Sentiment Analysis of Pandemic Tweets with COVID-19 as a Prototype

Mashail Almutiri, Mona Alghamdi, Hanan Elazhary
Information Systems and Technology Department, University of Jeddah, Jeddah, Saudi Arabia

Abstract—**One of the most important applications of text mining is sentiment analysis of pandemic tweets. For example, it can make governments able to predict the onset of pandemics and to put in place safe policies based on people's feelings. Many research studies addressed this issue using various datasets and models. Nevertheless, this is still an open area of research in which many datasets and models are yet to be explored. This paper is interested in the sentiment analysis of COVID-19 tweets as a prototype. Our literature review revealed that as the dataset size increases, the accuracy generally tends to decrease. This suggests that using a small dataset might provide misleading results that cannot be generalized. Hence, it is better to consider large datasets and try to improve analysis performance on it. Accordingly, in this paper we consider a huge dataset, namely COVIDSenti, which is composed of three sub datasets (COVIDSenti_A, COVIDSenti_B, and COVIDSenti_C). These datasets have been processed with a number of Machine Learning (ML) models, Deep Learning (DL) models, and transformers. In this paper, we examine other ML and DL models aiming to find superior solutions. Specifically, we consider Ridge Classifier (RC), Multinomial Naïve Bayes (MNB), Stochastic Gradient Descent (SGD), Support Vector Classification (SVC), Extreme Gradient Boosting (XGBoost), and the DL Gated Recurrent Unit (GRU). Experimental results have shown that unlike the models that we tested, and the state-of-the-art models on the same dataset, SGD technique with count vectorizer showed quite constantly high performance on all the four datasets.**

Keywords—COVID-19; deep learning; machine learning; sentiment analysis; text mining; tweets

I. Introduction

Text Mining (TM) deals with the automatic extraction of interesting information from text. It uses data mining techniques to extract information that is hidden within huge amounts of unstructured textual data. Such extracted information is typically transformed into a structured format that can be further processed, possibly using Natural Language Processing (NLP) techniques. Sentiment Analysis (SA) is a text mining approach that utilizes data science techniques including Machine Learning (ML) and Deep Learning (DL) to analyze text and identify subjective information. It is concerned with assessing feelings and opinions, and classifying them into polarities, which are typically either positive, negative, or neutral. Social media are popular networks such as Facebook and X (formerly, Twitter) that are used by users to share their reviews about various topics and incidents. Additionally, one of the main uses of the Internet is checking reviews of others and expressing personal opinions. Since social media posts are mainly about expressing feelings and opinions, some researchers believe that Opinion Mining (OM) refers to social media SA. SA and OM are often used interchangeably. We adopt the term SA in this paper.

One of the most important and popular social media platforms is X in which users express their opinions using *tweets*. Since social media has a great influence on society, tweets about pandemics would most probably stimulate fear and agony. Tweets data mining can be very helpful in generating important health-related facts. For example, many research studies have shown that tweets can be exploited in the prediction of the onset of pandemics or diseases. SA of tweets is thus of special importance during pandemics. It is of great benefit to people's lives as it makes governments able to put in place safe policies based on inferred people's feelings.

A recent example of such pandemics is Coronavirus Disease 2019 (COVID-19). On March 11th, 2020, the World Health Organization (WHO) announced COVID-19 as a pandemic. Since at that time, no known effective vaccines or treatments existed, the governments and public health sectors had to take some precautionary decisions to avoid the spread of the infection, including isolation, quarantine, and emergency lockdown. During this period, COVID-19 had a negative effect on various aspects of people’s lives, and because the lockdown gave them more free time, the subject of the greatest portion of posts and tweets at that time was that pandemic. Many research studies have been concerned with SA of COVID-19 tweets using various datasets and various models (ML, DL, and transformers) [1-9]. Nevertheless, SA of pandemic tweets in general is still an open area of research in which many datasets and numerous models are yet to be explored.

This paper is concerned with the SA of tweets during pandemics and considers COVID-19 as a prototype. In other words, the research problem is how to analyze tweets during pandemics and decide whether their sentiment is positive, negative, or neutral. Literature review has revealed that accuracy generally tends to decrease with the increase of the dataset size. This suggests that results based on relatively small-sized datasets might be misleading. Hence, we decided to work with large datasets and try to find better solutions for analyzing them. Towards this goal, we manipulated a huge dataset, namely COVIDSenti and its three sub-datasets (COVIDSenti_A, COVIDSenti_B, and COVIDSenti_C) [8]. These datasets have been processed with a number of ML models, DL models, and transformers. In this paper, we examine other ML and DL models aiming to find superior solutions. The contributions of the paper can be summarized as follows:

- Searching for better performing ML and DL model(s) for SA of COVIDSenti tweets through a set of experiments.
- Comparison of the results of the various models with the state of the art to gain insights that can guide future research.

The rest of this paper is organized as follows: Section II presents related work on COVID-19 tweet analysis. Section III explains the proposed methodology that this article follows. Section IV discusses the datasets and the data preprocessing techniques. Models and experimental settings are presented in Section V. In Section VI, we compare their performance with the state of the art. Finally, Section VII depicts conclusions and draws directions for future work for further improvements.

II. Related Work

Several research studies studied sentiment analysis of COVID-19 tweets. We discuss a recent sample in the order of the dataset size. For example, Shofiya and Abidi [1] extracted 629 Canadian tweets from an open-source publicly available IEEE website. They used SentiStrength tool to detect sentiment polarity in combination with Support Vector Machine (SVM) classifier. The highest achieved accuracy of 87% was achieved on 10% test data. They concluded that a large dataset is required to increase the performance of the algorithm. On the other hand, Chintalapudi et al. [2] considered Indian tweets and obtained from github.com a dataset consisting of 3,090 tweets extracted from the Indian Twitter platform. The tweets were classified into “afraid,” “sad,” “angry,” and “happy.” They compared Bidirectional Encoder Representations from Transformers (BERT), Logistic Regression (LR), SVM, and Long Short-Term Memory (LSTM). The accuracy has been used as a metric to evaluate the models. The results showed that the BERT model outperformed the other models with 89% accuracy. Gupta et al. [3] also considered Indian tweets and processed 7,284 tweets having the keyword *India lockdown*. They compared MNB, Bernoulli Naïve Bayes (NB), LR, linear SVC, AdaBoost, Ridge classifier, passive aggressive (PA) classifier, and a perceptron using accuracy, precision, recall, and F1-score. In their experiments, linear SVC with unigram showed best performance with 84.4% accuracy, 83.5% precision, 82.4% recall, and 82.5% F1-score.

Ramya et al. [4] considered a slightly larger dataset consisting of 11,000 tweets (10,000 for training and 1000 tweets for testing) and used NB and LR to process them combined with *n-grams*. They also used accuracy as their metric. Interestingly, in their experiment, NB showed accuracy of about 92.49% for short tweets (less than 70 characters) and much lower accuracy of only 60.56% in the case of longer tweets.

Other researchers considered relatively larger datasets. For example, Goel and Sharma [5] collected a dataset comprised of 42,468 tweets. They analyzed them using SVC, Random Forest (RF), a neural network (NN) which is a combination of convolutional NN (CNN) and LSTM, and BERT. To evaluate the performance, they used Area Under the Receiver Operating Characteristic Curve (AUC). In their experiment, the best model was RF with 96% AUC accuracy. Vernikou et al. [6] used another dataset consisting of 44, 955 tweets. Their goal was to classify tweets into negative, neutral or positive. They used seven different DL models based on LSTM, and a set of ML models (MNB, Decision Tree (DT), and RF). The models were evaluated using accuracy, precision, recall, and F1-score. In their experiment, one of the LSTM-based models, namely BERT Tokenizer LSTM showed the best performance with 90% accuracy, precision, recall, and F1-score. Qi and Shabrina [7] extracted a total of 77,332 unique tweets and processed those using RF, MNB and SVC models. Performance was evaluated using precision, recall, F1-score, and accuracy. They also considered three different methods of feature representation, namely, bag of words (BoW), TF-IDF, and Word2Vec. In their experiments, the SVC using BoW or TF–IDF showed the best performance with accuracy of 71%.

Naseem et al. [8] prepared the largest dataset that we encountered in our literature review. This is the COVID Senti dataset which is composed of 90,000 unique tweets from 70,000 users. This dataset was divided into three subsets, namely, COVID Senti_A, COVID Senti_B, and COVID Senti_C. This dataset, that we adopt in this paper, is discussed together with its subsets in more details in Section IV. The four datasets were analyzed using SVM, NB, RF, and DT. In addition to those ML models, they utilized CNN and Bi-directional LSTM (Bi-LSTM). This is in addition to a set of hybrid models and transformers that we do not consider in the current research. Accuracy has been used as a metric to evaluate performance. Experimental results have shown that among the ML models considered, SVM and RF with FastText embedding showed accuracy of 84.5% on COVIDSenti. On the other hand, among the DL models considered, CNN with Glove word embedding showed higher performance of 86.9% on the same dataset.

Jalil et al. [9] used the same dataset in their experiments. They examined a set of different ML models, namely K-Nearest Neighbors (KNN), LR, ensemble, XGBoost, SVM, NB, DT, and RF. They also examined a set of deep learning models based on CNN and BiLSTM. This is in addition to a set of hybrid models and transformers, which as previously noted, is not considered in this research study. They also used accuracy for performance evaluation. Among the ML models, XGBoost showed the best performance with 89.81% accuracy on COVIDSenti. On the other hand, among the DL models, a combination of CNN, LSTM and Glove word embedding showed the best performance with 87.06% accuracy on the same dataset.

The related work is summarized in Table I. As shown in the table, researchers used different models and datasets for SA of COVID-19 tweets. We notice that as the dataset size increases, the accuracy generally tends to decrease. This suggests that using a small dataset might provide misleading results that cannot be generalized. Hence, it is better to consider large datasets and try to improve analysis performance on it. Accordingly, in this paper we consider the largest dataset that we encountered, namely, COVIDSenti and its three subsets. Also, we limit our research study to ML and DL models aiming to find better solutions that would be used in case of any future pandemic. Towards this goal, we consider additional models in comparison to those used on the same dataset [8-9].

TABLE I. SUMMARY OF RELATED WORK

Ref No.	Dataset	Models	Metrics	Best Results
[1]	629 tweets extracted from an open-source free IEEE website	SVM	confusion matrix, precision, recall, and F1-score	87% accuracy on 90% training data
[2]	3,090 tweets from the Indian Twitter platform	BERT, LR, SVM, LSTM	accuracy	BERT with 89% accuracy
[3]	7,284 tweets having the keyword *India lockdown*	MNB, Bernoulli NB, LR, Linear SVC, AdaBoost, Ridge classifier, passive aggressive (PA) classifier, perceptron	accuracy, precision, recall, F1-score	Linear SVC with unigram with 84.4% accuracy, 83.5% precision, 82.4% recall, and 82.5% F1-score
[4]	11,000 tweets (10,000 for training and 1000 tweets for testing)	NB and LR	accuracy	NB with 91% accuracy on short tweets
[5]	42,468 tweets	SVC, RF, NN, BERT	AUC	RF with 96% AUC
[6]	44, 955 tweets	Seven LSTM-based models, MNB, DT, RF	accuracy, precision, recall, F1-score	Bert Tokenizer LSTM with 90% accuracy, precision, recall, and F1-score
[7]	77,332 tweets	RF, MNB, SVC	precision, recall, F1-score, and accuracy	SVC using BoW or TF–IDF with 71% accuracy
[8]	COVIDSenti, composed of 90,000 tweets and its three subsets (COVIDSenti_A, COVIDSenti_B, and COVIDSenti_C)	ML (SVM, NB, DT, RF), DL (CNN, BiLSTM), and a set of hybrid models and transformers	accuracy	ML models: SVM and RF with FastText embedding with 84.5% accuracy on COVIDSenti DL models: CNN with Glove word embedding with 86.9% accuracy on COVIDSenti
[9]	COVIDSenti, composed of 90,000 tweets and its three subsets (COVIDSenti_A, COVIDSenti_B, and COVIDSenti_C)	ML (KNN, LR, Ensemble, XGBoost, SVM, NB, DT, RF), DL (CNN, BiLSTM), and a set of hybrid models and transformers	accuracy	ML models: XGBoost with 89.81% accuracy on COVIDSenti DL models: a combination of CNN, LSTM and Glove word embedding with 87.06% accuracy on COVIDSenti

III. METHODOLOGY

The methodology pipeline shown in Fig. 1 follows a typical data analytics lifecycle. As shown in the figure, COVID-19 tweets are first pre-processed, and features are extracted from them. Then two experiments are conducted. The first considers a set of ML models preceded by data vectorization (represented as numeric vectors). The second, on the other hand, involves the Gated Recurrent Unit (GRU) DL model. This involves a different data representation technique. This is followed by discussing and comparing the results of both experiments. Finally, we compare our results with those of the state-of-the-art surveyed research.

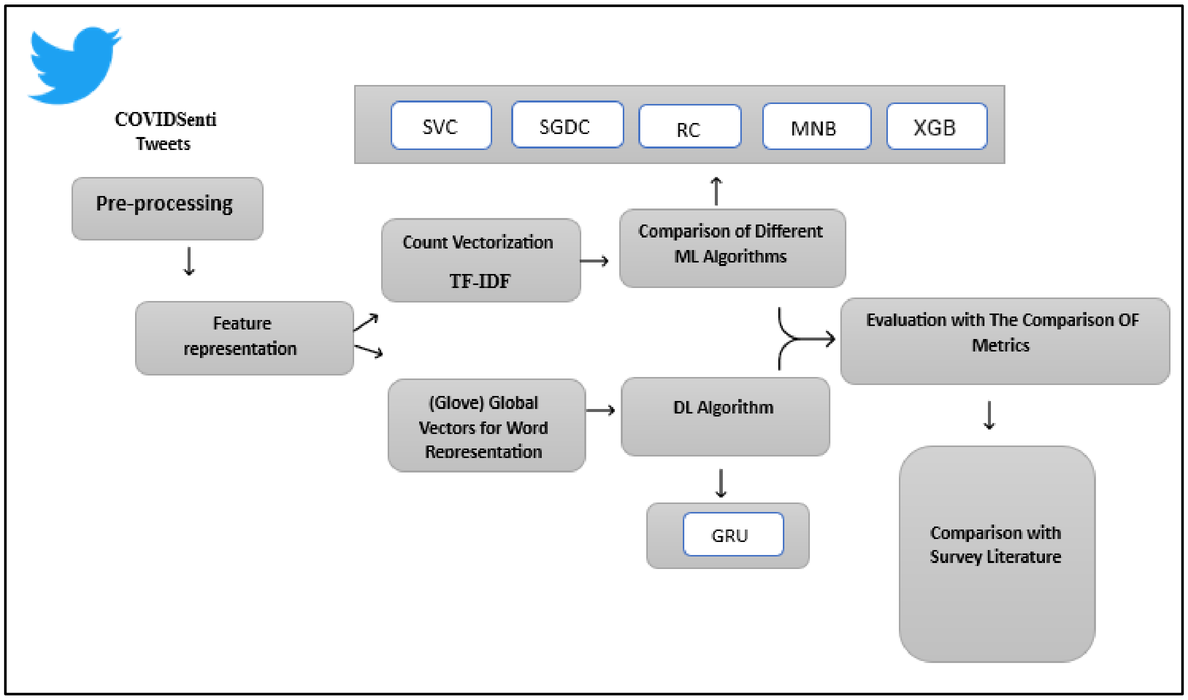
COVIDSenti
Tweets
Pre-processing
Feature representation
Count Vectorization
TF-IDF
Comparison of Different ML Algorithms
SVC
SGDC
RC
MNB
XGB
(Glove) Global Vectors for Word Representation
DL Algorithm
GRU
Evaluation with The Comparison OF Metrics
Comparison with Survey Literature

Fig. 1. Overview of the proposed approach.

IV. Datasets and Data Preprocessing

In this section, we discuss the details of the datasets utilized in the experiments and how we pre-processed them before building our models.

A. Datasets Details

As previously noted, in this research study, we consider the COVIDSenti dataset. This dataset is sourced from the open-source hosting site GitHub and consists of 90,000 unique tweets from 70,000 users about COVID-19 from February 2020 to March 2020. Naseem et al. [8] divided the COVIDSenti dataset into three subsets for evaluation and generalization purposes: COVIDSenti_A, COVIDSenti_B, and COVIDSenti_C. They treated them as four different datasets and we adopt the same approach. Each of the four datasets, for classification purposes, has three sentiments: positive, negative, and neutral. Table II provides an overview of these datasets. As shown in the table, overall, the neutral sentiments form the highest percentage of COVIDSenti and its subsets.

B. Data Preprocessing

Tweets generally form a huge, noisy dataset that requires numerous pre-processing steps. The tweets in our datasets were processed using Python libraries namely Natural Language Toolkit (NLTK), NeatText, and Regular Expression (RE). The following techniques were applied in the following order:

- Any hashtag in the dataset conveys important information. The topic of almost every social media site is represented using a hashtag such as #COVID19, #CoronavirusOutbreak, #COVID, and #Coronavirus. As a result, we simply eliminated the "#" symbol.
- To avoid considering words with uppercase letters different from the same words with lowercase letters, all words are converted to lowercase.
- Stop words are the common frequently occurring words, which should be ignored because they do not provide any meaningful information. Removing these stop words is especially useful when building text classification models to reduce the amount of data.
- The fourth step is to remove hyperlinks, @ mentions, multiple white spaces, emojis, and punctuation, as well as special characters. This is because all these do not affect the understanding of the sentences of the tweets and do not help in detecting sentiment.
- We used a lemmatization technique to reduce inflected words to their basic forms, e.g., “Mostly” to “Most” or “viruses” to “virus.”

V. Models and Experimental Setting

In this section, we discuss the details of the ML and DL models used in our experiments. We also discuss the details of the experiments including the experimental setting and hyperparameters.

A. ML Models

The Python library Scikit-learn has many ML classifiers. We employed five ML classification models as follows:

Multinomial Naïve Bayes (MNB) - This idea of this model is based Naïve Bayes Theorem, which calculates the probability of each of a set of classes, and the class with the highest probability is considered the winning class for classification. It is thus suitable for text classification with multiple classes [6].

Linear Support Vector Classification (Linear SVC) – This model treats each data item as a vector and searches for a linear separator of the classes in their vector space. In higher-dimensional spaces, a hyperplane is computed. Features extracted from the input data are classified based on this plane. To obtain the optimal hyperplane, a margin between the classes is maximized based on the distance between the nearest vectors of the classes, which are called support vectors [3].

Extreme Gradient Boosting (XGBOOST) – This model is based on the decision tree classifier. An ensemble of such trees is usually employed, e.g., gradient boosting machines. XGBoost is an extension to this model for speedup and improved performance, with minimal resources [10].

Ridge Classifier (RC) – This is a variation of Ridge Regression. This linear classifier is suitable when the number of features is high and exceeds the number of observations regardless of whether the problem is binary or multi-class [11].

Stochastic Gradient Descent (SGD) – The idea of gradient descent is to use data to compute the gradient of an objective function to reach its minima. One of the three variants of gradient descent is SGD, which frequently updates the model's parameters with high variance, causing significant variations. This gives it the capability to find new and hopefully, superior local minima in comparison to its counterparts [12] [13].

TABLE II. Overview of the Datasets

Dataset	Positive		Negative		Neutral		Total	
	count	%	count	%	count	%	count	%
COVIDSenti_A	1,968	6.6	5,083	16.9	22,949	76.5	30,000	100
COVIDSenti_B	2,033	6.8	5,471	18.2	22,496	75.0	30,000	100
COVIDSenti_C	2,279	7.6	5,781	19.3	21,940	73.1	30,000	100
COVIDSenti	6,280	7.0	16,335	18.2	67,835	74.9	90,000	100

TABLE III. DEFAULT HYPERPARAMETERS

RC		Multinomial NB		SGD				Linear SVC		XGBoost			
Parameter	**Value**	**Parameter**	**Value**	**Parameter**	**Value**	**Parameter**	**Value**	**Parameter**	**Value**	**Parameter**	**Value**	**Parameter**	**Value**
alpha	1.0	alpha	1.0	loss	hinge	random_state	None	penalty	12	max_depth	3	colsample_bytree	1
fit_intercept	True	force_alpha	warn	penalty	12	learning_rate	optimal	loss	squared_hinge	learning_rate	0.1	colsample_bylevel	1
copy_X	True	fit_prior	True	alpha	0.0001	eta0	0.0	dual	True	n_estimators	100	reg_alpha	0
max_iter	None	class_prior	None	l1_ratio	0.15	power_t	0.5	tol	0.0001	silent	True	reg_lambda	1
tol	0.0001			fit_intercept	True	early_stopping	False	C	1.0	objective	multi:softprob	scale_pos_weight	1
class_weight	None			max_iter	1000	validation_fraction	0.1	multi_class	ovr	booster	Gbtree	base_score	0.5
solver	auto			tol	0.001	n_iter_no_change	5	fit_intercept	True	n_jobs	1	random_state	0
positive	False			shuffle	True	class_weight	None	intercept_scaling	1	nthread	None	seed	None
random_state	None			verbose	0	warm_start	False	class_weight	None	gamma	0	missing	None
				epsilon	0.1	average	False	verbose	0	min_child_weight	1		
				n_jobs	None			random_state	None	max_delta_step	0		
				random_state	None			max_iter	1000	subsample	1		

B. DL Model

The basic idea of any DL architecture is to emulate how the human brain works. Such architecture is typically layered, and each layer provides the input to the next. The network learns as each training dataset item is fed to it and saves what it has learnt in the form of weights. As an output is generated, it is compared to the correct desired one and in the case of a mismatch, the weights are updated. This continues until the global error is minimized. The network then becomes ready for its task.

One of the most popular DL architectures is Recurrent Neural Network (RNN). The typical structure of an RNN cell is shown in Fig. 2. As shown in the figure, over time, the network is visualized as a set of similar sequential feedforward cells. This gives it the ability to memorize data with long-term dependencies such as language models. Nevertheless, as the length of such a sequence increases, the problem of vanishing gradient affects the ability of the model to continue learning. Accordingly, other variants have been proposed.

LSTMs are the most common types of RNNs intended to address the vanishing gradient problem of a typical RNN cell. A typical LSTM cell is shown in Fig. 3. As shown in the figure, its success stems from the existence of a cell state that acts as a memory and three gates, namely input, output and forget, to decide how much information to input, output, or forget respectively. This provides a LSTM cell with a relatively long memory in comparison to the short memory of a basic RNN cell. This gives it the name 'Long Short Term Memory.'

GRU [15] is a simplified variant of LSTM. A typical GRU cell is depicted in Fig. 4. As shown in the figure, the GRU cell has no cell state like LSTM cells. Instead, it uses an invisible state. It also has two gates only, a reset gate and an update gate, that incorporate the functions of the input and forget gates of LSTM. The update gate acts like a long-term memory while the reset gate acts as a short-term memory. This makes GRU easier and faster to train and run. Nevertheless, GRU cells might not be as capable as LSTM cells when it comes to memorizing long-term dependencies as is the case with language models. Thus, this tradeoff needs to be taken into consideration to benefit from their efficiency while avoiding their drawback.

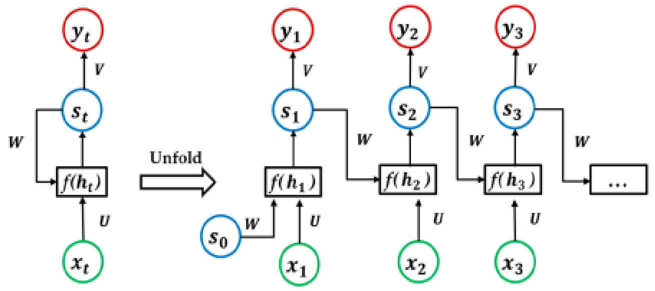

Fig. 2. Schematic diagram of RNN cell [14].

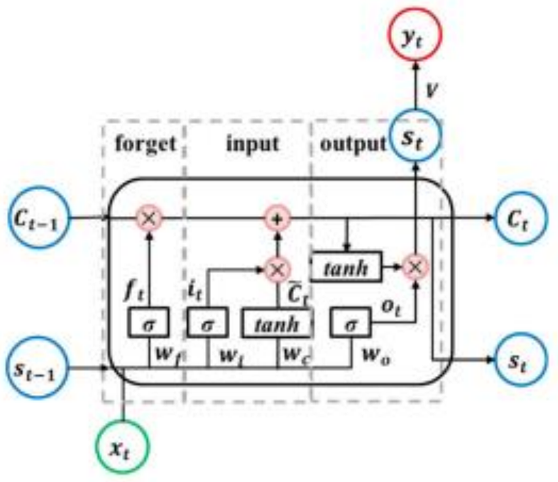

Fig. 3. Schematic diagram of LSTM cell [14].

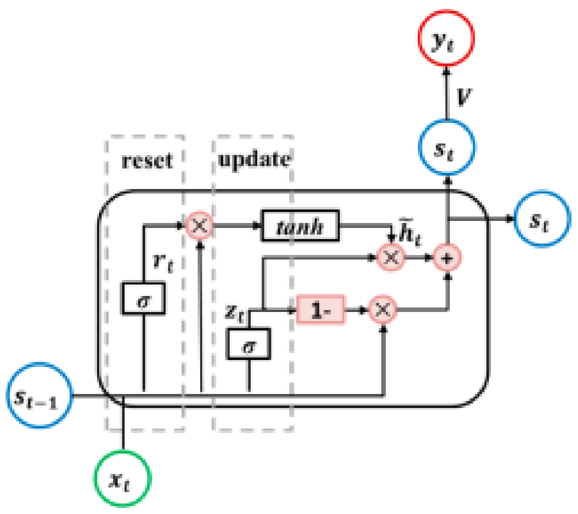

Fig. 4. Schematic diagram of GRU cell [14].

C. *Experimental Setting of the ML Models*

The problem with text is that it is generally considered unstructured. To be able to process it, it needs to be preprocessed and converted into numeric data. This is enabled through NLP feature extraction techniques. In this paper, we used two different feature extraction techniques, namely count vectorizer and TF-IDF. In count vectorizer approach, text in a given document or tweet is vectorized based on its count. This is computed for each word that appears in the entire corpus. On the other hand, TF-IDF also takes into consideration the uniqueness and thus the importance of each term in the whole corpus [16]. In addition to vectorizing text, each ML model that we employed has its own set of hyperparameters. For each ML model, we utilized the default values as shown in Table III.

1) K-Fold Cross-Validation (KCV) is a commonly used approach for evaluating the performance of ML models while reducing the chance of overfitting. In KCV, the dataset is first partitioned into a set of K equal-sized folds. Each fold is used once as the test data, while using the rest of the data for training. After exhausting the folds (through K-iterations, the average of a given performance metric is computed [17]. Stratified K-Fold Cross-Validation (SKCV) is an extension of KCV, where class distribution in the original data is taken into consideration when sampling [18]. Accordingly, SKCV is preferred over KCV in the case of unbalanced class distributions [19]. In our experiments, we used SKCV, specifically 10-fold cross validation, to split the data into training and testing, while computing the average accuracy of the different folds.

D. *Experimental Setting of the DL Model*

In our experiments, we decided to use the efficient GRU model. The model layers include, in addition to the basic components, one embedding layer, two dropout layers, and a dense layer. The embedding layer is used for preprocessing and vectorizing a vocabulary of size 10,000. It is followed by a dropout layer, with dropout rate of 0.2 to help reduce the complexity of the model and the probability of the model overfitting. GRU layer follows the drop out layer with 128 units followed again by another dropout layer with a rate of 0.2. At the end, a dense layer with 3 neurons and a Softmax activation function is used to get the probabilities of the possible classes. In the experiments, we used Global Vectors (GloVe)-based word representation to learn word embeddings from text documents. A pre-trained word embedding GloVe with two billion tweets, 27 billion tokens, and 1.2 million vocabularies was used to generate a 100-dimensional vector from text.

Finally, we divided the dataset into training data and testing data subsets. 80% of tweets were used for training purposes and the remaining 20% were used for testing. Training tweets help identify the data patterns and thus reduce error rates of the test data subset used for the assessment of the model performance. We also used Adam optimizer to minimize the cross-entropy loss with a batch size of 36 and 10 epochs. This is in addition to using Softmax activation for multi-classes probabilities, and early stopping to reduce overfitting on a Google Colab GPU.

TABLE IV. ML CLASSIFIERS WITH DIFFERENT EVALUATION METRICS USING COUNT VECTORIZER

Model dataset		**COVIDSenti_A**				**COVIDSenti_B**				**COVIDSenti_C**				**COVIDSenti**			
		Proposed model/Metric (Accuracy – Precision – Recall – F1-Score)															
		Accuracy	**Precision**	**Recall**	**F1-Score**	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
	RC	84.84%	84%	85%	84%	84.12%	83%	84%	83%	82.78%	82%	83%	82%	84.79%	84%	85%	84%
	MNB	78.51%	75%	79%	76%	78.03%	75%	78%	76%	76.97%	74%	77%	75%	78.81%	76%	79%	77%
Count Vectorizer	**SGD**	**86.75%**	**86%**	**87%**	**86%**	**86.07%**	**85%**	**86%**	**85%**	**85.05%**	**84%**	**85%**	**84%**	**86.19%**	**85%**	**86%**	**85%**
	Linear SVC	84.73%	84%	85%	84%	84.27%	84%	84%	84%	82.76%	82%	83%	82%	85.60%	85%	86%	85%
	XGBoost	84.99%	84%	85%	83%	84.11%	84%	84%	82%	82.43%	82%	82%	81%	84.10%	84%	84%	82%

VI. Results and Discussion

In this section, we provide the results and discussion of the ML models and the DL model. We then compare both results to each other and to the results of the surveyed research.

A. Results of the ML Models

Table IV shows comparison between the five ML models that used count vectorizer embeddinhg in terms of accuracy, precision, recall, and F1-score. The results of the experiment show that the SGD model performed better than all the other models that used count vectorizer on COVIDSenti_A, COVIDSenti_B, COVIDSenti_C, and COVIDSenti with 86.75%, 86.07%, 85.05%, and 86.19% accuracy respectively. Moreover, RC, Linear SVC, and XGBoost showed almost similar performance with average accuracy of 84.85%, 84.17%, 82.66%, and 84.83% on the four datasets respectively. Finally, MNB showed the worst performance with accuracies of 78.51%, 78.03%, 76.97%, and 78.81% respectively. The same is also true about the other performance metrics except that when considering the whole dataset, namely COVIDSenti, linear SVC was almost as good performing as SGD. Thus, it may be considered to be the second best. What is interesting about SGD is that it is the only model to show almost the same performance on all datasets with little variability.

Table V presents the results of comparing the five ML models on COVIDSent_A, COVIDSenti_B, COVIDSenti_C, and COVIDSenti in the case of the TF-IDF feature extraction technique. Unlike the case of count vectorizer, linear SVC outperformed SGD, RC, XGBoost and MNB and showed the best performance with 85.50%, 85.10%, 83.65% and 85.59% accuracy on COVIDSenti_A, COVIDSenti_B, COVIDSenti_C and COVIDSenti respectively. In this table, we compare our models with the best models in the surveyed literature using the same datasets, and with embeddings other than count vectorizer. Looking at the table, it is clear that linear SVC was also able to outperform these models across all datasets. Nevertheless, its performance is still lower than that of SGD with count vectorizer. Additionally, its performance is not as uniform as that of the latter. Again, MNB showed the worst performance among its counterparts.

B. Results of the DL Model

Fig. 5 and Table VI show the learning accuracy and loss curves and the classification reports of the GRU-based DL model that we employed. As shown in the figure, learning proceeded successfully until different epochs for each dataset, after which though training accuracy continued increasing and training loss continued decreasing. Similarly, validation accuracy started decreasing and its loss started increasing suggesting overfitting. Hence, we had to use early stopping that was different from the different datasets. Additionally, as shown in Table VI, the neutral class had the best performance among its other two counterparts.

In Table VII, we compare the results of our proposed DL model with those of the best performing models in the research studies using the same datasets. As shown in the table, our model was not able to outperform the other models. Nevertheless, none of the model was the best across all datasets and none showed uniform performance among them either. For example, though DCNN-(GloVe+CNN) was the best on COVIDSenti_C and COVIDSenti, Conv1D-LSTM + Glove was the best on COVIDSenti_A and COVIDSenti_B. Additionally, both had variable performance among the four datasets.

C. Discussion of Results

Based on the experiments and the related studies on the same datasets, among the ML models [8, 9], the proposed SGD model with count vectorizer showed the best performance, and linear SVC with TF-IDF showed the second best. On the other hand, among the DL models [8, 9], none was the best in all cases. DCNN- (GloVe+ CNN) showed the best performance in the case of COVIDSenti and COVIDSenti_C, while Conv1D-LSTM + Glove showed the best performance in the case of COVIDSenti_A and COVIDSenti_B. These models had slightly higher performance in comparison to SGD with count vectorizer. Nevertheless, the latter was the only model that showed almost uniform performance among the four datasets. This implies that the proposed SGD model with count vectorizer is the most reliable among its counterparts.

Another important observation in our experiments is that the highest performances encountered, whether using ML models or DL models, were all lower than 90%. This suggests that the capabilities of the various ML and DL models might be limited to this performance level. This is the main challenge and limitation of our work, which indicates that it's worth exploring large language transformers. This is the topic of our future work.

TABLE V. ML Classifiers with Different Evaluation Metrics Using TF-IDF or FastText

Model dataset		**COVIDSenti_A**	**COVIDSenti_B**	**COVIDSenti_C**	**COVIDSenti**
Existing model/Accuracy [8, 9]					
TF-IDF	SVM	83.9%	83.0%	82.8%	84.5%
FastText	RF	82.3%	84.1%	80.2%	84.5%
Proposed model/Accuracy					
TF-IDF	RC	84.67%	84.1%	82.76%	84.82%
	MNB	77.31%	76.03%	74.49%	76.28%
	SGD	83.31%	82.52%	80.86%	81.16%
	Linear SVC	**85.50%**	**85.10%**	**83.65%**	**85.59%**
	XGBoost	84.54%	83.6%	82.18%	83.95%

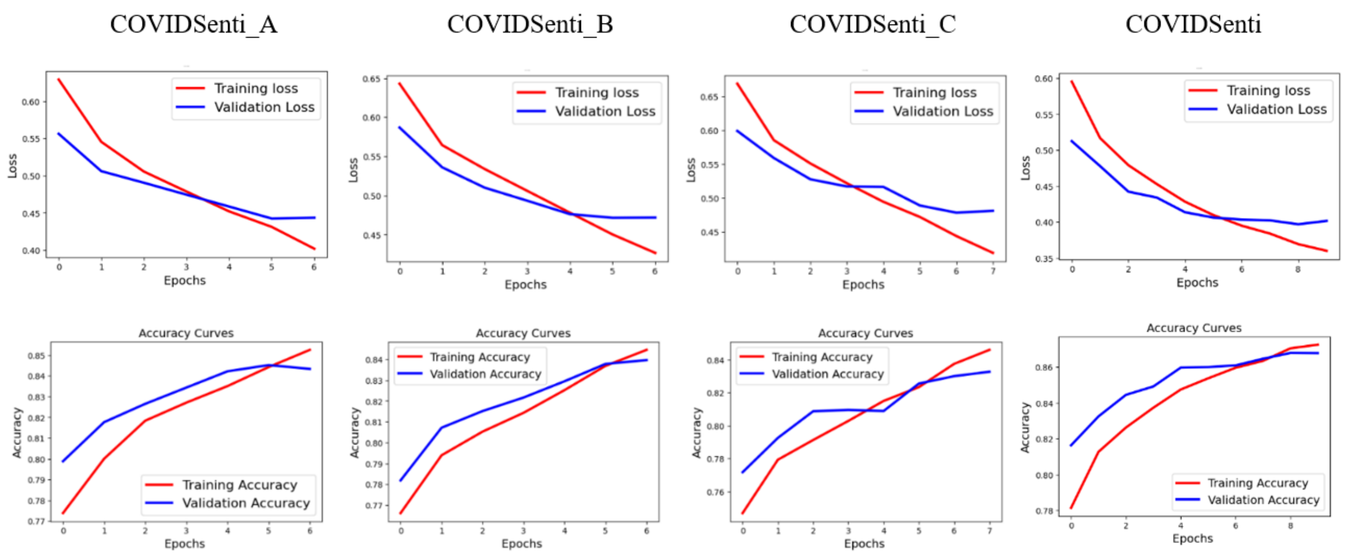

Fig. 5. GRU model accuracy and loss for each dataset.

TABLE VI. DL CLASSIFICATION REPORT USING GLOVE

Sentiment	**COVIDSenti**			**COVIDSenti_A**			**COVIDSenti_B**			**COVIDSenti_C**		
	Precision	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
Neutral	88.8%	94.5%	91.5%	86.0%	95.1%	90.3%	84.9%	95.9%	90.0%	83.5%	96.3%	89.5%
Negative	80.8%	68.1%	73.9%	75.3%	54.4%	63.2%	80.1%	53.7%	64.3%	83.3%	53.5%	65.1%
Positive	73.2%	86.8%	84.3%	84.0%	83.3%	86.8%	84.3%	84.0%	83.3%	86.8%	84.3%	46.6%
Accuracy	86.8%			84.3%			84.0%			83.3%		

TABLE VII. COMPARISON OF PROPOSED DL CLASSIFIER ACCURACY WITH BASELINE

Model dataset	**COVIDSenti_A**	**COVIDSenti_B**	**COVIDSenti_C**	**COVIDSenti**
Existing model/Accuracy [8, 9]				
DCNN- (GloVe+ CNN)	83.4%	83.2%	**86.4%**	**86.9%**
Conv1D-LSTM + Glove	**87.0%**	**86.1%**	84.4%	**86.9%**
Proposed model/Accuracy				
GRU+GloVe pretrain	86.8%	84.3%	84.0%	83.3%

VII. CONCLUSION AND FUTURE WORK

This paper is concerned with sentiment analysis of tweets during pandemics with COVID-19 as a prototype. Our related work review showed that as the dataset size increases, the accuracy generally tends to decrease. This suggests that using a small dataset might provide misleading results that cannot be generalized. Hence, it is better to consider large datasets and try to improve analysis performance on it. Accordingly, in this paper we considered a huge dataset namely COVIDSenti and its three sub-datasets. We experimented with a set of machine learning techniques (MNB, SVC, XGBoost, RC, and SGD) and a customized deep-learning GRU model. The experiments showed that unlike the models that we tested, and the state-of-the-art models on the same dataset, SGD technique with count vectorizer showed quite constantly high performance on all the four datasets. As future work, we intend to use grid search with the ML models to figure out whether we could obtain even better results. We will also examine additional ML and DL models aiming at achieving higher performance. This is in addition to possibly other datasets. Finally, since all results of ML and DL models were less than 90%, we intend to start working with large language model transformers to figure out whether superior results could be achieved.

REFERENCES

[1] C. Shofiya and S. Abidi, "Sentiment analysis on COVID-19-related social distancing in Canada using Twitter data," *International Journal of Environment Research and Public Health*, vol. 18, 2021.

[2] N. Chintalapudi, G. Battineni and F. Amenta, "Sentimental analysis of COVID-19 tweets using deep learning models," *Infecteoud Disease Reports*, vol. 13, 2021.

[3] P. Gupta, S. Kumar, R. Suman and V. Kumar, "Sentiment analysis of lockdown in India during COVID-19: A case study on Twitter," *IEEE Transactions on Computational Social Systems*, vol. 8, 2021.

[4] B. Ramya, S. Shetty, A. Amaresh and R. Rakshitha, "Smart Simon bot with public sentiment analysis for novel COVID 19 tweets stratifcation," *SN Computer Science*, vol. 2., 2021.

[5] R. Goel and R. Sharma, "Studying leaders & their concerns using online social media during the times of crisis: A COVID case study," *Social Network Analysis and Mining*, vol. 11, 2021.

[6] S. Vernikou, A. Lyras and A. Kanavos, "Multiclass sentiment analysis on COVID-19-related tweets using deep learning models," *Neural Comuting and Applications*, vol. 34, 2022.

[7] Y. Qi and Z. Shabrina, "Sentiment analysis using Twitter data: a comparative application of lexicon- and machine-learning-based approach," Social Network Analysis and Mining, vol. 13, 2023.

[8] U. Naseem, I. Razzak, M. Khushi, P. W. Eklund and J. Kim, "COVIDSenti: A large-scale benchmark Twitter data set for COVID-19 sentiment analysis," *IEEE Transactions on Computational Social Systems*, vol. 8, 2021.

[9] Z. Jalil et al., "COVID-19 related sentiment analysis using state-of-the-art machine learning and deep learning techniques," *Frontiers in Public Health*, vol. 9, 2022.

[10] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785-794.

[11] I. Sarker, "Machine learning: Algorithms, real world applications and research directions," *SN Computer Science*, vol. 2, 2021.

[12] S. Ruder, "An overview of gradient descent optimization algorithms," *Clinical Orthopaedics and Related Research*, 2016.

[13] M. Alagözlü, "Stochastic Gradient Descent Variants and Applications," *Università della Svizzera Italiana*, 2022.

[14] H. Zhao, Z. Chen, H. Jiang, W. Jing, L. Sun and M. Feng, "Evaluation of three deep learning models for early crop classification using Sentinel-1A imagery time series—A case study in Zhanjiang, China," *Remote Sensing*, vol, 11, no. 22, 2019.

[15] J. Chung, C. Gulcehre, K. Cho and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," presented in *Deep Learning and Representation Learning Workshop*, 2014.

[16] Y. Fu and Y. Yu, "Research on text representation method based on improved TF-IDF," *Journal of Physics*, 2020.

[17] M. Wayahdi, D. Syahputra and S. Ginting, "Evaluation of the K-Nearest neighbor model with K-fold cross validation classification," *Data Mining, Image Processing, Artificial Intelligence, Networking*, vol. 9, 2020.

[18] S. Prusty, S. Patnaik and S. Dash, "SKCV: Stratified K-fold cross-validation on ML classifiers for predicting cervical cancer," *Frontiers in Nanotechnology*, 2022.

[19] S. Widodo, H. Brawijaya and S. Samudi, "Stratified K-fold cross validation optimation on machine learning for prediction," *Sinkron*, vol.7, no. 4, 2022.