Emotional Tweets: A Convolutional-BiLSTM Approach to Emotion Classification

Hongyuan Ran

University of California, Irvine

ran_hongyuan@qq.com

Abstract. We introduce a neural network model using a convolutional and bidirectional long short-term memory framework for the purpose of emotion classification. Our work utilizes a corpus of more than 200,000 English tweets sourced from the Twitter API, collected between January 2020 and September 2021, related to the COVID-19 pandemic. By employing pre-trained 50-dimensional GloVe word embeddings for vectorizing the short-form text data, our model achieves a significantly higher accuracy than what would be anticipated from random classification. It demonstrates a 65.39% success rate in categorizing each tweet into one of the five key emotions, markedly outperforming the random baseline of 41.5%. Along with discussing pertinent previous studies that have shaped our model's design, we also detail the necessary steps for data acquisition and processing to generate the data used, as well as the method for developing, training, and fine-tuning the model.

Keywords: emotion classification; Machine learning model; neural network; BiLSTM; social media analysis; Twitter data; text data processing.

1 Introduction

We use language not just to convey facts, but also our emotions. Assessing emotional expressions in textual data can be a daunting task for human interpreters, and when dealing with hundreds of thousands or even millions of words without the aid of software, it becomes virtually impossible. In this report, we share our efforts in creating, refining, and training a neural-network model rooted in a convolutional and bidirectional long short-term memory framework, with the goal of categorizing the dominant emotion in brief text data samples, such as tweets.[1] This model was trained using recent tweets from the last two years, curated from the Twitter API in relation to the COVID-19 pandemic, to maintain the freshness of the dataset and avoid the scrutiny of an overly-studied or standard dataset. The need for such a model is clear: With an accurate and dependable model that can parse emotional expressions in publicly accessible data from a widespread platform like Twitter, interested entities could assess public reaction to, for instance, significant global incidents. A practical example would be policymakers using the tool to measure public emotional reactions to understand the effect of their policies, thereby using these insights to influence and direct future policy.

A. Rauf et al. (eds.), *Proceedings of the 3rd International Conference on Management Science and Software Engineering (ICMSSE 2023)*, Atlantis Highlights in Engineering 20,
https://doi.org/10.2991/978-94-6463-262-0_74

We provide a detailed chronicle of our project journey, from the initial literature review to establishing the data gathering and processing pipeline, to the design and training of our model and the evaluation of its effectiveness. Furthermore, we comprehensively discuss potential socio-ethical implications and outcomes of our work, as well as its possible future applications.

2 Prior Work & Research

This section delves into and encapsulates preceding studies addressing challenges akin to the one we've identified. There are several techniques for categorizing the sentiment of tweets, where each tweet is allocated a label of positive or negative sentiment. However, discerning a specific emotion like happiness, sadness, fear, anger, surprise, or denial, associated with a particular tweet proves more challenging.[2] Nevertheless, different approaches have been employed to address this issue, falling into two primary categories: psychological principles and machine learning methodologies. The psychological background [3] necessary for this application includes establishing methods to identify the emotion conveyed in a tweet. These principles prove beneficial for manually annotating Twitter datasets, which can subsequently be input into machine learning models. In terms of machine learning techniques, various strategies have been adopted, including logistic regression, naive bayes emotion classifier, transformers like BERT, CNNs, RNNs, among others. We concentrate on three particular previous studies, exploring them thoroughly. These papers employ techniques such as RNN architecture, CNN architecture, and the utilization of BERT.

Prior Work #1: "EMOCOV: Machine 1earning for emotion detection, analysis and visnalization using COVID-19 tweets"[4]

The goal of the authors of this study is to understand societal reactions to COVID-19 by analyzing the sentiment of related tweets. They have specifically designed a model that assigns one of the following labels to tweets: neutral, optimistic, happy, sad, surprise, fear, anger, denial, joking, and pessimistic. In addition, they have crafted a model that identifies the section of each tweet that contributes most to its overall emotional tone. Since our research is more closely linked with the study's primary objective of deducing the overall sentiment of a tweet, we focus on examining its model. As for data collection, the authors opted to generate their own dataset, having found no existing datasets suitable for their purposes. Tweets were gathered using the Twitter Streaming API via Python's Tweepy Library, with a subset of the data manually labelled by three students. The model structure adopted in this study is explained below:

- *Input:* A vector representing a preprocessed tweet is fed into the network.
- *Embedding:* By using the GloVE algorithm, each word in the input (tweet) is mapped to an embedding vector.
- *Bidirectional Long-Short Term Memory (Bil .STM):* Useful features are extracted using the Bil.STM.
- *Attention:* Determines how semantically close words are through an attention score.
- *Auxiliary Features:* Using Python's Natural Language Toolkit, features such as emoticons and punctuation are extracted.

- *Another Bil STM Layer:* Incorporates output from the auxiliary layer.
- *Output:* Sigmoid activation function is used in a fully connected layer to output the emotion.

Training, validating and testing was performed on the manually labeled portion of the dataset and on an augmented dataset with existing labeled datasets. The model was evaluated using four different metrics as follows. For the manually labeled set, the results were as follows; accuracy: 0.8647, Jaccard: 0.5366, F1-Micro: 0.5514, F1-Macro: 0.5392. For the augmented data: accuracy: 0.8951, Jaccard: 0.6475, F1-Micro: 0.6893, F1-Macro: 0.6342.

Prior Work #2: "Multi-Channel Convolutional Neural Network for Twitter Emotion and Sentiment Recognition"[5]

Similar to prior work #1, Similar to prior work #1, the aim of this study is to identify emotions in tweets. Yet, unlike earlier studies, the approach used in this study involves a multi-channel convolutional neural network (CNN), to classify tweets into various emotional categories: joy, sadness, anger, love, thankfulness, fear, surprise, guilt, and disgust. The CNN model was selected for its rapid computational capabilities and its high performance with noisy data, such as the often grammatically incorrect tweets. For gathering data, several pre-existing labeled datasets were employed. Each dataset was individually tested with the final model, with performance compared across each one. Data cleaning procedures included employing the Tweet'Tokenizer package, transforming tweets to lowercase, formatting user mentions, expanding abbreviated words, and utilizing the emoji Python library. The classification model includes the following layers;

- *Embedding*: Two GloVe vectorizations are used. The first one embeds the tweets themselves and the other one uses auxiliary features such as emoticons, emoticons and hashtags.
- *Convolutional*: Two CNN layers are applied to each of the embedded matrices along with a RelU activation function.
- *Pooling*: A max-over pooling strategy is used for both channels to determine which features are most important.
- *Hidden:* The feature vectors created from the pooling layer along with a few external lexicon feature vectors are concatenated into one feature vector. Then it is passed into another ReLU activation function.
- *Output:* A fully connected layer with a softmax activation function is used to output the prediction.

To assess the model, three distinct versions of it were examined. The initial variant solely utilized tweet embeddings, the second incorporated auxiliary features along with the tweets, and the final one integrated external lexicon features. The third variation yielded the highest accuracy, attaining approximately 87%.

Prior Work #3: Emotion and Sentiment Analysis of Tweets Using BERT[6]

The prior research aims to conduct sentiment analysis by classifying tweets under one of the given categories: sadness, fear, anger, or happiness. The methodology employed in this study is based on the BERT model. An already available tagged dataset

is used in this research. For maintaining an even distribution of data, the researchers have chosen an equal number of tweets from each emotion group. The model used is as follows:

- Preprocess tweets: Removing mentions, urls and retweets.
- Tokenize tweets.
- Feed into BERT pretrained model. This paper used the BERT-Base version which has twelve encoders with eight layers in each encoder, four layers of multi-head self attention followed by four forward layers.
- Apply a dense classification layer followed by a softmax to determine the predicted sentiment.

Moreover, the researchers conducted experiments with the BERT structure, first testing the uncased BERT base model, then the cased version. The uncased model displayed an accuracy rate of 89%, while the cased version demonstrated a slightly higher accuracy at 90%.

Another noteworthy study is 'Determining Word–Emotion Associations from Tweets by Multi-Label Classification'. In this work, the authors proposed a methodology for expanding a multi-label emotion lexicon based on a collection of unlabelled tweets using multi-label classification [7]. This approach presents an alternative way of handling the challenge of emotion classification in tweets, focusing on the association between words and emotions.

Prior Work *#4: "A Deep Learning-Based Approach for Multi-Label Emotion Classification in Tweets"[8]*

This paper proposes a deep learning model for multi-label emotion classification in tweets. The authors argue that tweets often express multiple emotions simultaneously, and therefore, a multi-label classification approach is more appropriate than a single-label approach. The model they propose uses a combination of convolutional neural networks (CNNs) and long short-term memory (LSTM) networks, which are both types of artificial neural networks. In terms of system performance, the proposed model achieved an accuracy of 0.89, a precision of 0.90, a recall of 0.89, and an F1-score of 0.89. These results indicate that the proposed model can effectively classify the emotions in tweets, even when multiple emotions are present. This research provides a valuable reference point for our work, as it demonstrates the potential effectiveness of deep learning approaches for emotion classification in tweets.

Relevance to our work:

After reviewing various earlier studies, we've chosen to employ a hybrid of strategies recommended by prior works #1 and #2, specifically incorporating both Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) approaches. As previously noted, CNNs are excellent for handling noisy data and for extracting features regardless of their spatial location within the dataset. Conversely, RNNs are well-equipped to process sequential data, a trait applicable to tweet analysis. Given the relevance of both methods to tweet analysis and their robust performance individually, we plan to develop a model incorporating both strategies. We will also use a Bi-directional Long Short Term Memory (BiLSTM) to mitigate the vanishing gradient problem often associated with RNNs. BiLSTM is favored over LSTM as it can access information

from both forward and backward directions. Additionally, we will utilize GloVe for data vectorization, as earlier research indicates its effectiveness for this task. A summary of the model we've implemented is provided below and will be explored in greater depth in the following sections.

1. Processing and cleaning the data: Removing punctuation and emoticons.
2. Tokenizing the data and turning the labels into a one-hot vectorization.
3. Vectorizing the data: Using GloVe embeddings.
4. Splitting data into training, validation and test set.
5. Convolution Layer: Generate features.
6. Batch Normalization: Allows training to be faster.
7. ReLU activation function: Applying non-linearity.
8. Pooling Layer: Extract most prominent features.
9. BIiLSTM Layer: RNN layer.
10. Linear Layers and apply softmax to generate outputs.

3 Data Collection & Processing

The first stage in our data gathering involved acquiring the actual content of tweets. Initially, we extracted the tweet ID numbers from an existing dataset from earlier research. Using these IDs in combination with the Tweepy Python library for Twitter's API, we pulled the corresponding tweets, downloaded them, and documented them, together with their labels and other relevant details, in another csv file. This file is located in the data_from_api directory in the Github repository. The Python script data_intake.py contains the source code used for this primary data collection and API engagement, and it's designed to automate the entire process. However, we encountered several challenges during this phase, the most notable being the rate limit set by Twitter on the volume of API interactions per given time period. Given that our level of access was free/educational, the API only allowed us to make 900 queries every 15 minutes or one query per second. Consequently, to assemble a sufficiently large dataset, we automated the Twitter API querying process over approximately 58 hours of operation. One objective during data processing was to operate within reasonable space restrictions to avoid dealing with massive data files that would be too large to efficiently load into the model. Hence, we divided our data collection into several manageable parts. Even though the runtime complexity wasn't a major concern, primarily due to the API rate limit acting as the primary constraint, we adhered to best practices to ensure the program's performance was not affected by unnecessary operations or similar factors. We successfully collected a total of 207,863 tweets. After collection, we further processed the tweets to only include the content of the tweet and the main emotion label for easy use in training the model with batches.

To guarantee the precision of labeling and to minimize the instances of non-descriptive "no specific emotion" samples, we assigned an emotion to each tweet based on the highest score determined in this study. However, we introduced a filter employing a threshold value. If the emotional scores for any tweet differ by 0.01 (with scores spanning [0, 1]) or less, the tweet is assigned a 'no emotion' label. The finalized dataset,

following this process, contains the tweet content and their respective labels, which could be: 'fear', 'sadness', 'happiness', 'anger', or 'no emotion'. This data is available in the tweet_labels directory, and the source code for generating the labels is found in actual_code/label_data.py.

For collecting data for the test set, we opted for the second methodology outlined in the original proposal. Using Twitter's search API in combination with the Tweepy Python library, we gathered a total of 400 tweets, which were filtered based on the English language and keywords indicative of strong primary emotional expression (for instance, words like "amazing" or "fantastic" suggesting happiness). This strategy allows for a generalized evaluation accuracy that is not influenced by potential biases within the COVID-19 focused dataset. Moreover, it facilitates manual labeling to ensure the quality of the test set, meaning that labels for the test examples are human-verified.

A query consists of a list of six keywords associated with one specific label, joined by 'or' operators, and excluding retweets. Each query pulls 100 tweets per labeled category (fear, happiness, anger, sadness). The 'no emotion' set is omitted from the retrieval process as tweets lacking emotion, which are often factual or news-oriented, could not be easily extracted using keywords. These tweets can vary greatly in their coverage and usually don't have easily identifiable common words. The context of each retrieved tweet is manually checked to align with its label. The data can be found in the manually_labeled directory, and the source code for executing the queries is incorporated in manual_label_tweets_gen.py.

During the data processing phase, actual tweets gathered from the API are processed through a tokenizer function. This function removes punctuations and emoticons, subsequently transforming the sentences into lists of word tokens. Each list corresponds to a single tweet, achieved by using Keras' text_to_word_sequence. The tweet's related emotion label is concurrently converted into a one-hot encoded list. An instance of this process could involve transforming the tweet "I hate birthdays!" and its emotion 'anger' into the following format: [I, 'hate', 'birthdays'], [‘1’, ‘0’, ‘0’, ‘0’, ‘0’]. Our main dataset and manually labeled set are tokenized using tokenize_csv and tokenize_manual functions respectively in main_model.ipynb. The output of this tokenization is then embedded using pre-trained GloVe embedding that has 6B tokens, a 400k vocabulary, and a 50-dimension feature. The selection of this specific embedding is due to its minimum viable design. It uses substantially less memory than the 42B and 840B embeddings, and it is believed that the 50-dimensional vector representation strikes a reasonable balance between computational efficiency and abstraction. As a result of this process, each tweet is converted into a 50x50 matrix representation, and tweets with less than 50 words are zero-padded. The function load_pretrained_embeddings is used to load the GloVe pre-trained embeddings into a dictionary, with word tokens as keys and embeddings as values. The tweet_vectorize function employs this embedding dictionary to embed its input. The final GloVe embedding output is then randomized along with their labels across the entire set, before it's partitioned into 80% for training, 10% for validation, and 10% for testing. The obtained test set is merged with the manually labeled set made with keywords to enable more extensive testing standards. The code for

this is located within the split_train_val_test function and can be found in our repository. Lastly, Figures 1 and 2 demonstrate a comparison of data distribution before and after our preprocessing, contrasting our results with those of the previous research team.

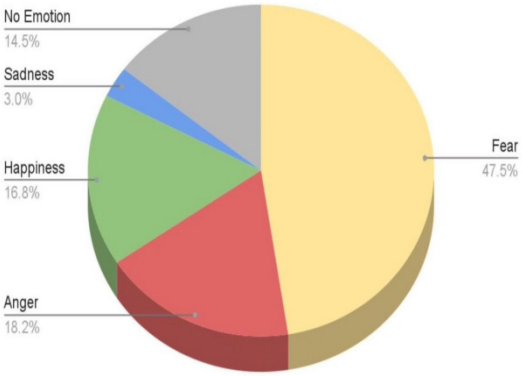

Fig. 1. Tweet Emotion Distribution, as per Gupta et al

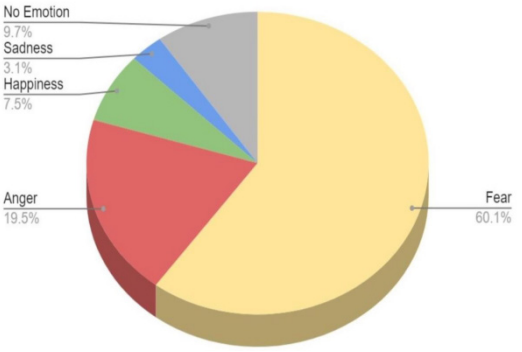

Fig. 2. Tweet Emotion Distribution with 0.01 Score Thresholding

4 Model Implementation & Training

To execute and train our model, we convert the processed and vectorized data into tensor entities. These have dimensions of 50 x 50 x N for the data tensor, and 1 x 5 x N for the labels tensor, where N signifies the total number of examples we possess, 207,863 in this case. Each layer of the data tensor corresponds to a 50 x 50 matrix representation of a tweet from our dataset, which has been vectorized using previously mentioned pre-trained GloVe embeddings. Likewise, each layer of the label tensor holds a 1 x 5 one-hot encoded vector representing the dominant emotion expressed in the matching layer of the data tensor. Figure 3 below offers a visual explanation of the dimensions of the tensor data we're dealing with.

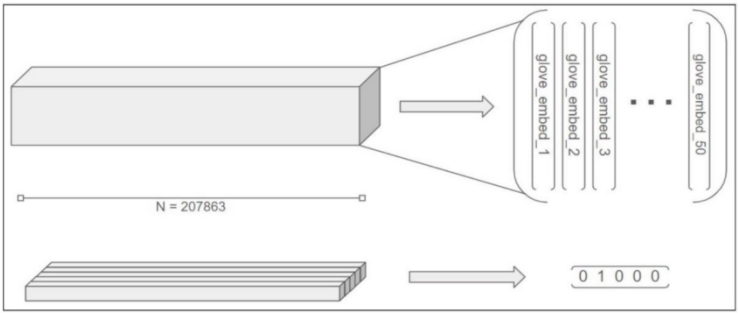

Fig. 3. Dimensionality Visualization of Data Tensors

Considering the data's structure, we start to build our convolutional bi-LSTM model. We start with a convolutional layer, using a kernel size of 3 and a stride of 1 as the initial parameters. This establishes a 50 x 3 convolution window that progresses over each tweet's matrix representation, moving from one token vector to another. The convolution layer assists in the extraction of interrelation and information present between adjacent words in a specific tweet. Post-convolution, the outputs undergo batch normalization to ensure uniform input ranges across mini-batches during the training phase. Ideally, such normalization promotes a stable learning and training process, thereby reducing the training duration. Post-normalization, the inputs go through a Rectified Linear Unit (ReLU) activation function, followed by max-pooling to emphasize the most significant information. After this sequence of operations – convolution, normalization, activation, and pooling – the data proceeds to the bi-LSTM layer. To maintain dimensional compatibility between the convolutional output and the bi-LSTM input, a minor computation of the anticipated output shape is conducted, based on the input data's shape and the convolution kernel size. Using this computed input shape and a single-layer bi-LSTM of size 4, the data is fed into the bi-LSTM. The output from the bi-LSTM layer goes into two successive fully-connected, dense layers, transforming the output shape into a 1 x 5 prediction vector suitable for applying a softmax function. This prediction is then one-hot encoded for comparison with the corresponding tweet's label to evaluate accuracy. Visualizations of the layers used in the model are provided in Figures 4 and 5. Specific details and information about our model implementation can be viewed in our GitHub repository.

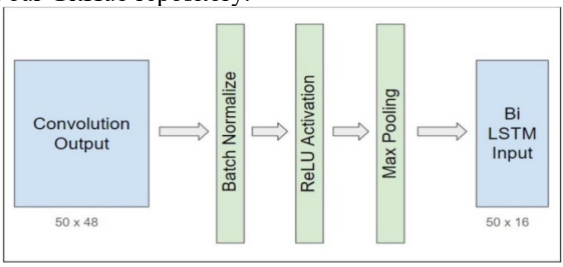

Fig. 4. Visualization of Convolution through Maximum Pooling Model Layers

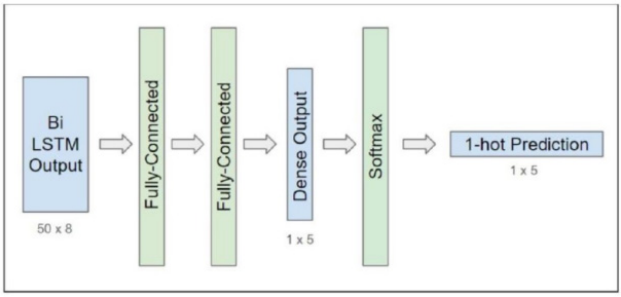

Fig. 5. Visualization of Bi-LSTM through Output Prediction Model Layers

During the model's training phase, we utilized batching primarily due to the large size of the dataset. Initial attempts to run training cycles on the entire dataset, which contained 166000 tweets, failed as the tensor object was too large for the model to handle due to GPU cache size limitations. Consequently, we opted to train in batches, where the batch size became an adjustable hyperparameter linked with the training process. We investigated a variety of parameters both for the model, including the convolution kernel size, stride size, or bi-LSTM hidden layer size, and for the training process, such as the learning rate, number of training epochs, or the batch size during hyperparameter tuning. Despite the project's time limitations, which prevented us from applying a thorough hyperparameter exploration method like grid searches, we managed to significantly enhance model accuracy through manual parameter search and tuning.

5 Results

Starting with an initial parameter set that included {epochs = 10, learning rate = 0.001, batch size = 10000, loss = BCELoss, output channels = 30, kernel size = 3, stride = 1, hidden size = 4}, we attained accuracies of 60.49% on the training set and 60.34% on the validation set. After manually tuning these parameters, we discovered that a different set of parameters {epochs = 50, batch size = 500, loss = CELoss, output channels = 50}—with all other parameters remaining the same—led to a considerable increase in accuracies, with 69.2% on the training set and 65.77% on the validation set, and a final test set accuracy of 65.39%. Importantly, the parameters that appeared to have the most immediate and substantial effect on accuracy improvement were training with a smaller batch size over more epochs. You can find all the combinations of various parameter values that were manually tested and tuned in the Jupyter Notebook containing our model, which is accessible on our GitHub repository. Figures 6 and 7 below depict the model's loss and accuracy over 50 epochs of training with the final set of hyperparameters. As a comparison, a random baseline classifier picking classes uniformly at random would expect an accuracy of 41.55%, making our model's performance of slightly above 65% accuracy significantly better.

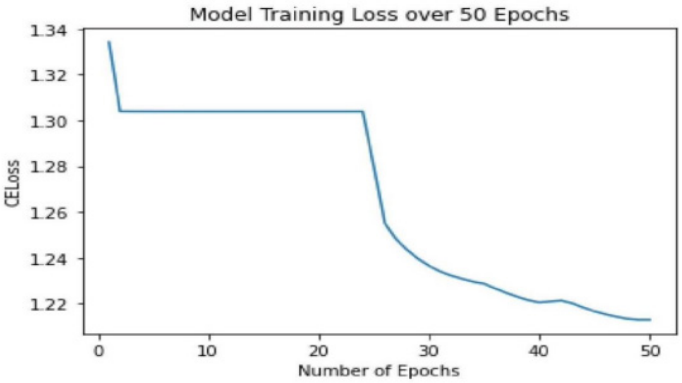

Fig. 6. Model Training Loss with Final Parameter Set

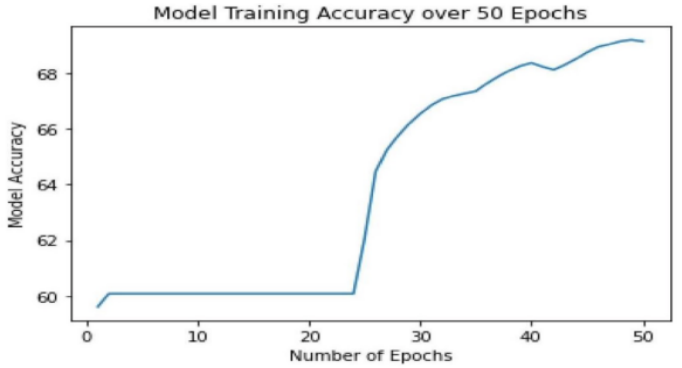

Fig. 7. Model Training Accuracy with Final Parameter Set

6 Broader & Ethical Impact Statement

Our model's fundamental goal is to identify sentiment components in casual writing platforms, thereby aiding diverse decision-making processes. With a few minor adjustments, our model can be employed on other informal communication platforms such as Facebook and Quora to examine the sentiment in opinions. On a theoretical level, this approach can be generalized to the broader process of using algorithms to extract hidden information from human-written texts. Consequently, this can serve as a valuable tool in sectors like marketing, governance, and policy-making. For instance, in product marketing, sentiment analysis of customer reviews can guide product development and research. Given the design of our model to utilize the COVID-19 dataset, it could also be leveraged to gain a better understanding of public reactions to significant events and evaluate the effectiveness of current policies, thereby informing future decisions. Nevertheless, the scope of its applications raises certain concerns: are ordinary citizens comfortable with governments and other third-party entities using technology built on their data? These technological grey areas present complex socio-ethical dilemmas that

require attention in various areas of machine learning and artificial intelligence research. In the context of our model, significant concerns include privacy, model design specifics, and potentially jeopardizing the preservation of lesser-known languages through unintentional exclusion.

The first ethical concern we need to address pertains to privacy. Our model functions by gathering tweets from the general public. A potential problem with this method is that users are not explicitly consenting to have their tweets used for model training, and they might not be aware of this potential application of their data. Despite all Twitter users accepting the privacy terms and conditions while setting up an account, most of them do not read these conditions and, consequently, remain unaware of the potential uses of their generated and shared content. Furthermore, it's challenging to outline all the possible purposes for which user content could be utilized in the terms and conditions, given that tweets are public and can be applied to various models and complex applications. As a result, the official terms and conditions often fail to precisely outline potential uses of the tweets, which could be a concern for users. One possible solution to this issue could be to provide users with an option to completely prevent their tweets from being accessible via Twitter APIs.

Alongside the privacy implications directly tied to the construction of our model, it's also crucial to consider the potential problems stemming from its expansion and adaptation. To enhance the model's precision, one potential tactic could be to examine each user's historical activity to categorize new posts. This involves identifying patterns in a specific user's past posts, which could feasibly increase the accuracy of future predictions. However, this enhancement leads to increased privacy worries since models might then maintain records of the dataset's users. Moreover, as previously highlighted, our model can serve functions such as scrutinizing public reactions to new legislation or international incidents, which could heighten user concerns over how their posts are used. Therefore, the various potential expansions and improvements to our work inevitably carry associated ethical and privacy challenges.

A further ethical concern related to our project revolves around the model's capability to adapt to various languages. A significant portion of the digital content created and disseminated on today's internet is in a small selection of widely spoken languages such as English, Chinese, Spanish, among others [9]. As a result, less prevalent languages are not fully integrated into the highly tech-oriented contemporary society or emerging technological fields like Natural Language Processing (NLP), given their limited digital footprint. The data available to represent languages with a smaller online presence is considerably less. This data scarcity makes it challenging to adequately train models that would yield high performance. Thus, a model like ours, built on the existing data corpus, unintentionally perpetuates the marginalization of minority languages due to lack of representation, significantly limiting the technology's accessibility to communities that speak these underrepresented languages. Therefore, it's crucial to prioritize the model's ability to adapt to a wide array of languages, thus enhancing its accessibility.

7 Conclusion

In summary, we find that a convolutional bi-LSTM based structure serves as a feasible groundwork for the task of emotion identification in text. Although the accuracy levels of our model fall short of those reported in earlier studies, it notably outperforms the expected baseline, and we have highlighted several potential enhancements. Future efforts for enhancing the model's performance could involve more comprehensive and systematic hyperparameter tuning during training, possibly through algorithmic methods like grid search. Additionally, insights from previous studies suggest that the integration of multi-headed attention and auxiliary features like emoticons, URL links, or hashtags could considerably enhance predictive accuracy. Therefore, the addition of diverse layers in our model or the inclusion of various extra features in the dataset, or the employment of higher-dimensionality embeddings of the dataset could be potential steps towards boosting model performance. Furthermore, we propose that the model could be adapted and used for other types of text data beyond tweets, adding to its versatility and expanding its range of applications.

References

1. Mohammad, S. (1970) #emotional tweets, ACL Anthology. Available at: https://aclanthology.org/S12-1033/ [Accessed: 30-Jan-2023].
2. Janssens, O., Walle, R.V. de and Hoecke, S.V. (1970) A learning based approach for real-time emotion classification of Tweets, SpringerLink. Available at: https://link.springer.com/chapter/10.1007/978-3-319-19003-7_7 [Accessed: 16 July 2023].
3. A. Bandhakavi, N. Wiratunga, S. Massie, and D. P, "Emotion-aware polarity lexicons for Twitter sentiment analysis," 09-Dec-2017. [Online]. Available: https://onlinelibrary-wiley-com.myaccess library.utoronto.ca/doi/pdfdirect/10.1111/exsy.12332. [Accessed: 30-Jan-2022].
4. Y. Kabir and S. Madria, "EMOCOV: Machine learning for emotion detection, analysis and visualization using COVID-19 tweets," Online Social Networks and Media, 16-May-2021. [Online]. Available: https: //reader.elsevier.com/reader/sd/pii/S2468696421000197?token=8CF8B1C1797BC134400A3D6383E7H6185BE52029E035B9B611C4105072C35C060B74165F49B2COFOA3A2C4ECB5 5E7BDE&originRegion=us-ecast-1&originCreation=20220124035631. [Accessed: 30-Jan-2022]
5. J. Islam, R. E. Mercer, and L. Xiao, "Multi-Channel Convolutional Neural Network for "Twitter Emotion and Sentiment Recognition," 02-Jun-2019. [Online]. Available: https://aclanthology.org/N19-1137.pdf. [Accessed: 30-Jan-2022].
6. A. Chiorrini, C. Diamantini, A. Mircoli, and D. Potena, "Emotion and sentiment analysis of tweets using BER'T." [Online]. Available: http://ceur-ws.org/Vol-2841/DARLI-AP_17.pdf
7. Marque, F.B. (no date) Determining word-emotion associations from tweets by ... - IEEE xplore. Available at: https://ieeexplore.ieee.org/abstract/document/7817108/ [Accessed: 16 July 2023].
8. Jabreel, M. and Moreno, A. (2019) A deep learning-based approach for multi-label emotion classification in Tweets, MDPI. Available at: https://www.mdpi.com/2076-3417/9/6/1123 [Accessed: 16 July 2023].

9. J.Johnson, “Internet: Most common languages online 2020,” Statista, 26-Jan-2022. [Online]. Available: https://www.statista.com/statistics /262946 /share-of-the-most-common-languages-on-the-inter net/. [Accessed: 09-Apr-2022].