Chapter 7

Sentiment Analysis on Hindi Tweets during COVID-19 Pandemic

7.1 Introduction

COVID-19, or Corona Virus Disease of 2019, was declared by the World Health Organization (WHO) on 11th march 2020 as a pandemic. People around the world faced or are still facing a national lockdown and maintaining social distancing. Today, the Internet has become a necessary part of civilized daily life. People often use blogs, forums, e-news, and social networking sites like Facebook, Twitter for getting the information and expressing their views and opinions on different issues. The web is more increasingly becoming a decision-making source. A large amount of content is generated on the web every day containing a lot of opinionated data. Often mining such data and automatically extracting user sentiment become important for various reasons like profitability for a business enterprise, gauging public sentiment, or even perceiving security issues. Micro-blogging sites like Twitter are famous for sharing opinions in short on a wide variety of topics by millions of users. Twitter users use a hashtag (starting with #), which groups the tweets on a similar topic containing that particular hashtag. When a massively large number of

people use a specific hashtag, it becomes a trend that attracts more people to participate in the discussion. Rapid growth in the development of access technologies and devices helped people to connect each other across barriers and thus growth in the number of social media users. During the COVID-19 pandemic, governments declared nation-wide lockdown across the world severely restricting physical interactions. Social media became the most popular communication channel, with the volume of tweets increased like never before. Sentiment analysis of such texts can come in handy to monitor and gain an overview of the pandemic's broader public opinion. Although several collections of such social media data have been created in English and other European languages, there is a dearth of such data in Hindi. Hindi is one of India's most widely spoken languages and a native language for around 300 million Indians. In this paper, we create a collection of Hindi tweets for sentiment analysis during a crisis and provide a benchmark for such short informal texts.

The Internet is serving as a universal and the most cost-effective source of information complemented by the growth of social media. Blogs, reviews, tweets, posts, discussions on social media, other than catering the news and situational information of the world, also reflect emotions of people. The attitude, views, feelings, emotions constitute an essential part of analyzing the behavior of a person, which can be referred to as the sentiments. Sentiment analysis, also known as opinion mining, deals with inspecting these sentiments directed towards any entity. Twitter is a powerful medium of people-to-people communication for sharing their feelings and thoughts on any topic or article, resulting in an enormous amount of unstructured information. Lack of social interaction in society has led people to spend much of their time in social media platforms during COVID-19. Microblog sites such as Twitter let the pandemic-struck world express their daily thoughts in real-time. Millions of tweets are put up each day. Although short and informal, these tweets capture all types of emotions, and therefore, can help us analyze society's finely grained emotions.

Sentiment analysis of these texts can be valuable and useful to study many physiological as well as psychological patterns in public during a pandemic. However, on the other hand, these tweets are short informal texts, and contain a lot of noise and are rich in language ambiguities.

Sentiment analysis is a type of text mining which classifies the text into different classes. Sentiment analysis uses some techniques to categorize sentiments into classes like positive, negative, and neutral. This type of division of classes is called the polarity of text. Most of the previous works focused on English or similar resourceful monolingual texts. Recent developments have also seen researchers working on code mixed data-sets. The prevalence of code mixed data-sets in a multilingual society like India is pretty standard. The dataset that we developed is primarily followed Hindi but contains English Hashtags as well.

The main purpose behind creating such a dataset in Hindi is to strengthen text processing research in low-resource languages. We have put in our best efforts to construct an annotated sentiment corpus and firmly believe that the corpus will prove to be of great help for the researchers in various natural language processing tasks on social media. Moreover, the tweets are limited to only 280 characters, and most of the published tweets do not even hit the prescribed character limit. The unstructured social media posts often consist of abbreviations, misspelled words, slang, and non-standard punctuation, making it challenging to perform any task in a low-resource language like Hindi. The previous works which have inspired this paper have analyzed sentiment analysis in a resourceful language with only monolingual utterances. The goal behind creating and working on such a dataset for Hindi is to ensure that enough dataset in such a low resource Hindi language is available for research purposes.

Post	Label
1: हिन्दू हृदय सम्राट मा.मुख्यमंत्री (उ.प्र) 23 करोड़ प्रदेशवासी आपके साथ है निश्चित ही हम सब कोरोना को मिल के हराएंगे Translation - Hindu Heart Emperor Mr. Chief Minister (U.P.) 23 crore people of the state are with you, surely we will all defeat Corona together.	Positive
2: भाजपा नेताओं ने तबलीग़ी जमात से देशभर में फैले संक्रमण को 'कोरोना जिहाद- मानव बम' कहा Translation - BJP leaders called the infection spread across the country from Tabligi Jamaat 'Corona Jihad - Human Bomb'.	Negative
3: चिकित्सा एवं स्वास्थ्य मंत्री ने कहा कि कोरोना पॉजीटिव मरीजों की सेवा में लगे चिकित्सक और नर्सिंग स्टाफ को संक्रमण से मुक्त रखने के प्रति सरकार गंभीर है। Translation - The Medical and Health Minister said that the government is serious about keeping the doctors and nursing staff engaged in the service of corona positive patients free from infection.	Neutral

FIGURE 7.1 SAFH data annotation

7.2 Contributions

As there is no prior work on the Hindi pandemics dataset, we collected Hindi posts from Twitter during the worldwide lockdown in April and May 2020. For Twitter posts, we use REST API in our program to download tweets. REST API takes words as queries, and multiple queries combined as a comma-separated list. Tweets from the previous ten days can be searched using this API. We downloaded 1,10,239 tweets from Twitter in CSV format, which consists of information such as timestamp, URL, text, user, re-tweets, replies, full name, and id. Noisy tweets are the ones that consist only of hashtags or URLs. Also, some tweets written in other languages like Gujarati, Tamil, Marathi, Bengali, etc. are also considered as noisy and hence removed from the corpus. Furthermore, all those tweets which written in code mixed like Hindi +English or Hindi +any Indian language were transliterated, and thus, keeping only the pure Hindi tweets. As a result, a dataset of 10,011 Hindi tweets was created and labelled by annotators as detailed below. Figure7.1 shows some examples of sentiment analysis in Hindi (SAFH) dataset.

T1 displays a positive sense of emotion by displaying the virtue of unity, whereas T2 displays a negative sentiment as it is showing hatred against a particular community. However, the third tweet T3, is neutral. The major challenge involved in annotation was

tackling satirical tweets. The satirical tweets have a gap between the intended and literal meaning. It is a widely studied linguistic phenomenon and can be a difficult task to annotate such tweets due to its ambiguous interpretation. The annotators usually had a good debate before annotating such tweets.

7.2.1 Annotation

The corpus had a lot of duplicate texts. Duplication was a significant issue faced by us due to the retweet feature of Twitter. We began annotating the tweets after erasing the duplicate tweets from our corpus. Many tweets were in Marathi but written in the Devanagari script. All such tweets were removed from the dataset. We annotated the tweets based on three standard emotions, negative, neutral, and positive.

Annotation of the dataset to detect sentiment was carried out by two human annotators having linguistic background and proficiency in Hindi and English. A sample annotation set consisting of 60 tweets (20 positive, 20 negative, and 20 neutral) selected randomly from all across the corpus was first provided to both the annotators to have a reference baseline to differentiate between positive, negative, and neutral text. Later, to validate the quality of annotation, we calculated the inter-annotator agreement (IAA) for sentiment annotation between the two annotation sets of 10,011 Hindi tweets using Cohen's Kappa coefficient. Kappa score is 0.882, which indicates that the quality of the annotation is quite good and acceptable.

7.2.2 Data Statistics

The total number of tweets that were retrieved from Twitter was 1,10,239. The tweets were then filtered, as described above, and pre-processed to create our corpus of 10,011 tweets. Table ?? shows the distribution of data across different emotions. It clearly shows that the distribution of negative tweets exceeds the count of positive and neutral text.

Table 7.1 Distribution of labels combinations in SAFH data.

Label	#posts
Positive	3080
Negative	3726
Neutral	3205

This explains that people are distressed because of the pandemic and are letting out their emotions on social media sites. It is observed that some of the hashtags containing Hindi keywords are written in Roman script. This led to the creation of a transliterated dataset. Transliteration is a type of conversion of a text from one script to another that involves swapping letters. Both the datasets have been split into the test and train dataset in a 70-30 ratio respectively. It is observed that some of the tweets contained multiple phrases depicting different emotions, and some of the tweets included humors as well.

7.3 Methods

The classification of the sentiment of the tweets is based on generating word-embedding and supervised learning. Word embedding are an effective way to represent words - i.e., words with the same meanings are described in the same way in the vectors. As the quality of word embeddings depends on the quality of input data, representing the data in a vector form is essential. Nowadays, embedding of words into low dimensional space is mostly suggested. Word2Vec is a statistical method for learning word embedding from a large text corpus. It outputs a high-dimensional vector space, where each word from the corpus is assigned a vector, and words with familiar contexts are placed proximally close in the vector space. We have chosen Fasttext, a pre-trained word embedding developed and open-sourced by Facebook. It provides word embedding for Hindi (and 157 other languages) and is based on the CBOW (Continuous Bag-of-Words) model. The CBOW model learns by predicting the current word based on its context, and it was trained on Common Crawl

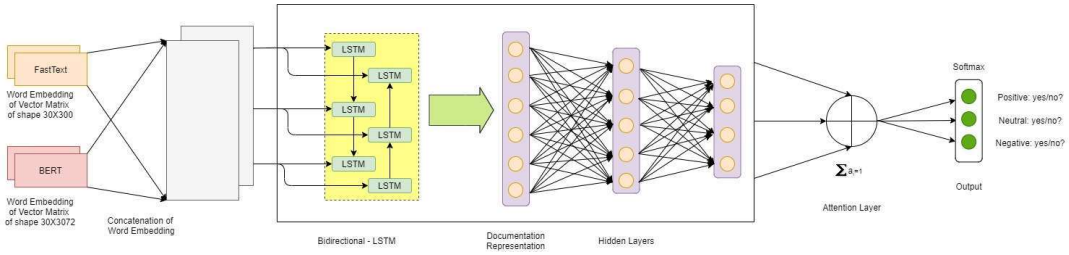

FIGURE 7.2 Architecture of the proposed work

and Wikipedia. Fasttext has given remarkable results in the English language. Still, in a regional language like Hindi, it is found that due to the unavailability of a large corpus of data, the experiments are done with a regular Deep learning algorithm with a traditional approach. Here, we use and discuss a novel architecture to analyze the performance w.r.t. BiLSTM neural network. The architecture of the proposed word has been mentioned in Figure 7.2. We used different embeddings in our work.

7.3.1 BERT

BERT (Bidirectional Encoder Representations from Transformers) proposed by Delvin et al. [211], which is a contextualized word representation model designed to train deep bidirectional representations by masking language model from unlabeled texts in all the layers by the combination of two unidirectional language model (from right to left and left to right). BERT utilizes a masked language model to predict random words masked in a sequence and subsequently learn bidirectional representations. The masked language model's primary purpose is to mask a word randomly in a sentence with some probability. The masked word is replaced with a token[MASK] by the model. The model then tries to predict the word masked by the model using the context from both the left and right of the masked word with the transformer's help. In the experiment, we have used BERT base multilingual model with 12 layers (i.e., transformer blocks) and 12 self-attention heads with 768 hidden feed-forward network units. Suppose a sentence sequence S is masked

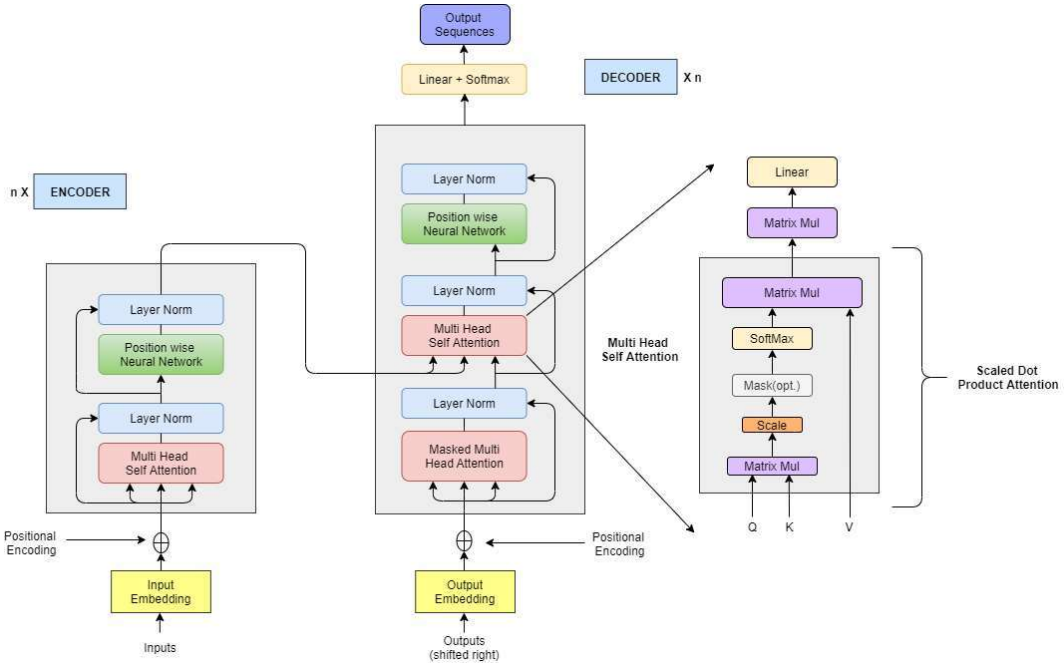

FIGURE 7.3 Architecture of the BERT

by the word W(word), and the transformer extracts the corresponding representation X (word); for masking word prediction task, the loss function is defined as shown here.

$$loss_i = \sum L(x_M^i) F(x_M^i) \tag{7.1}$$

Where $L(x)$ is the loss function of the i-th masked word X in the masked prediction task. $F(x)$ is the weight of the word X. The dimension used is 3072 for each word representation for the size of the vocabulary. Since all the tweets are not of equal length, we have used padding to make them equal. We have padded each tweet to its average length. We have used post padding to make sentence length 30 for every tweet. The tweets having a more considerable length than the average size have been truncated to the average and the ones having smaller length have been appended with zeros to make the length equal to the average length. Figure 2 shows the working module of the BERT embedding.

7.3.2 FastText

FastText was created by Facebook [181], which is an extension of Word2Vec and used for learning word embedding. It treats every word as the smallest unit whose vector representation is found by n-grams of character. Finding the vector representation for rare words can be easily done. The n-gram concept can convert the words that are not present in the dictionary into vectors [182]. The dimension used is 300 for each word representation for the size of the vocabulary. Since all the tweets are not of equal length, we have used padding to make them equal. We have padded each tweet to its average length. We have used post padding to make sentence length 30 for every tweet. The tweets having a more considerable length than the average size have been truncated to the average and the ones having smaller length have been appended with zeros to make the length equal to the average length.

7.3.3 Self-attention

The attention mechanism was introduced in 2014 by Bahdanau et al. [215], and since then, it has been applied successfully to various NLP (e.g., machine translation) tasks. The attention mechanisms use the process of computing a context vector by performing a weighted average on the encoder of the hidden states that contains the most relevant information from all of the encoders for the next decoder step. The weighted average is determined by an alignment score between the encoder state and the previous hidden state of the decoder. Self-attention is a mechanism that can assign 'attention' to a key vector or an important word vector. It allows the architecture, in a sense, to emphasize attention vectors. We proposed a model incorporating the Attention Layer. The word vectors go through the Attention Layer, generating attention-weighted features [216].

Mathematically, the previous decoder state is considered the query vector Q, whereas the encoder hidden states are regarded as the key $K(k_1, k_2, k_3, ...)$ and value $V(v_1, v_2, v_3, ...)$

vectors. Note that the values and keys can be different sets of vectors. The weights determined with the help of the compatibility function between the query and the keys as in equation 7.2, and the scaling factor used is $\frac{1}{\sqrt{d_z}}$.

$$m_{ij} = \frac{QKT}{\sqrt{d_z}} \tag{7.2}$$

Next to the attention is produced α_{ij} in equation 7.3, is computed using compatibility function m_{ij} with the help of softmax function over the scaled inner product.

$$\alpha_{ij} = \frac{e^{m_{ij}}}{\sum_{k=1}^{n} e^{m_{ik}}} \tag{7.3}$$

Finally, the weighted sum of the attention and a value is calculated as equation 7.4.

$$Z_i = \sum_{j=1}^{n} \alpha_{ij} V \tag{7.4}$$

7.3.4 BiLSTM with BERT + Fasttext

The deep learning model takes input as the embedding layer, which encodes each token in the dataset used by the model. We have used different embedding and concatenated them to serve as the input to the model proposed. One of the embeddings used is Fasttext for generating embedding, and the dimension used is 300 for each word representation for the size of the vocabulary. The contextualized word vectors that are predicted by BERT embedding of 3072 dimensions are being concatenated with the Fasttext to generate a single input embedding layer. After the embedding layer, the sequence of word vectors is fed into Bidirectional LSTM, an extension of the traditional LSTM, to train two LSTMs on the input pairs. The second LSTM is a reversed copy of the first one so that we can take full advantage of both past and future input features for a specific time step and generate the documentation representation. The outputs of the BiLSTM model are incorporated with

self-attention layers followed by three hidden layers. The Self-Attention-based BiLSTM neural networks model is proposed for the sentiment polarity classification which has highly correlated with the aspect-terms and the opinion terms in the sentence. Therefore, how to concentrate on these aspect-terms is very important in the sentiment analysis task. However, the standard BiLSTM cannot recognize which is the more important part of sentiment analysis. To solve this problem, a simple and effective self-attention mechanism is introduced that can capture the important part of a sentence. Finally, the sentiment tendency of the comments is obtained. The activation function of BiLSTM is a softmax function. The dropout mechanism was introduced to prevent the over-fitting phenomenon in the training process, and the dropout discarding rate was set to 0.5.

7.3.5 BiLSTM+ fastText with self attention

FastText assumes a word to be formed by n-grams of character and helps to generate the embedding layer. The embedding layer acts as the input layer to the BiLSTM sequence processing model, which takes the input in both forward and backward directions. The neural network is attached to the self-attention layer. Self-attention allows the model to look at the other words in the input sequence to get a better understanding of a certain word in the sequence. Then the polarity of the classes has been determined, which is incorporated with the dropout.

7.3.6 BiLSTM with fastText

One of the word embedding used is the fastText, which is an extension to Word2Vec proposed by Facebook. FastText can generate embedding for the words that do not appear in the training corpus. This can be done by adding the character n-gram of all the n-gram representations. Using the fastText embedding we have generated the vector-matrix of 30x300 dimensions and been fed to the BiLSTM neural network with three hidden layers.

Then the softmax activation function is used to determine the probability of the different classes.

7.4 Experiments

The details of experiments are given below.

7.4.1 Pre-processing

The cleansing of duplicates in the dataset mentioned in Section 3 is carried out in the pre-processing of the text. All the links and URLs in the tweets are removed. The user names which are often mentioned in the tweets are removed. All the punctuation marks, hashtags, and spaces in a tweet are removed. All the numbers in the tweets are drawn. Everything other than text can add up to noise in the text and hence is removed.

The extraction of keywords in Twitter is harrowing because of noise. So, to avoid this, a preprocessing step is performed before feature extraction. Stopword removal is an essential type of preprocessing technique in text processing because it can reduce the length of a document without affecting its sentiments. Preprocessing includes various steps such as remove website URLs, remove hashtags, Twitter mentions(), stopwords, and special characters and punctuations. The most challenging task was to differentiate between these two symbols, '।' and '।.' The symbol in the first is the delimiter in the Hindi Language. Still, in some documents, the second symbol has been used, and it consumed a lot of time to identify this issue. All the numbers and the unnecessary spaces in the tweets are removed.

7.4.2 Benchmark Systems

After creating the annotated corpus, sentiment detection in the Hindi dataset is performed first. We provide a simple baseline for determining the emotions in the dataset by applying various classification algorithms. We break down our sentimental analysis into two sub-processes, pre-processing of the raw text and the classification of the tweets based upon their emotions.

Support Vector Machine

We evaluate our model with C-Support Vector Classification with L2 regularization. SVM takes care of multi-class classification problems by treating it as a one-vs-one scheme. SVM Algorithms have many advantages in terms of complexity. It is a robust algorithm and has performed the best in our problem statement. It is memory-efficient because of the Kernel trick and defines the optimal hyperplane in N-dimensional space. This makes it optimal to use it in case of a higher number of dimensions.

Random Forest

The corpus that we have worked on has variations because of the character limit by Twitter. This has resulted in the decision tree being unstable with the complete tree not being generated or can also cause overfitting. This variation can be alleviated by using the decision tree within an ensemble, Random Forest. Random forest uses several decision tree classifiers to improve the metrics we have used in our algorithms. We have used 100 trees in our forest and have set our max depth as 150. The criterion was Ginni impurity.

Logistic Regression

The logistic regression model's input features are term frequency-inverse document frequency (TF-IDF) values up to 3 grams. L2 Regularization is applied with the Logistic

regression model with the help of the Limited-memory BFGS optimization algorithm. The optimization algorithm uses less memory and is a conventional algorithm used in parameter estimation. The training algorithm makes use of the cross-entropy loss function.

Multinomial Naive Bayes

Naive-Bayes classifier is a probabilistic model derived from the Bayes theorem that finds the probability of hypothesis activity to the given evidence activity. We evaluate the MNB model with our data using α=1 with TF-IDF vectors.

XG Boost

XG Boost is a gradient boosting classifier with a decision tree-based ensemble approach. It is a robust algorithm that provides a parallel tree boosting. The ensemble model endeavors to create a reliable model based on all the weaker models. We have evaluated this model using the extracted TF-IDF feature vectors.

k-Nearest Neighbour $(k-NN)$

k-nearest neighbour is a non-parametric learning algorithm. There are not any assumptions made about the data used in the model. This feature of $k-NN$ makes it a handy feature since datasets do not follow any theoretical assumption, e.g., such as linear-separability, uniform distribution, etc. We made use of 3 nearest neighbors for our case. All points in the neighborhood have uniform weight, and the most appropriate algorithm based on the dataset is passed onto the model using the fit method.

The above algorithms are trained by making use of feature vectors. The feature vectors are either extracted by applying the term frequency-inverse document frequency in the corpus. It is a feature vector intended to reflect how important a word is to a document or a corpus. We have also made use of the word embeddings to extract features from text.

7.5 Results

The sentiment classification results using different models has been shown using precision in Table 7.2, recall in Table 7.3, Table 7.4 presents F_1-score, and Accuracy in Table 7.5. Our corpus contains a dataset with a larger number of negative sentiment tweets compared to positive and neutral. Table 7.1 shows that out of 10,011 tweets, 37.21% of the tweets belong to the Negative class. The recall, and F_1-scores are highest for the negative class, even though precision is slightly better or comparable to negatives. Although all the models have given almost similar results, SVM has shown the best among all the machine learning algorithms. It has a precision of 0.61, a recall of 0.61, an F_1-Score of 0.61, and an accuracy of 0.61. The SVC classifier carries out the multi-class classification through SVM, which treats the problem in a One-vs-One fashion. The algorithm is highly effective in high dimensional spaces, which is demonstrated throughout this work.

Table 7.2 Precision

Classifier	Positive	Negative	Neutral	Macro Avg	Weighted Avg
Support Vector Machine	0.68	0.66	0.50	0.61	0.61
Random Forest	0.64	0.66	0.51	0.60	0.60
Logistic Regression	0.64	0.62	0.49	0.58	0.59
Multinomial Naive Bayes	0.62	0.59	0.50	0.57	0.57
XG Boost	0.63	0.52	0.46	0.54	0.54
k-NN	0.70	0.44	0.50	0.55	0.54

Table 7.3 Recall

Classifier	Positive	Negative	Neutral	Macro Avg	Weighted Avg
Support Vector Machine	0.62	0.73	0.48	0.61	0.61
Random Forest	0.63	0.69	0.49	0.60	0.61
Logistic Regression	0.64	0.71	0.49	0.59	0.59
Multinomial Naive Bayes	0.61	0.78	0.33	0.57	0.58
XG Boost	0.48	0.77	0.33	0.53	0.54
k-NN	0.26	0.84	0.29	0.46	0.48

Table 7.4 F_1-score

Classifier	Positive	Negative	Neutral	Macro Avg	Weighted Avg
Support Vector Machine	0.65	0.69	0.49	0.61	0.61
Random Forest	0.64	0.67	0.50	0.60	0.60
Logistic Regression	0.63	0.66	0.46	0.58	0.69
Multinomial Naive Bayes	0.62	0.67	0.40	0.56	0.57
XG Boost	0.55	0.62	0.39	0.52	0.52
k-NN	0.38	0.58	0.37	0.44	0.45

Table 7.5 Accuracy

Classifier	Accuracy
Support Vector Machine	0.61
Random Forest	0.61
Logistic Regression	0.59
Multinomial Naive Bayes	0.58
XG Boost	0.54
k-NN	0.48

Deep learning-based algorithms are tested also on the SAFH dataset. The performance of each model tried is measured in terms of the F_1-scores. Experimental results show a macro average F_1-score using BERT embedding generation, which is significantly less in the Hindi dataset. The main reason for this observation was that models like BERT predict vector values of individual tokens, not sentences. This causes sentence aggregations to mis-align the vectors due to lexical differences in languages.

Further, the experiment was also performed using the extension of Word2Vec, i.e., FastText word embedding, and experimental results show that macro avg $F_1 - score$ increased. The proposed algorithm performs well for positive, negative, and neutral comments. In our further versions of the experiments, we analyzed the impact of the concatenation of both the embedding generated, i.e., BERT and FastText. Experimental results show an improvement in overall sentiment analysis performance, especially for positive reviews. After integrating the attention layer with the BiLSTM model on the FastText embedding, the $F_1 - score$ improved for both positive and negative sentiment.

Table 7.6 BiLSTM + FastText with self attention on SAFH COVID-19 dataset

Classifier	Precision	Recall	F_1 score	Support
Positive	0.73	0.71	0.72	945
Neutral	0.73	0.76	0.75	1089
Negative	0.60	0.45	0.51	970
Macro Avg	0.69	0.64	0.66	3004
Weighted Avg	0.69	0.64	0.66	3004

Table 7.6 shows the result on BiLSTM + FastText with self attention for SAFH dataset. We obtained the accuracy on BiLSTM + FastText with self attention is 54.5 %, the lowest.

Table 7.7 BiLSTM with FastText on SAFH COVID-19 dataset

Classifier	Precision	Recall	F_1 score	Support
Positive	0.71	0.72	0.71	945
Negative	0.74	0.75	0.74	1089
Neutral	0.58	0.44	0.50	970
Macro Avg	0.68	0.63	0.65	3004
Weighted Avg	0.68	0.64	0.66	3004

Table 7.7 shows the result of BiLSTM with FastText for SAFH COVID-19 dataset.

Table 7.8 BiLSTM with BERT + FastText on SAFH dataset

Classifier	Precision	Recall	F_1 score	Support
Positive	0.67	0.62	0.65	945
Negative	0.70	0.71	0.71	1089
Neutral	0.50	0.50	0.50	970
Macro Avg	0.63	0.61	0.62	3004
Weighted Avg	0.63	0.62	0.62	3004

Table 7.8 shows the result of BiLSTM with BERT + FastText for Hindi language.

Table 7.9 shows the accuracy of Hindi language on all proposed model.

7.6 Summary

We created a Hindi dataset for sentiment analysis on social media during crisis. During CoVID-19 pandemic, Hindi data of one month's time over Twitter show dominance of a

Table 7.9 Accuracy of the proposed models on SAFH dataset

Model	Hindi
BiLSTM with BERT + Fasttext	80.21%
BiLSTM + Fasttext with self attention	54.5%
BiLSTM with Fasttext	78.26%
BiLSTM with BERT	54.5%

sense of negativity in the SM. Even though good number of positive tweets were there with equal number of neutral ones. People held comparatively more negative to neutral views during lockdown possibly because of anxiety and uncertainties looming large that time. The second outcome derived from this work is the performance of different ML and DL classifiers. While SVM found to be the best among ML classifiers for Hindi sentiment detection on our data, BiLSTM with BERT and FastText was the overall best performer among all (considering ML as well as DL).