper is divided into a few sections. Section 2 deals with the Related Work part which demonstrates the research performed by other researchers on the same or related topic. Section 3 will deal with Research Methodologies and implementation, which will demonstrate all the technical aspects of the research. Section 4 will deal with the Results evaluation which demonstrates how accurately the models performed in detecting emotions. Section 5 will demonstrate conclusions and future work. Section 6 demonstrates the acknowledgment part and Section 7 shows the list of all the references used in this research project.

2 Related Work

There are many scholars who have worked on this problem domain. This section will show a few related works already performed by scholars. This section will be divided into 3 sub-parts. The first subsection will show how few research papers used machine learning to solve the problem. The second subsection will deal with research papers that only used deep learning models to solve the problem. The third subsection will deal with research papers that used both machine and deep learning models to solve the problem. There are few papers that worked on the monolingual text and the rest papers worked on bilingual code mixed texts.

2.1 Emotion detection using Machine Learning models

The emotion detection problem can be solved using various supervised machine learning models. Vijay et al. (2018) worked on bilingual Code Mixed Hindi–English tweets. They used Twitter's API and collected 350K tweets. During pre-processing, they cleaned the data by removing tweets that are completely in English or Hindi texts. They filtered out 5.5K tweets which are only bilingual Hinglish code mixed tweets and later created a corpus that includes 2866 tweets. The annotated each of 2866 tweets manually and used emotion classes such as Happy, Sad, Angry, Fear, Disgust, or Surprise. They have used Word N-Grams, Character N-Grams, Emoticons, Punctuation's, Repetitive words, Uppercase words, and Intensifiers to convert text data into feature vectors. They used these vectors and trained the SVM model. By using char n-grams feature extraction, they were able to boost the accuracy by 16%. This paper is a very good example of how text classification needs to be performed as they have used basic steps to predict emotions from bilingual code mixed texts.

Micro-blogging websites such as Facebook and Twitter data can be classified as positive, neutral, or negative as well. Sulthana et al. (2018) used monolingual English Twitter's data for their research project. They used official Twitter's API for data collection and annotated each tweet as positive, neutral, or negative using keywords, i.e., they used specific #tags example 'happy' to search tweets using Twitter API and labeled all those tweets as 'positive'. Like that they annotated 14K tweets. They have used 3 supervised machine learning models as SVM, Naive Bayes, and Linear Regression. They have also used 10 fold cross-validation for their models. The authors used Linear Regression to predict the polarity of the tweets, which outperformed SVM, and Naive Bayes models. The Linear Regression approach provided 85% accuracy which is more than SVM and Naive Bayes. However, there is a limitation in this research that is annotating tweets using keywords, as there are many words that show different sentiment based on the line context, example - *'happy'* keywords can be used like this *'Why are you happy? You are ugly and looks like a chimpanzee'.* Tweets like this can be annotated as 'Positive' because of *'happy'* keywords, which is actually a 'Negative' tweet. This annotation technique is the one limitation that can not be 100% sure about labeling the tweets.

Comments of micro-blogging websites such as Twitter, Facebook, and YouTube for the agriculture domain have been collected by Singh et al. (2019). They collected bilingual Code Mixed English–Punjabi comments. They created a corpus of 11K comments. The authors used a regular expression to clean the data by removing punctuation, #tags, links. Authors used an abbreviation list to change the spelling of a few words such as *'nyc'* can be corrected as *'nice'.* They collected a list of words from the sentiment repository and used them for annotating comments and labeled them as positive, neutral, or negative. The authors used SVM and Naive Bayes for predicting sentiments. They used both uni-grams and n-grams feature extraction techniques and they were able to achieve a better result for the n-grams technique. However, the annotation part is a limitation that can be improved as they have used a list of words to annotate data into positive, neutral, or negative sentiments, which can falsely annotate data because some positive words can be used in negative comments and vice versa.

Facebook data is used by Tiwari and Sinha (2020); Srividya and Sowjanya (2019) for sentiment analysis. In both papers, authors collected monolingual English Facebook data using Facebook's GraphAPI and annotated them as positive, neutral, or negative. Tiwari and Sinha (2020) have performed pre-processing using Java programming language and further used the corpus for training supervised machine learning models. Srividya and Sowjanya (2019) used python for pre-processing and they used supervised machine learning models for sentiment analysis. Both papers have used SVM and Naive Bayes as classification models and both papers were able to achieve 70% accuracy for Naive Bayes and 90% accuracy for the SVM model.

Twitter data has been used by Mandal et al. (2018). Authors have collected 89K tweets and after further pre-processing, they used 5K Code Mixed English–Bengali tweets for their research purpose. They have used a very unique method to annotate data. They created two annotation systems, the first system was for language labeling, and the second system was for sentiment labeling. These systems labeled tweets automatically into positive, neutral, or negative classes. Further, they used manual evaluation of both the system's output. The authors created a few rules for tagging and were able to achieve a 0.90 kappa value. The authors used Naive Bayes, Linear regression, and stochastic gradient descent. The authors were able to achieve good results using this unique polarity tagging system.

2.2 Emotion detection using Deep Learning models

Code Mixed Hindi–English tweets are used by Wadhawan and Aggarwal (2021). They have collected 150K tweets using Twitter's API. After pre-processing they have annotated tweets as happy, sad, angry, fear, disgust, or surprise. They have used #tags to search tweets and used those #tags to label the data, example *'happy'* keywords are used to extract tweets which are having a *'happy'* word in them and annotate all those tweets as 'happy' emotion. They have created their own word embedding system to convert text data into vector representation which can be understood by deep learning models. They have also used Word2Vec and Facebook's FastText for embedding. Further, they used these vectors to train deep learning models such as CNN, LSTM, and Bi-LSTM. They have also used transformer-based models such as BERT, RoBERTa, and ALBERT. BERT model performed best among all of these and they were able to achieve an accuracy of 71%. They have used a good amount of data to train the model, however, they have annotated them based on keywords, which will be a problem because keywords when used with different words can result in different contexts, example *'happy'* keyword can be used in a 'sad' emotion tweet as well.

Twitter data has been used by Younas et al. (2020) and authors collected 20,700 tweets using Twitter's API. After pre-processing, they labeled tweets and divided them into positive, neutral, or negative sentiments. They have used both multilingual and bilingual tweets for their dataset. Further, they used transformers to predict the sentiments. The authors used mBERT and XLM-R transformers. mBert is built with a BERT base with 12 layers of transformers and 768 hidden layers. Authors were able to achieve 71% accuracy for code-mixed English and Roman Urdu texts. The authors used 16 batch sizes, epochs as 3 for both mBERT and XLM-R models. They have used 2e-5 learning rate for the mBERT transformer model and 2e-6 learning rate for the XLM-R transformer model. These hyper-parameter tuning boosted the accuracy and F-score to 71%.

Authors have tried some best deep learning models to detect emotions out of Code Mixed Hindi–English texts. Author Sasidhar et al. (2020) have used data of Vijay et al. (2018) and collected their own data and made a total count of 12K text data, which include 3 emotions such as happy, sad, or angry. Authors have used Word2Vec to generate word vectors from text data. They used both the continuous bag of words and skip-gram methods of Word2Vec for creating word vectors. Authors have used CNN and RNN for text classification. For CNN they used 1-D CNN and for RNN they used LSTM and BiLSTM for text classification. LSTM and BiLSTM performed better than the CNN model and they were able to achieve 81% accuracy on RNN models. They have used a combination of CNN-LSTM and CNN-BiLSTM models, which outperformed normal CNN and RNN models. CNN-LSTM performed well and gave 82% accuracy and CNN-BiLSTM outperformed all the 4 models and they were able to achieve 83% accuracy on the CNN-BiLSTM model. The authors used hyper-parameters for CNN-BiLSTM models like 0.3 dropouts, ReLU activation for 1D-CNN, Tanh for Bi-LSTM, and Softmax for dense layer, optimizer as RMSprop, Loss function as Categorical Cross-Entropy, and batch size as 50.

Sentiment analysis of code mixed text (SACMT) has been experimented by Choudhary et al. (2018). They have collected data from open source, which already has annotation as positive, neutral, or negative. Authors have used 3 datasets for the research purpose such as English tweets, Code Mixed Hindi–English tweets, and Semeval dataset. Authors have used pre-processing using a clustering-based approach to detect the variation of transliterated words. The authors have used the Bi-LSTM model for sentiment analysis. Authors were able to achieve 80% accuracy for the English tweets dataset and 77% accuracy for the code mixed dataset. However, the Semeval dataset was not good with the RNN model and they achieved only 71% accuracy on this dataset. This clustering pre-processing approach used in SACMT outperformed the state-of-the-art technique of sentiment analysis by around 7-8% accuracy and 10% F-score.

Facebook comments are used by Mukherjee (2019) for sentiment analysis with 3 polarity levels, i.e., positive, neutral, or negative. They used 2 famous people's public accounts, i.e., Narendra Modi and Salman Khan as wide varieties of comments are there from various parts of India. They have used LSTM deep learning model for sentiment analysis. This LSTM model used joint-learning from both character and word features. They have pre-processed and corrected some common spelling mistakes using python re library. The LSTM model was trained with a different combination of parameters. They used a combination of hyper-parameters are RMSprop with Categorical Cross-Entropy, RMSprop with Focal Loss, RMSprop with MSE, adamx with MSE, adamx with Focal Loss, and adamx with Categorical Cross Entropy. The last combination of adamx with Categorical Cross Entropy performed best among all other variations and was authors were able to achieve 70% accuracy and 66% F1-score by using RNN's LSTM model for text classification.

The authors have used the same deep learning models for 2 datasets. K. et al. (2018) collected Code Mixed English–Bengali texts dataset from NLP tool contest and they created Tamil movie review dataset manually. The authors tried to compare the performance of the same CNN model on both datasets. The polarity has 3 levels which are positive, neutral, or negative. They have used CNN model for text classification. They have used activation functions such as ReLU, Tanh, and Sigmoid. They used 0.5 dropouts with a batch size of 64, no of filters as 128, and epochs as 200. This performed well for Code Mixed English–Bengali texts, and they achieved 73% accuracy on this dataset, however, the monolingual Tamil movie dataset was not able to perform well on the same CNN model with the same hyper-parameters, which gave only 51% accuracy on Tamil dataset. However, the word vector was generated using word indexing which was the reason for low performance. Using other feature generation methods like Word2Vec can improve the performance for both datasets and they have mentioned this in future work.

3 Emotion detection using both Machine and Deep Learning models

Both supervised machine and deep learning models can be used to predict the emotions from a Code Mixed Hindi–English texts. Singh (2021) has used both supervised machine and deep learning models for their dataset, which they have collected from Twitter's API and comment section of video streaming platforms. They have used 4 labels for their research purpose, i.e., happy, sad, angry, or fear. The authors have annotated the dataset manually. They are using 1600 data for their research. They have performed pre-processing and changed a few Hindi words' spelling. They solved this issue using clustering of skip-grams vectors. The authors used Naive Bayes, LSTM, Sub-words LSTM, and SVM models. The authors used char n-grams and word n-grams for feature extraction. Authors were able to achieve 75% accuracy for the Naive Bayes word-grams model and 77% accuracy for Sub-word LSTM. The only limitation here is less amount data. Less amount of data may be the reason for over-fitting deep learning models, like LSTM and Sub-word LSTM.

In this research, paper Kastrati et al. (2021) authors have used multiple models to predict monolingual Albanian language text. They scrapped data from the Facebook comment section of the COVID-19 topic. Authors manually annotated 10,700 data into 3 polarity levels, i.e., positive, negative, or neutral. After pre-processing, they used both Contextualized and Static word embedding. Authors used SVM, Naive Bayes, Random Forest, and Decision Tree for supervised machine learning models and were able to achieve 70 to 71% accuracy for all the machine learning models. However, when using deep learning models like CNN, BiLSTM, and BERT, they were able to achieve 71-72% accuracy. This 1-2% difference in accuracy is mostly because of different embedding used for different models.

These authors have worked and created their own pre-processing methods. Yadav et al. (2020) collected 6400 data and annotated them as positive, neutral, or negative. For pre-processing they have created their own list of stop-words. Authors have also created their own lemmatization corpus. The authors used text blob to decide the polarity of the texts. Further, authors have used SVM, Naive Bayes, SGD, and XGB for supervised machine learning models and the Bi-LSTM model for supervised deep learning model. They have used TF-IDF vectorizer to extract features from textual data. Further, they used an ensemble model for all 4 machine learning models, to find the best model with the best parameters. The authors were able to achieve 74% accuracy for the ensemble model and 73% accuracy for the Bi-LSTM model. The F-score of the ensemble machine learning model is 0.69, however, F-score for the Bi-LSTM model is only 0.59

Authors Tho et al. (2020) reviewed 230 journal papers from the internet. They mentioned that most of the research papers are using supervised machine learning models than supervised deep learning models. The most popular machine learning models are SVM, Naive Bayes, and Logistic Regression, these models perform best for Code Mixed sentiment analysis. Among all 3 SVM performed best, however, they also tested a few deep learning models like CNN and LSTM but that was not able to give better accuracy and performance than machine learning models.

Authors Ahmad et al. (2019) gave a review on sentiment analysis of Code Mixed texts, especially for Indian languages. They have mentioned that apart from the machine and deep learning models, Lexicon-based sentiment analysis is another approach to solve the problem. The lexicon approach can be used at the document and sentence level. They have mentioned that dictionary or corpus-based methods are followed for Lexicon based approach. They have concluded that lack of lexicon tools, stop-words, spell checkers, grammar checkers are unavailable for Indian languages, which makes the research complex and result in low or average performance on the machine and deep learning models.

In this research paper Mishra et al. (2018), authors have used both machine learning and neural networks for sentiment analysis. They have collected around 24K data and annotated it as positive, neutral, or negative. The authors have created 2 datasets, the first, is Code Mixed Hindi–English data and the second is Code Mixed English–Bengali data. After pre-processing, they have used a TF-IDF vectorizer with character n-grams in the range from 2 to 6 to extract the features. They have used an ensemble voting classifier of three machine learning models SVM, Logistic Regression, and Random Forest. For Run1 they used voting classifier and for RUN2 they used the SVM model, out of which the Run2 SVM model outperformed the Run1 ensemble model. They have used MLP (multi-layer perceptron) and Bi-LSTM models for neural networks. The authors were able to achieve a 0.69 F1-score for the SVM model. However, Bi-LSTM was able to give only 0.54 F1-score and the MLP model showed lesser performance than Bi-LSTM, i.e., 0.53 F1-score.

This research paper Wehrmann et al. (2017) is using Twitter data for sentiment analysis. Authors are using a monolingual and multi-lingual dataset for their research. They have used ’Word-Level Embedding’ and ’Character-Level Embedding’. These embedding techniques are used to extract features from tweets. They have used the SVM and LSTM models for sentiment analysis. Authors, mentioned that machine translation is one of the approaches for sentiment analysis, where tweets of different languages can be translated to the English language, and then can be used for sentiment analysis, however, authors said traditional based SVM approach outperforms machine translation technique and hence they are able to get good results without using machine translation approach for their SVM model.

4 Methodology

The research topic is in the domain of data science. Domain knowledge is the key to performing this research. Knowledge Discovery in Databases (KDD) is one of the dominant approaches in the field of data science. This project is based on Knowledge Discovery in Databases (KDD) approach rather than Cross-Industry Standard Data Mining method (CRISP-DM) because CRISP-DM is for an industry-based approach which is required for deployment of the project for business applications. This research project’s main objective is to focus on Knowledge gain while working on different data selection, pre-processing, feature extraction, and model selection with accurate result evaluation. There are no fixed algorithms or implementation to *’Identifying Emotions for Code Mixed Hindi—English Tweets’*, all the processes are selected based on the requirement, and the knowledge will be used to incorporate into another complex system for further action.

4.1 Data Collection

This research project required Code Mixed Hindi–English tweets, which have to be labeled with 7 emotions, such as *’Happy’, ’Sad’, ’Angry’, ’Fear’, ’Disgust’, ’Surprise’, or ’No emotions’*. However, there are many datasets available online for sentiment analysis but those are monolingual English texts, and they have limited labels of sentiments such as positive, negative, or neutral. The unique combination of emotions requires a good amount of data to train the supervised machine and deep learning models. Data has been collected from Twitter’s database using the official API *’Tweepy’* released by Twitter. In order to extract data from the Twitter database, first, it is necessary to sign-up on *’https://developer.twitter.com’* account. After account creation, there is an option to

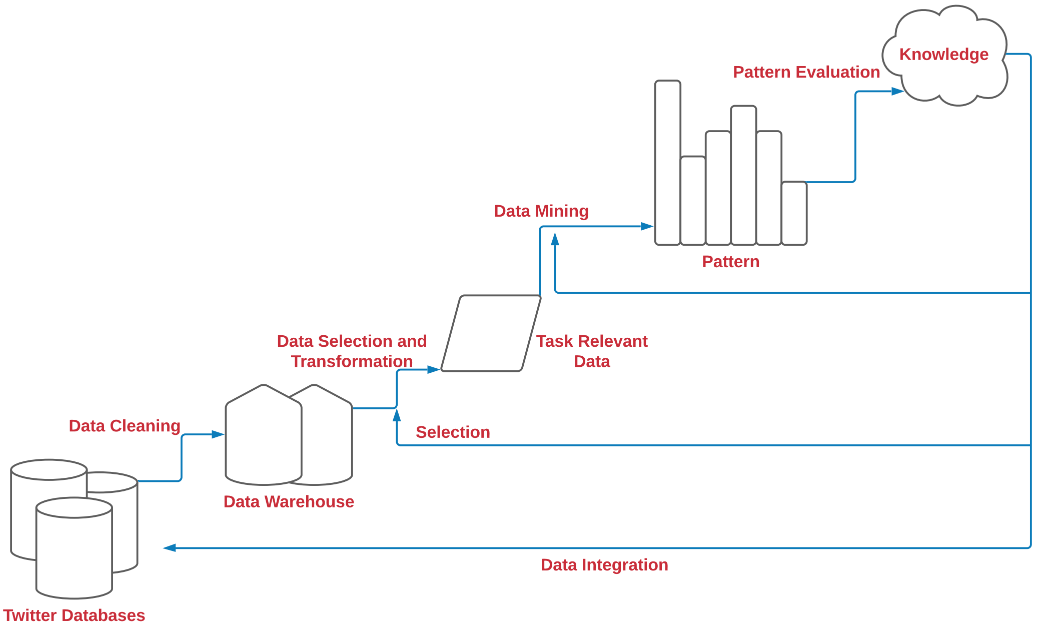


Figure 1: KDD for Identifying Emotions using Twitter data

create an app, which will allow us to interact with Twitter's database. This app will provide *'Consumer Key', 'Consumer Secret', 'Access Token', and 'Access Token Secret'*. In order to use *'Tweepy'* API to extract data, we need to have *'Keys and Tokens'*. Along with this few unique parameters, like language, and #tags were used to search for good quality data, which will be explained briefly in the implementation section.

4.2 Data Pre-processing

Data collection from any website or database required a good cleaning and pre-processing to utilise and get the best result out of it, as normally there are many noises present in Twitter's tweets such as #tags, @, RT, links, photos, videos, emojis, etc. All these noises can lead to the undesirable output. Also, this project requires Code Mixed Hindi–English tweets. While collecting data, there were many tweets which was blank or just 1-2 words, or tweets just in the English language, or tweets in complete Hindi language. All those unnecessary tweets were removed from the dataset. There were many Hindi or English words, which was written with different spellings for example *'mujhe'* which means *'me'* in English, can be written as *'mujh', 'mujhee', 'mjhe', 'muje', 'muj'*, all these words sounds same and have the same meaning. These words are filtered out and changed to 1 specific word using Python's *'Regular Expression - re library'*.

4.3 Feature Extraction

Data is in the form of words or sentences and machine learning models are not able to understand words, these models require data in the numerical form. It is important to convert words and sentences into numerical vectors, this process is known as vectorization or word embedding in Natural Language Processing (NLP). Tweets are required to be divided into smaller parts or words, this process is known as Tokenization. Tokens help in developing the model by understanding the context of the sentence by analysing the word sequence. During vectorization, the size of the vector will be equal to the number of unique words in the corpus. The vectors will be assigned to each word based on the word frequency in every sentence. These numerical vectors are understandable by machine learning and deep learning models. Also, feature extraction will help and improve the training time by reducing the amount of redundant data. This project uses various vectorizers for machine learning models such as TF-IDF and CountVectorizer. Supervised deep learning model will use word embeddings such as Word2Vec and Doc2Vec.

4.4 Data Transformation

Deep learning models accept input of the same shape and size, which requires padding. All the tweets present in the dataset are of different lengths, some are short sentences and some are long sentences, in order to use these data as an input, padding is required. This project is using CNN and LSTM models, which will accept input of the same shape and size. Since all sentences are not of the same size, a maximum number of words or sentence length can be defined and during the padding process, at the end of each sentence, 0 will be added for each number of words to make the each input tweet of the same size. Example: max number of words assigned for a sentence = 100, and if a sentence is having only 25 words, then the rest of 75 words will be padded with 0, i.e., 75 zeroes will be added at the end of the sentence, this will make all short sentence size equal to 100 words, which will eventually make input shape and size equal. The data need to be divided into train and test sets, this was achieved using scikit-learn's train_test_split(). The data is divided into an 80:20 ratio, i.e., 80% for a train set and 20% for the test set. This is a necessary step to evaluate the performance of the machine and deep learning models on new data, i.e, not used in the training process by the model.

4.5 Models Selection

After all the above steps, it is required to use the good machine and deep learning models for training. This research has used both supervised machine and deep learning models. This is a classification problem, which will have 7 classes. Tho et al. (2020) has reviewed 230 research journals, authors mentioned that few of the machine and deep learning models work best and give a better result when compared with other models. Below are the models used in this research project:

1. **Support Vector Machine:** Support Vector Machine is one of the best text classification algorithm, which determines the best decision boundary which perfectly divides the vectors that belong to a given category and rest vectors which does not belong to that category. SVM accepts text in the form of numerical vectors, which makes the feature extraction process compulsory. SVM algorithm decides where to draw perfect *'hyperplane'* between a group of 2 vectors categories, which will divide the space using *'hyperplane'* into 2 sub-spaces, i.e., one for the group of vectors that belong to that category and one for the group of vectors that do not belong to that category. Moreover, the SVM algorithm works very well with a small or medium amount of training data. SVM tries to eliminate the over-fitting problem by penalizing some data points, which allows entering the group of vectors intentionally, this can be achieved using *'gamma'* parameter. SVM also accepts wide range of *'kernels'* as a parameter such as *'rbf', 'linear', 'poly', 'sigmoid'*. These 2

parameters along with the math behind it, makes this algorithm powerful which perform well for text classification problem.

2. **Multinomial Naive Bayes:** The Naive Bayes algorithm is very simple, yet powerful and fast in the prediction of a classification problem. Naive Bayes works using Bayes' Theorem, which uses the probability of the events. This algorithm assumes all variables independently, which means the presence of one variable will not affect the presence of another one. Naive Bayes is simple and can work well with low to a large amounts of training data, also it is very fast and can be used for real-time predictions, which makes it cheap and can perform well even on low configuration systems. Naive Bayes is highly scalable with a number of data points and predictors. Naive Bayes is not sensitive and can work well with irrelevant features and mislabeled data.

3. **Logistic Regression:** Logistic Regression is a simple classification algorithm, which can be used to find the probability of an event, i.e., whether the event is a success or failure. Logistic Regression can be used for both binary and multinomial problems. This algorithm has no assumptions about class distribution in feature space. Logistic Regression can also be used to classify unknown records, also it explains model coefficients for feature importance. Logistic Regression prevents over-fitting for low dimensional datasets, however high dimensional datasets can overfit, over-fitting can be avoided by using *'L1 and L2 Regularisation'* techniques

4. **Random Forest:** Random Forest is based on a bagging algorithm, which uses an ensemble learning technique. Random Forest creates many trees based on the subsets of the dataset, which combines the output of all the trees. This can prevent over-fitting problems automatically and also improves the accuracy by reducing the variance. Random can handle missing values and feature scaling such as *'standardization and normalization'* are not required. Random Forest performs well with outliers and can handle them automatically, also this algorithm is less affected by noise. Random Forest is a very stable algorithm, which can handle a new data point easily, as it can only impact a single tree, and the rest of the other trees are unaffected.

5. **Convolutional Neural Network (CNN):** CNN is a feed-forward neural network. CNN shows a great performance for various NLP tasks, such as sentiment analysis. CNN can detect patterns of multiple sizes example - 2,3,5 words by varying the size of the kernel and adding their outputs. Patterns can be expressions such as *'I like'*, *'I hate'* and regardless of their position, CNN will identify them in the sentence. This approach makes CNN very suitable for NLP tasks, such as sentiment analysis. CNN is very fast and because it uses GPU computational power.

6. **Long short-term memory (LSTM):** One of the limitations of the neural network is that there is no memory associated with it, which can be problematic for sequential data, such as text. RNN overcomes this limitation, which has a feedback look that works as a memory. This results in a footprint that is left by past inputs. LSTM overcomes this as well by having both long-term and short-term components. This makes LSTM very good for data that has sequence such as text because the meaning of a word depends on the previous words or can have some relation with previous words.

4.6 Model Evaluation:

Model evaluation is very necessary, as the performance of a model needs to be evaluated to improve the model for the next run with some changes if required. Since this research project is using a multi-class classification problem, the results are evaluated using a multi-class Confusion Matrix. The confusion matrix with the classification report will explain everything such as accuracy, F1-score, precision, and recall. Even just by looking at the confusion matrix, it can be easily understood that model is performing well if all the values in *'diagonal'* are very high compared to other rows and columns, this will explain that model is good at detecting classes that it belongs to. There is a problem in just showing overall accuracy, because, for multi-class classification, there may be chances that the model is good at predicting a few classes because of good training data, however, few classes are not showing good accuracy because of less training data or noise, which may result in decreasing overall accuracy of the model. This can be eliminated by looking at particular class accuracy in the classification report.

5 Design Specification

Figure 2 shows the design specification of this research project. Following the data collection, the dataset is cleaned and pre-processed for further use. Later, the dataset is partitioned into 80% train set and 20% test set. After splitting the data, it is necessary to extract features using vectorizer or word embedding, which will create a numerical vector that is understandable by machine and deep learning models. Data has been visualized using Python's *'Matplotlib and Seaborn'*. Extracted data will be used to train machine learning models such as SVM, Naive Bayes, Logistic Regression, and Random Forest. However, for deep learning models, the extracted features need to be tokenized and padded to make the shape and size of the input equal. These tokenized and pad_sequenced data will be used to train deep learning models, such as CNN and LSTM. In the end, all the results will be evaluated and compared to find the best machine or deep learning models for identifying emotions from Code Mixed Hindi–English tweets.

6 Implementation

6.1 Data Collection using Twitter's API

Twitter's official API *'Tweepy'* has been used to scrap data from twitter's database. During data extraction, few of the parameters are used to collect Code Mixed Hindi–English tweets, such as *lang='hi', tweet_mode='extended'*. Language parameter was used to collect data that is in Hindi language, this include both Code Mixed texts and tweets that are completely in Hindi language. Tweet_mode is set to extended to provide full tweets. Various #tags are used to collect specific data, this research project has tried to collect data from famous topics, such as *'cricket', 'ipl', 'bollywood', 'politics', 'congress', 'bjp', 'happy', 'sad', 'angry', 'fear', 'disgust', 'surprise', 'darr lag rha', 'mujhe bachao', 'haha', 'hearbreak', 'love', 'fail', 'amazing', 'unbelievable', demonetization', 'COVID-19', 'dhokha', 'gussa', 'mujhe gussa aa rha', 'cheat', 'shahid', 'lost', 'dil tootna', 'chup reh', 'shut up', 'aukaat me reh', 'aukaat', 'celebrate', 'laugh', 'win', 'vacation', 'good result', 'mujhe mat maaro', 'birthday surprise', 'anniversary surprise', 'dimag kharab mat kr', 'bhoot', 'ghost'*. More than 100K data was collected using Twitter's API. Along with

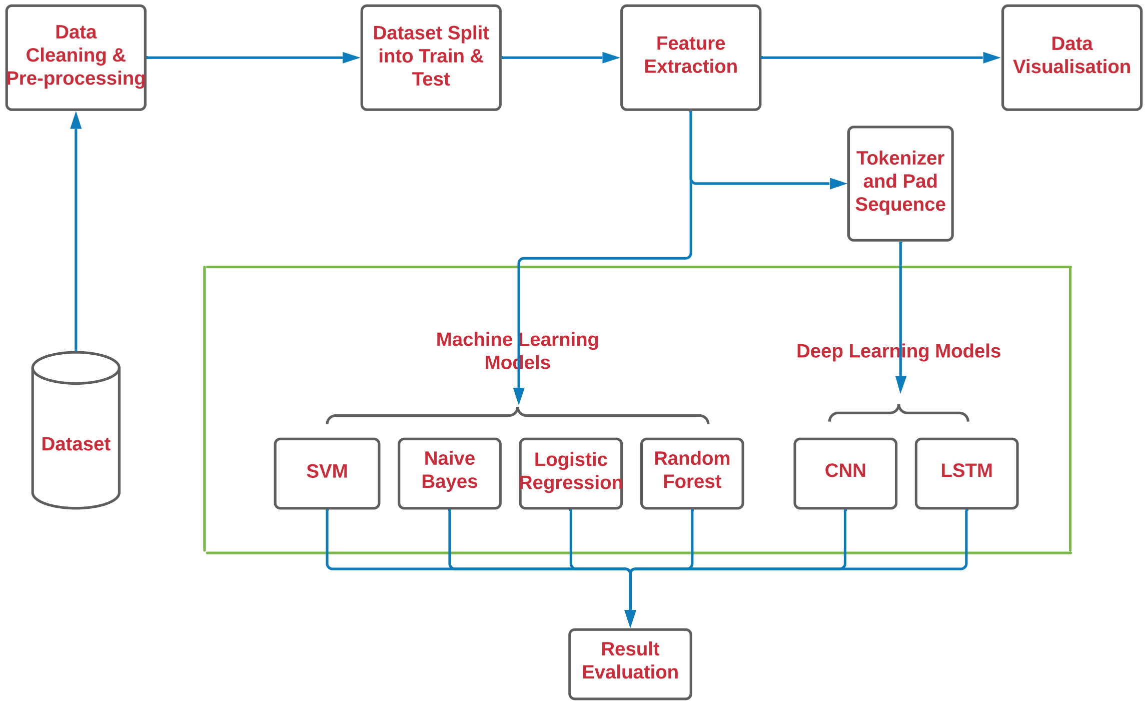


Figure 2: Design Specification

that, this research paper is also using a dataset of 2866 Code Mixed Hindi–English tweets annotated by authors of Vijay et al. (2018) which was available on their public GitHub repository. This dataset was used in Language Technologies Research Centre (LTRC) - IIIT Hyderabad. This dataset is of 2866 annotated tweets and has 6 classes of emotions such as *'Happy', 'Sad', 'Angry', 'Fear', 'Disgust', or 'Surprise'*.

6.2 Data Cleaning:

The dataset has much noise and unwanted tweets, such as tweets that are completely in Hindi or English, few tweets are too small, such as 1 or 2 words tweets. Dataset had tweets which had *'links','RT','@','photos', '#','emojis','flags','symbols','tweets in Hindi script', 'extra white spaces before after and in between sentences'*. All tweets were converted into lower case. All those unwanted noises need to be removed. Using Python's 'Pandas' and 'regular expression - re' libraries, all those unwanted noises were removed. There were many Hindi or English words, which was written many times using different spellings, such as *'tmhara'* which means *'your'* in English, can be written as *'tmharaa', 'tumhara', 'tumara', 'tmhare'* all these words sounds the same and have the same meaning. These words are filtered out and changed to 1 specific word using Python's *'Regular Expression - re library'*. This simple trick improved the accuracy of almost all the machine and deep learning models by around 15%. This was possible because all these 1 words had many instances, which cause the vectorizer to use those instances as a unique word, and hence when the dataset was vectorized, these single instances got assigned as a new word, which increased the overall unique words in the dataset. There were many tweets that had missing space such as *'bhthua' can be corrected as 'bht hua'*. These words were manually cleaned by putting missing white space between 2 such words.

6.3 Annotation:

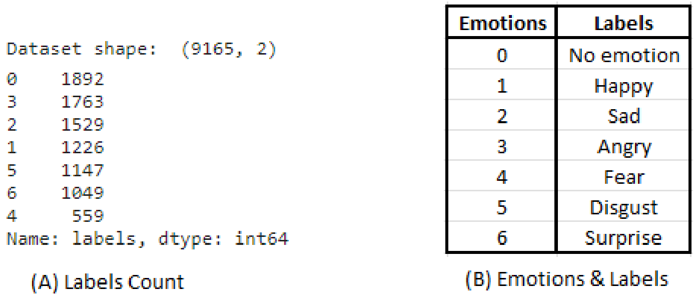
After cleaning the data, the dataset was left with 6299 Code Mixed Hindi–English tweets. Annotation is the most important part of any sentiment or emotion analysis problem. This Mozetič et al. (2016) shows the importance of annotation for sentiment analysis problem. Authors also mentioned inter agreement, which will demonstrate the similarity of annotated data between 2 or more annotators. The dataset was annotated manually by the author(me), who is a native Hindi speaker. This research project uses 7 emotions, which are *'Happy', 'Sad', 'Angry', 'Fear', 'Disgust', 'Surprise', or 'No emotions'*. It is important to check *'Cohen's Kappa and inter-rater agreement'*, for this purpose 2 more native Hindi speakers annotated data, and Kappa values were 0.79 and 0.90.

6.4 Data Pre-processing:

After creating dataset of 9165 (6299 + 2866) Code Mixed Hindi–English tweets with 7 classes, few more noises were cleaned such as punctuation's, numbers, and stop words. There is one open source Rana (accessed on 1st Oct 2021) repository, where author has created list of more than 1036 stop words, which includes both Hindi and English stop words.

6.5 Data Exploration:

Data exploration is needed to explore and find the crucial information's from the data. Figure 3 shows the count of each label in the dataset and the name of the emotion/class associated with the label. Figure 4 shows the length of the tweets and the most frequency of words used in the dataset.



```
Dataset shape:  (9165, 2)
0    1892
3    1763
2    1529
1    1226
5    1147
6    1049
4     559
Name: labels, dtype: int64
```

(A) Labels Count

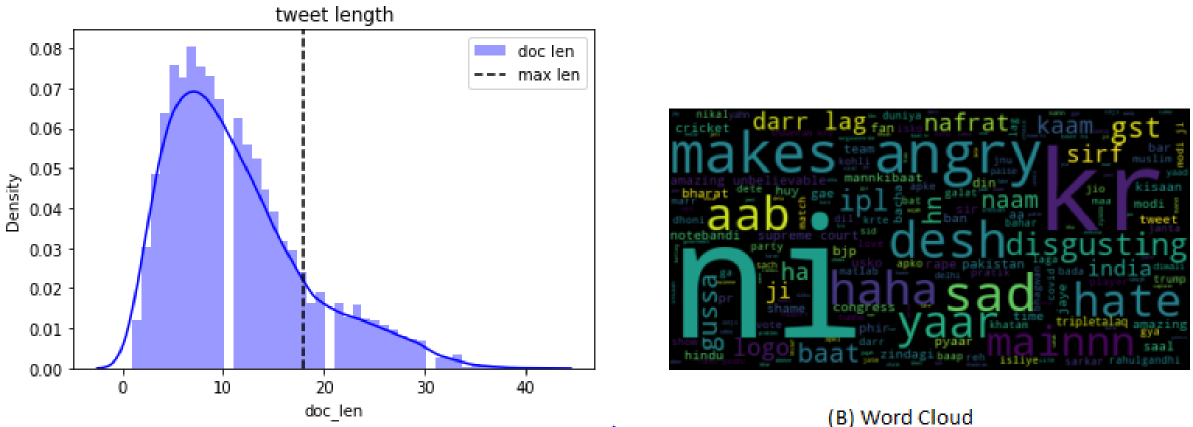
Emotions	Labels
0	No emotion
1	Happy
2	Sad
3	Angry
4	Fear
5	Disgust
6	Surprise

(B) Emotions & Labels

Figure 3: Data Counts and Labels name

6.6 Model Implementation:

After data cleaning, pre-processing, and feature extraction using vectorizers or word embedding, the next step is to train supervised machine and deep learning models to compare the performance of the model for Code Mixed Hindi–English tweets. All machine learning models have been hyper-parameter tuned using GridSearchCV, where value of



(A) Tweet's Length

(B) Word Cloud

Figure 4: Data Visualisation

cross-validation was set to *’cv = 5’*. ’Jupyter notebook’ and ’Google Colab’ has been used to perform all the implementations. Python has been used with many libraries to perform all the tasks.

1. **Support Vector Machine:** First machine learning model is Support Vector Classifier. For this model, the feature has been extracted using 2 vectorizers, i.e., ’TF-IDF Vectorizer - unigrams and bigrams’ and ’Count Vectorizer - unigrams’, also Word Embedding Doc2Vec’s DBOW (Distributed bag of words), DM (Distributed Memory), and combination of both (DBOW + DM) have been used to extract the features. The model is hyper-tuned with parameters such as *kernel = [’rbf’,’linear’,’poly’,’sigmoid’], gamma = [1, 0.1, 0.01, 0.001], and C = [0.1, 1, 10, 100, 1000]*. SVM is able to achieve maximum of 73.75% accuracy for ’TF-IDF Vectorizer’ when *’C’: 100, ’gamma’: 0.01, ’kernel’: ’rbf’*.

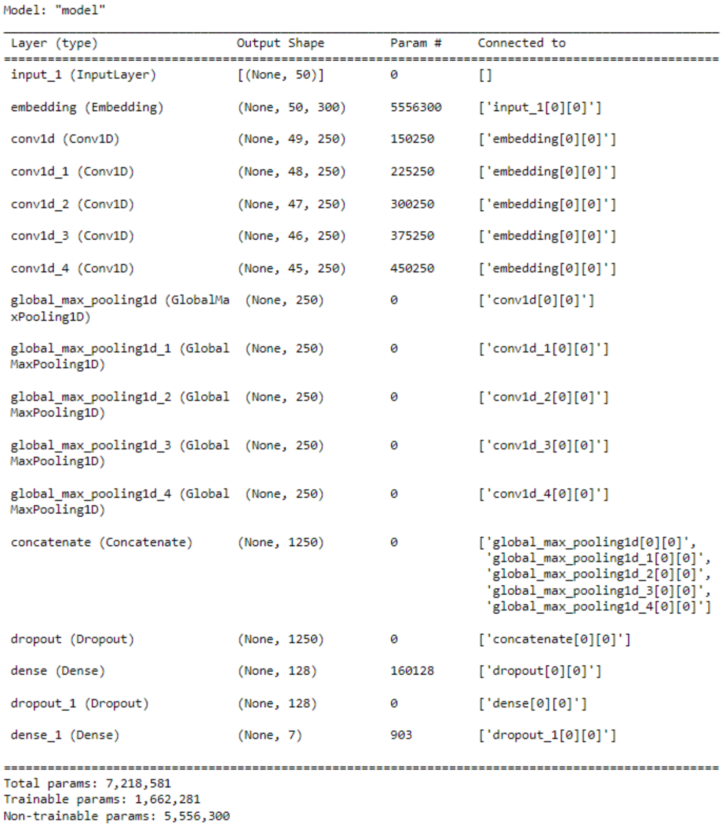
2. **Multinomial Naive Bayes:** For this model, the feature has been extracted using 2 vectorizers, i.e., ’TF-IDF Vectorizer - unigrams and bigrams’ and ’Count Vectorizer - unigrams’. The model is hyper-tuned with parameters such as *clf_alpha = [0, 0.1, 0.2, 0.5, 0.7, 0.9, 1, 1.1, 1.3, 1.5] and fit_prior = [True, False]*. The model is achieved accuracy of 69.06% for ’Count Vectorizer’ when *’alpha’: 1.5, ’fit_prior’: True.*

3. **Logistic Regression:** Logistic Regression has been trained with data which is extracted using 2 vectorizers, i.e., ’TF-IDF Vectorizer - unigrams and bigrams’ and ’Count Vectorizer - unigrams’, also Word Embedding Doc2Vec’s DBOW (Distributed bag of words), DM (Distributed Memory), and combination of both (DBOW + DM). This model is hyper-tuned using parameters such as *C = [0.001,0.01,0.1,0.5,1.0], multi_class = ’multinomial’*. This model is able to achieve maximum of 73.70% accuracy for ’Count Vectorizer’ when *’C’: 0.5.*

4. **Random Forest:** Random Forest Classifier is the last machine learning model used, this model is trained using data which is extracted using 2 vectorizers, i.e., ’TF-IDF Vectorizer - unigrams and bigrams’ and ’Count Vectorizer - unigrams’, also Word Embedding Doc2Vec’s DBOW (Distributed bag of words), DM (Distributed

Memory), and combination of both (DBOW + DM). This model is hyper-tuned using parameter such as *n_estimators = [1, 5, 10, 20, 50, 100]*. The model is able to achieve maximum accuracy of 72.66% for 'TF-IDF Vectorizer' when *'n_estimators': 100*.

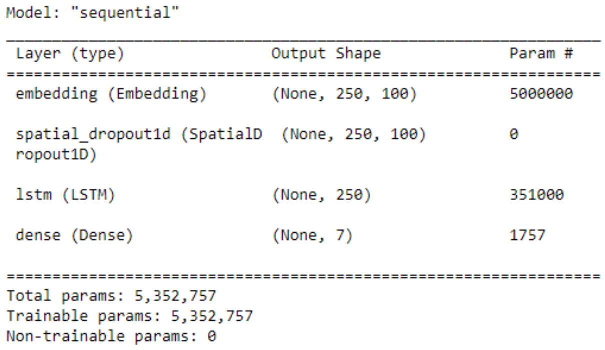
5. **Convolutional Neural Network (CNN):** CNN is the first deep learning models used in this research. First the data is tokenized using NLTK's 'word_tokenize' and then the data-frame is used by Word2Vec word embedding to extract the features. The parameters used are *batch size = 64, epochs = 5, embedding_dim = 300, filter_sizes = [2,3,4,5,6], dropout of 0.1 & 0.5, activation function = 'relu', Dense layer activation = 'softmax', loss = 'categorical_crossentropy', optimizer='adam', and validation_split = 0.1.* CNN model is able to achieve 68.24% accuracy on test set. Figure 5 shows the CNN model's summary.



```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 50)]         0           []

 embedding (Embedding)          (None, 50, 300)      5556300     ['input_1[0][0]']

 conv1d (Conv1D)                (None, 49, 250)      150250      ['embedding[0][0]']

 conv1d_1 (Conv1D)              (None, 48, 250)      225250      ['embedding[0][0]']

 conv1d_2 (Conv1D)              (None, 47, 250)      300250      ['embedding[0][0]']

 conv1d_3 (Conv1D)              (None, 46, 250)      375250      ['embedding[0][0]']

 conv1d_4 (Conv1D)              (None, 45, 250)      450250      ['embedding[0][0]']

 global_max_pooling1d (GlobalMa  (None, 250)         0           ['conv1d[0][0]']
 xPooling1D)

 global_max_pooling1d_1 (Global  (None, 250)         0           ['conv1d_1[0][0]']
 MaxPooling1D)

 global_max_pooling1d_2 (Global  (None, 250)         0           ['conv1d_2[0][0]']
 MaxPooling1D)

 global_max_pooling1d_3 (Global  (None, 250)         0           ['conv1d_3[0][0]']
 MaxPooling1D)

 global_max_pooling1d_4 (Global  (None, 250)         0           ['conv1d_4[0][0]']
 MaxPooling1D)

 concatenate (Concatenate)      (None, 1250)         0           ['global_max_pooling1d[0][0]',
                                                                  'global_max_pooling1d_1[0][0]',
                                                                  'global_max_pooling1d_2[0][0]',
                                                                  'global_max_pooling1d_3[0][0]',
                                                                  'global_max_pooling1d_4[0][0]']

 dropout (Dropout)              (None, 1250)         0           ['concatenate[0][0]']

 dense (Dense)                  (None, 128)          160128      ['dropout[0][0]']

 dropout_1 (Dropout)            (None, 128)          0           ['dense[0][0]']

 dense_1 (Dense)                (None, 7)            903         ['dropout_1[0][0]']

==================================================================================================
Total params: 7,218,581
Trainable params: 1,662,281
Non-trainable params: 5,556,300
```

Figure 5: CNN model's summary

6. **Long short-term memory (LSTM):** LSTM is the last deep learning model used in this research project. The data was tokenized and pad sequenced to make shape and size of input equal. The parameters used for this model are *epochs = 2, batch_size = 64, embedding_dim = 100, SpatialDropout1D = 0.4, Memory unit = 250, LSTM layer dropout = 0.2 and recurrent_dropout = 0.2, Dense layer activation = 'softmax', loss='categorical_crossentropy', and optimizer = 'adam', and validation_split = 0.1.* LSTM model is able to achieve 72.40% accuracy on test set. Figure 6 shows the LSTM model's summary.

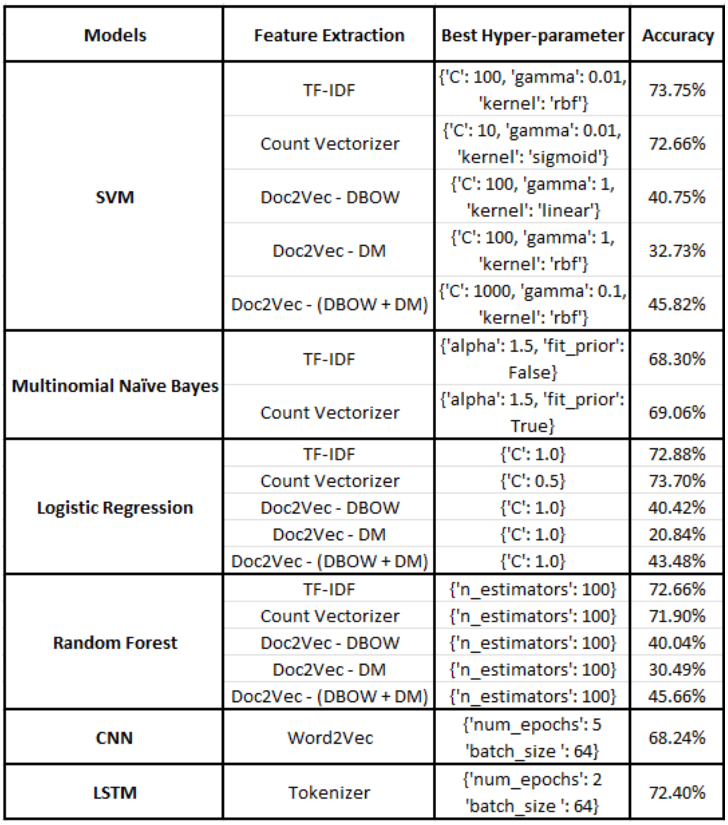


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 250, 100)          5000000

 spatial_dropout1d (SpatialD  (None, 250, 100)         0
 ropout1D)

 lstm (LSTM)                 (None, 250)               351000

 dense (Dense)               (None, 7)                 1757

=================================================================
Total params: 5,352,757
Trainable params: 5,352,757
Non-trainable params: 0
```

Figure 6: LSTM model's summary

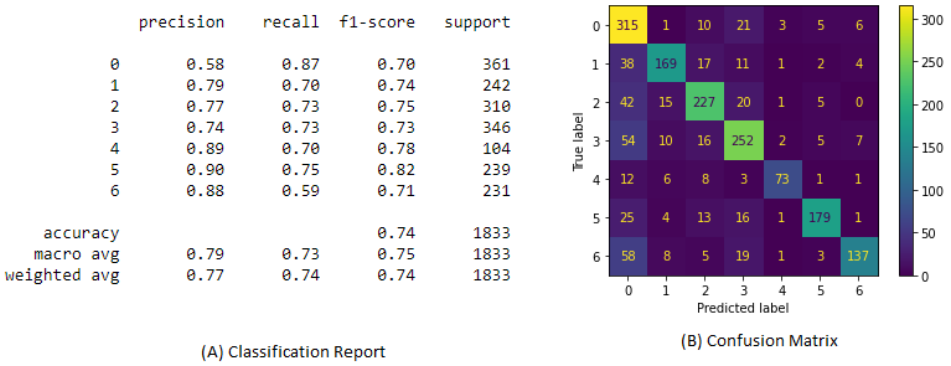
7 Evaluation

After training and finding accuracy for all the machine and deep learning models, it is observed from Figure 7 that the SVM model performed best for Code Mixed Hindi–English tweets, which gave an accuracy of 73.75% while using TF-IDF as vectorizer for feature extraction. Figure 8 shows the classification report and confusion matrix of the SVM model, which outperformed all the models.



Models	Feature Extraction	Best Hyper-parameter	Accuracy
SVM	TF-IDF	{'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}	73.75%
	Count Vectorizer	{'C': 10, 'gamma': 0.01, 'kernel': 'sigmoid'}	72.66%
	Doc2Vec - DBOW	{'C': 100, 'gamma': 1, 'kernel': 'linear'}	40.75%
	Doc2Vec - DM	{'C': 100, 'gamma': 1, 'kernel': 'rbf'}	32.73%
	Doc2Vec - (DBOW + DM)	{'C': 1000, 'gamma': 0.1, 'kernel': 'rbf'}	45.82%
Multinomial Naïve Bayes	TF-IDF	{'alpha': 1.5, 'fit_prior': False}	68.30%
	Count Vectorizer	{'alpha': 1.5, 'fit_prior': True}	69.06%
Logistic Regression	TF-IDF	{'C': 1.0}	72.88%
	Count Vectorizer	{'C': 0.5}	73.70%
	Doc2Vec - DBOW	{'C': 1.0}	40.42%
	Doc2Vec - DM	{'C': 1.0}	20.84%
	Doc2Vec - (DBOW + DM)	{'C': 1.0}	43.48%
Random Forest	TF-IDF	{'n_estimators': 100}	72.66%
	Count Vectorizer	{'n_estimators': 100}	71.90%
	Doc2Vec - DBOW	{'n_estimators': 100}	40.04%
	Doc2Vec - DM	{'n_estimators': 100}	30.49%
	Doc2Vec - (DBOW + DM)	{'n_estimators': 100}	45.66%
CNN	Word2Vec	{'num_epochs': 5 'batch_size ': 64}	68.24%
LSTM	Tokenizer	{'num_epochs': 2 'batch_size ': 64}	72.40%

Figure 7: Results of all the models.



```
              precision    recall  f1-score   support

           0       0.58      0.87      0.70       361
           1       0.79      0.70      0.74       242
           2       0.77      0.73      0.75       310
           3       0.74      0.73      0.73       346
           4       0.89      0.70      0.78       104
           5       0.90      0.75      0.82       239
           6       0.88      0.59      0.71       231

    accuracy                           0.74      1833
   macro avg       0.79      0.73      0.75      1833
weighted avg       0.77      0.74      0.74      1833
```

(A) Classification Report

True label \ Predicted label	0	1	2	3	4	5	6
0	315	1	10	21	3	5	6
1	38	169	17	11	1	2	4
2	42	15	227	20	1	5	0
3	54	10	16	252	2	5	7
4	12	6	8	3	73	1	1
5	25	4	13	16	1	179	1
6	58	8	5	19	1	3	137

(B) Confusion Matrix

Figure 8: Results of the SVM model.

8 Conclusion and Future Work

The main purpose of this research project is to find the best-supervised machine and deep learning models, which can identify emotions from Code Mixed Hindi–English tweets. This research is using 6 machine and deep learning algorithms such as 'SVM', 'Multinomial Naive Bayes', 'Logistic Regression', 'Random Forest', 'CNN', and 'LSTM'. The dataset was extracted using Twitter's official API - *'Tweepy'*. After data cleaning, the data was manually annotated into 7 classes of emotions, i.e., *'Happy', 'Sad', 'Angry', 'Fear', 'Disgust', 'Surprise', or 'No emotions'*. Next, the dataset was pre-processed by removing all noises. This research project has used 5 feature extraction techniques for creating numerical vectors from the data. It can be observed from Figure 7 that 'TF-IDF - unigrams and bigrams' is the best Vectorizer for Code Mixed Hindi–English tweets, however 'Count Vectorizer's - unigrams' performance is also very impressive and is very close to 'TF-IDF Vectorizer - unigrams and bigrams'. It can be observed that 'Doc2Vec' word embedding is not a recommended word embedding technique for Code Mixed Hindi–English tweets, as the performance of models when trained with data extracted using 'Doc2Vec' is very low, i.e., between 20% to 45%. The supervised deep learning models such as CNN and LSTM were able to achieve an accuracy of 68.24% to 72.40% respectively. The supervised machine learning models such as SVM, Logistic Regression, and Random Forest performed very well with both 'TF-IDF Vectorizer' and 'Count Vectorizer'. However, the SVM model along with 'TF-IDF Vectorizer - unigrams and bigrams' outperformed all the models and was able to achieve 73.75% accuracy and hence it can be concluded that the SVM model with 'TF-IDF Vectorizer - unigrams and bigrams' is the best and recommended supervised machine learning model for identifying emotions from bilingual Code Mixed Hindi–English tweets.

Future work on this Code Mixed Hindi–English tweets is very necessary to achieve higher accuracy. Some pre-trained models such as BERT, ALBERT, ROBERT, etc can be used with deep learning models such as CNN, LSTM, and Bi-LSTM. Also, feature extraction technique can be done by creating a word embedding manually, rather than using pre-trained word embeddings like Word2Vec, Doc2Vec, fastText, etc, because these word embeddings are pre-trained using English, German, French, Spanish, etc languages data, which is the result of low performance when used with bilingual Code Mixed Hindi–English tweets. [1]

[1] https://github.com/SanketSonu/NLP-Identifying-Emotions-for-Code-Mixed-Hindi-English-Tweets

9 Acknowledgement

I'm happy to say that I have completed this research project. I'd like to thank my supervisor Dr. Rejwanul Haque, for his guidance and assistant throughout the project work. He helped me and gave me hidden insights into this domain. His important observations and comments helped me to complete my project milestone on time. I'd also like to thank my friends and family to support me throughout the project execution period.

References

Ahmad, G. I., Singla, J. and Nikita, N. (2019). Review on sentiment analysis of indian languages with a special focus on code mixed indian languages, *2019 International Conference on Automation, Computational and Technology Management (ICACTM)*, pp. 352–356.

Choudhary, N., Singh, R., Bindlish, I. and Shrivastava, M. (2018). Sentiment analysis of code-mixed languages leveraging resource rich languages, *CoRR* **abs/1804.00806**.
URL: *http://arxiv.org/abs/1804.00806*

K., S., Ravikurnar, A., R C., V., Reddy D., A., M., A. K. and K.P., S. (2018). Sentiment analysis of indian languages using convolutional neural networks, *2018 International Conference on Computer Communication and Informatics (ICCCI)*, pp. 1–4.

Kastrati, Z., Ahmedi, L., Kurti, A., Kadriu, F., Murtezaj, D. and Gashi, F. (2021). A deep learning sentiment analyser for social media comments in low-resource languages, *Electronics* **10**(10).
URL: *https://www.mdpi.com/2079-9292/10/10/1133*

Mandal, S., Mahata, S. K. and Das, D. (2018). Preparing bengali-english code-mixed corpus for sentiment analysis of indian languages, *CoRR* **abs/1803.04000**.
URL: *http://arxiv.org/abs/1803.04000*

Mishra, P., Danda, P. and Dhakras, P. (2018). Code-mixed sentiment analysis using machine learning and neural network approaches.

Mozetič, I., Grčar, M. and Smailović, J. (2016). Multilingual twitter sentiment classification: The role of human annotators, *PLOS ONE* **11**(5): e0155036.
URL: *http://dx.doi.org/10.1371/journal.pone.0155036*

Mukherjee, S. (2019). Deep learning technique for sentiment analysis of hindi-english code-mixed text using late fusion of character and word features, *2019 IEEE 16th India Council International Conference (INDICON)*, pp. 1–4.

Rana, S. (accessed on 1st Oct 2021). Hinglishnlp, `https://github.com/TrigonaMinima/HinglishNLP/blob/master/data/assets/stop_hinglish`.

Salam, S. and Gupta, R. (2018). Emotion detection and recognition from text using machine learning, *International Journal of Computer Sciences and Engineering* **6**: 341–345.

Sasidhar, T. T., B, P. and P, S. K. (2020). Emotion detection in hinglish(hindi+english) code-mixed social media text, *Procedia Computer Science* **171**: 1346–1352. Third International Conference on Computing and Network Communications (CoCoNet'19).
URL: *https://www.sciencedirect.com/science/article/pii/S1877050920311236*

Singh, D. (2021). Detection of emotions in hindi-english code mixed text data, *CoRR* **abs/2105.09226**.
URL: *https://arxiv.org/abs/2105.09226*

Singh, M., Goyal, V. and Raj, S. (2019). Sentiment analysis of english-punjabi code mixed social media content for agriculture domain, *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*, pp. 352–357.

Srividya, K. and Sowjanya, A. M. (2019). Sentiment analysis of face book statuses, *International Journal of Recent Technology and Engineering (IJRTE)* **7**: 2277–3878.

Sulthana, A. R., Jaithunbi, A. K. and Ramesh, L. S. (2018). Sentiment analysis in twitter data using data analytic techniques for predictive modelling, *Journal of Physics: Conference Series* **1000**: 012130.
URL: *https://doi.org/10.1088/1742-6596/1000/1/012130*

Tho, C., Warnars, H. L. H. S., Soewito, B. and Gaol, F. L. (2020). Code-mixed sentiment analysis using machine learning approach – a systematic literature review, *2020 4th International Conference on Informatics and Computational Sciences (ICICoS)*, pp. 1–6.

Tiwari, S. and Sinha, A. (2020). Sentiment analysis of facebook data using machine learning, *International Journal of Innovative Research in Applied Sciences and Engineering* **4**: 2456–8910.

Vijay, D., Bohra, A., Singh, V., Akhtar, S. S. and Shrivastava, M. (2018). Corpus creation and emotion prediction for hindi-english code-mixed social media text, pp. 128–135.

Wadhawan, A. and Aggarwal, A. (2021). Towards emotion recognition in hindi-english code-mixed data: A transformer based approach, *CoRR* **abs/2102.09943**.
URL: *https://arxiv.org/abs/2102.09943*

Wehrmann, J., Becker, W., Cagnini, H. E. L. and Barros, R. C. (2017). A character-based convolutional neural network for language-agnostic twitter sentiment analysis, *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2384–2391.

Yadav, K., Lamba, A., Gupta, D., Gupta, A., Karmakar, P. and Saini, S. (2020). Bi-lstm and ensemble based bilingual sentiment analysis for a code-mixed hindi-english social media text, *2020 IEEE 17th India Council International Conference (INDICON)*, pp. 1–6.

Younas, A., Nasim, R., Ali, S., Wang, G. and Qi, F. (2020). Sentiment analysis of code-mixed roman urdu-english social media text using deep learning approaches, *2020 IEEE 23rd International Conference on Computational Science and Engineering (CSE)*, pp. 66–71.