Deep Learning-Based Sentiment Analysis on Twitter Data

Sudipta Ghosh

(23MA60R03)

Supervisor: Prof. Adrijit Goswami

Indian Institute of Technology Kharagpur
West Bengal-721302, India

Agenda

1. Introduction to Sentiment Analysis and Application
2. How to approach Sentiment Analysis
3. Twitter Dataset for Sentiment Analysis
4. Implementation
5. Conclusion

What is Sentiment Analysis?

Sentiment analysis is a natural language processing technique that identifies the polarity of a given text. There are different flavors of sentiment analysis, but one of the most widely used techniques labels data into positive, negative and neutral. For example, let's take a look at these tweets mentioning @VerizonSupport:

- *dear @verizonsupport your service is straight 💩 in dallas.. been with y'all over a decade and this is all time low for y'all. i'm talking no internet at all."* → Would be tagged as "Negative".

- *"@verizonsupport i've sent you a dm"* → would be tagged as "Neutral".

- *"thanks to michelle et al at @verizonsupport who helped push my no-show-phone problem along. order canceled successfully and ordered this for pickup today at the apple store in the mall."* → would be tagged as "Positive".

Why is sentiment analysis important?

Improving customer support

Sentiment analysis helps support teams provide personalized responses by understanding the customer's mood. AI-powered chatbots can detect and escalate urgent issues quickly, while machine learning algorithms rank topics by urgency and identify feedback showing frustration. These capabilities streamline support operations and enhance overall customer experience.

Building a stronger brand presence

Sentiment analysis enables brands to monitor social media and understand how customers truly feel about them. Beyond tracking sales, it reveals emotional responses to product launches and reviews, offering deeper insights into customer perception and brand impact..

Conducting market research

Sentiment analysis of the broader market helps organizations spot trends, track competitor performance, and respond to real-time events like viral posts or endorsements—unlocking timely growth opportunities.

How does sentiment analysis work?

Data Collection

01

Sentiment 140 dataset containing 1.6 million tweets. Data set is collected from Kaggle website where the data is stored in csv format.

Data Preprocessing

02

cleaning tweets by removing noise such as URLs, mentions, hashtags, emojis, and applying normalization and lemmatization

Model Building

03

- LSTM
- GRU
- BiLSTM

Implementation

04

Finally I have deployed the model in a web server using flask. A web interface screenshot is attached to showcase.

Data set Description

The data set used for this project is the **Sentiment140 Dataset**, which contains **1.6** million tweets collected via the Twitter API. It is commonly used for training sentiment analysis models. The data set includes the following columns:

- **target**: Indicates the sentiment polarity of the tweet — either positive or negative.
- **ids**: A unique identifier assigned to each tweet.
- **date**: The timestamp when the tweet was posted.
- **flag**: Refers to the query used to extract the tweet.
- **user**: The username of the account that posted the tweet.
- **text**: The actual content or message of the tweet.

Data Preprocessing

Html Tags and URL

"Check out my latest blog post on sentiment analysis!
 Read more at https://example.com/post123"

Lemmatization

Original Tweet:
"The kids were running and having fun in the park."

After Lemmatization:
The kid be run and have fun in the park.

Punctuation and StopWords

"I can't believe this is happening... what are you even doing?"

Punctuation: !, . . . , ?, "

Stopwords: I, can't, this, is, what, are, you

ChatWords

"BTW, just found the *perfect* coffee spot ☕… IG it's my new fave."

BTW: by the way
IG: i guess

Emojis & Hashtag

"Just got my new phone today 😍 #Joy"

Emoji Name:
`smiling_face_with_heart_eyes`

Occurrences of repeated letters

Before:
I am sooooo happppppppy.

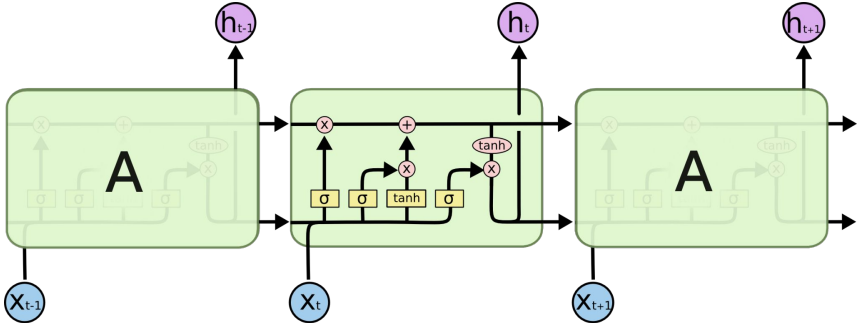
After:
I am so happy.

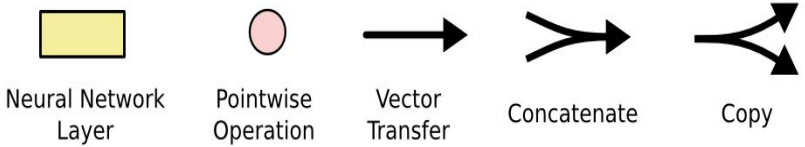
Deep Learning Model : LSTM model architecture

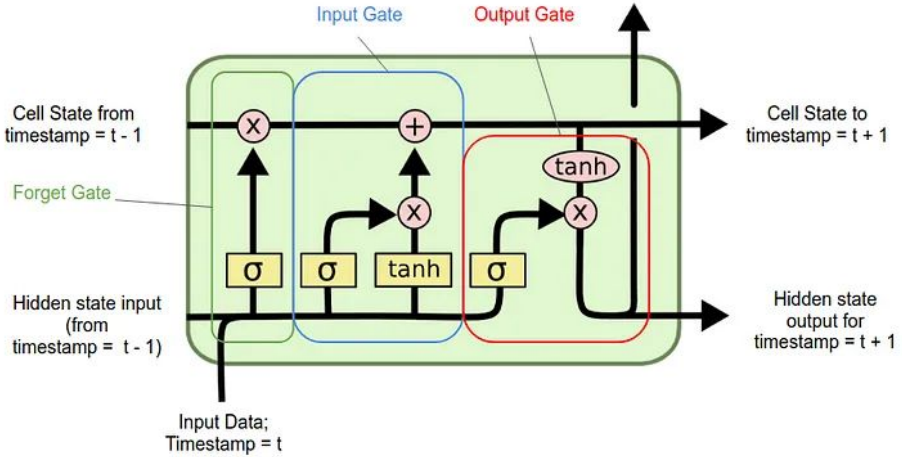
Lstm Architecture:

- **Embedding Layer:** Converts word indices to 128-dim dense vectors using a vocabulary of 20,000 words and max sequence length of 150.
- **LSTM Layer:** Uses 128 LSTM units with L2 regularization; outputs only the final hidden state for classification.
- **Dropout Layer:** Randomly drops 50% of neurons during training to prevent overfitting.
- **Dense Layer:** Fully connected layer with 64 units and ReLU activation; applies L2 regularization.
- **Output Layer:** Single neuron with sigmoid activation; outputs probability for binary sentiment classification.

LSTM Cell

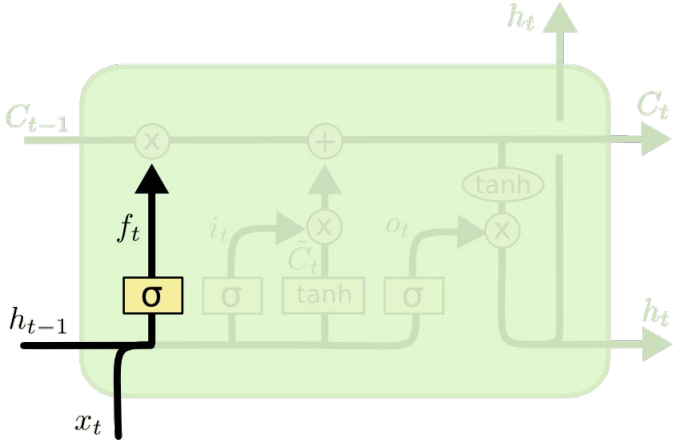

h_{t-1}
h_t
h_{t+1}
A
A
tanh
σ
σ
tanh
σ
x_{t-1}
x_t
x_{t+1}


Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy


Input Gate
Output Gate
Cell State from timestamp = t - 1
Cell State to timestamp = t + 1
Forget Gate
tanh
σ
σ
tanh
σ
Hidden state input (from timestamp = t - 1)
Hidden state output for timestamp = t + 1
Input Data; Timestamp = t

Forget Gate

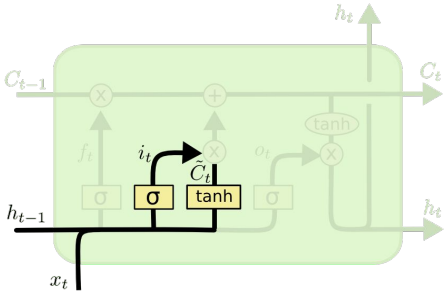
The first step in our LSTM is to decide what information we're going to throw away from the cell state.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

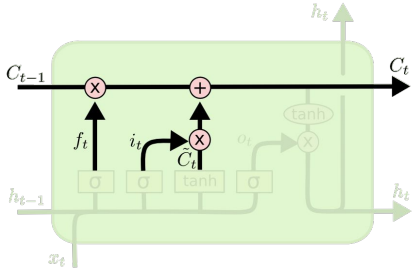
Input Gate

The next step is to decide what new information we're going to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values we'll update. Next, a tanh layer creates a vector of new candidate values, C_{t}, that could be added to the state.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

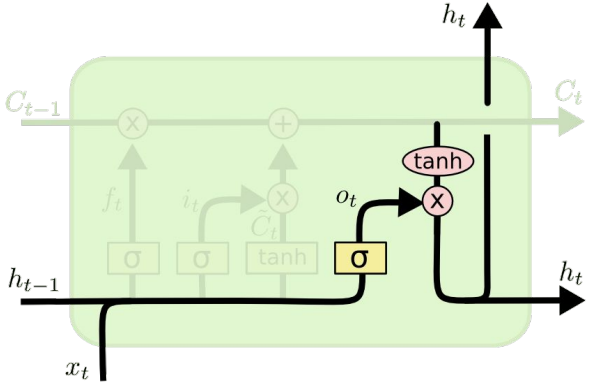
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Output Gate

Finally, we need to decide what we're going to output. This output will be based on our cell state, but will be a filtered version.

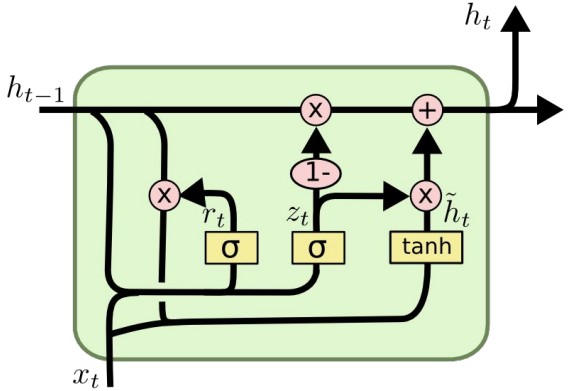


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

GRU Unit

A slightly more dramatic variation on the LSTM is the Gated Recurrent Unit, or GRU, introduced by Cho, et al. (2014). It combines the forget and input gates into a single "update gate."



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

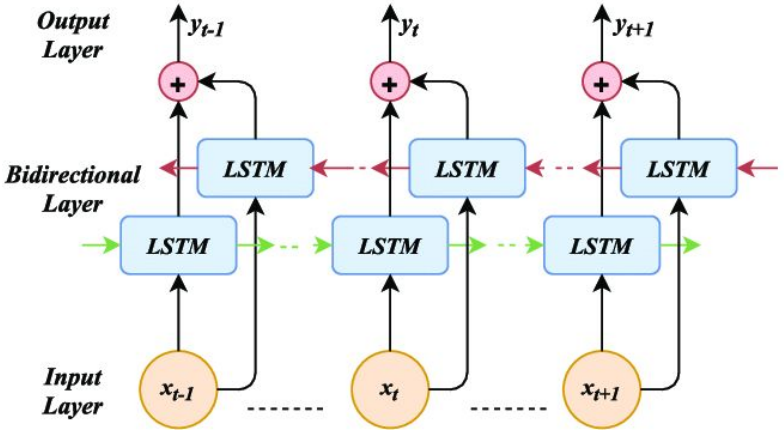
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

BiLSTM

A Bidirectional LSTM (BiLSTM) is an enhancement of the standard LSTM that enhances contextual understanding by processing input data in both forward and backward directions. It consists of two LSTM layers:

- One layer processes the sequence from left to right (past to future)
- The other processes it from right to left (future to past).



BiLSTM Model Summary

Layer (type)	Output Shape	Param #
embedding (Embedding)	(128, 100, 100)	77,609,200
bidirectional (Bidirectional)	(128, 100, 512)	731,136
bidirectional_1 (Bidirectional)	(128, 256)	656,384
dense (Dense)	(128, 64)	16,448
dropout (Dropout)	(128, 64)	0
dense_1 (Dense)	(128, 1)	65

Total params: 237,039,701 (904.23 MB)

Trainable params: 79,013,233 (301.41 MB)

Compile The Model

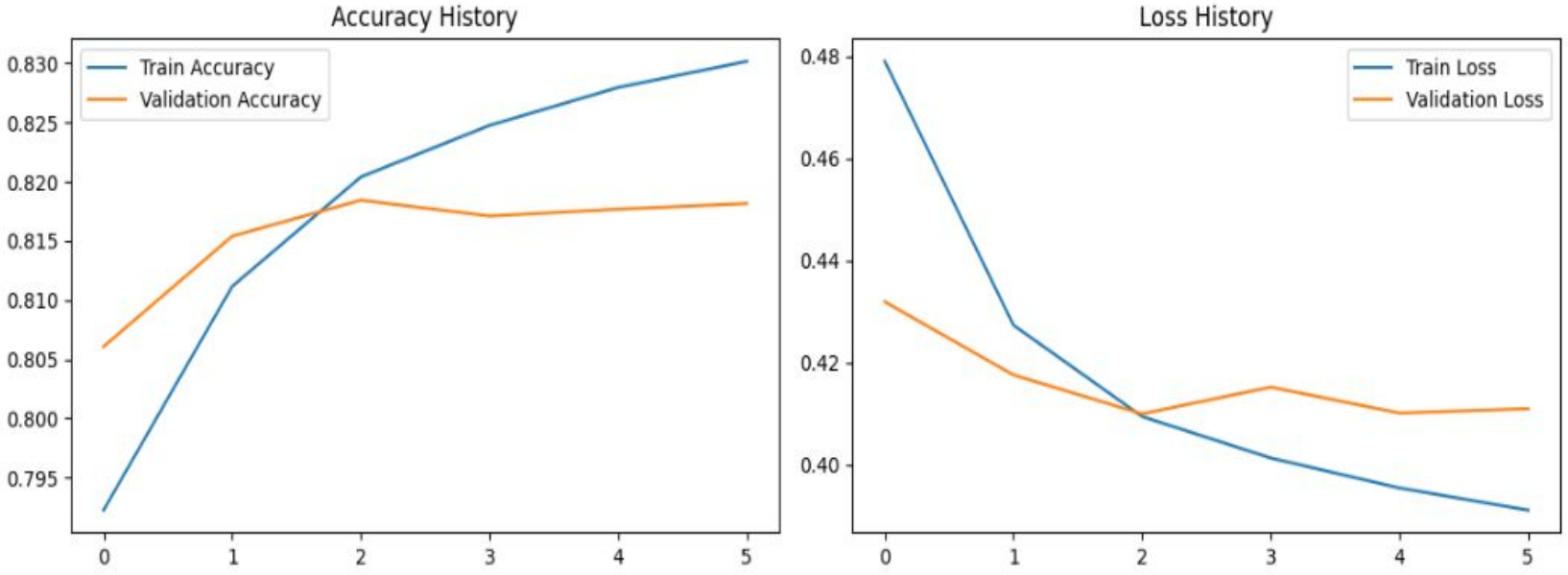
- **optimizer='adam' :** Sets the optimization algorithm to Adam, a widely used and efficient method for training deep learning models due to its adaptive learning rate capabilities.

- **loss='binary_crossentropy' :** Defines the loss function as binary cross-entropy, which is appropriate for binary classification tasks by measuring the difference between predicted probabilities and actual class labels.

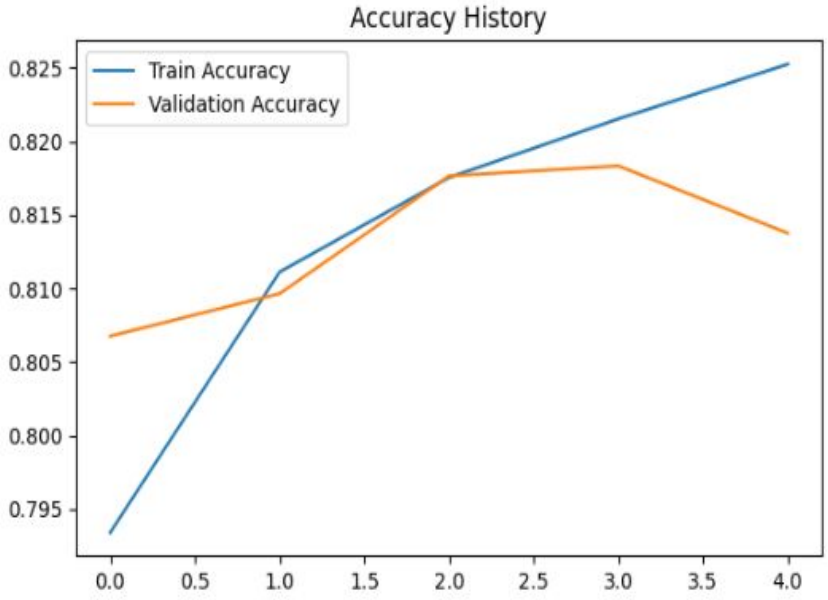
- **metrics='accuracy' :** Specifies accuracy as the performance metric, which reflects the proportion of correctly classified instances during training and evaluation.

Plotting The Loss And Accuracy

BiLSTM Model



GRU Model


Accuracy History
Train Accuracy
Validation Accuracy
0.825
0.820
0.815
0.810
0.805
0.800
0.795
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Loss History
Train Loss
Validation Loss
0.47
0.46
0.45
0.44
0.43
0.42
0.41
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

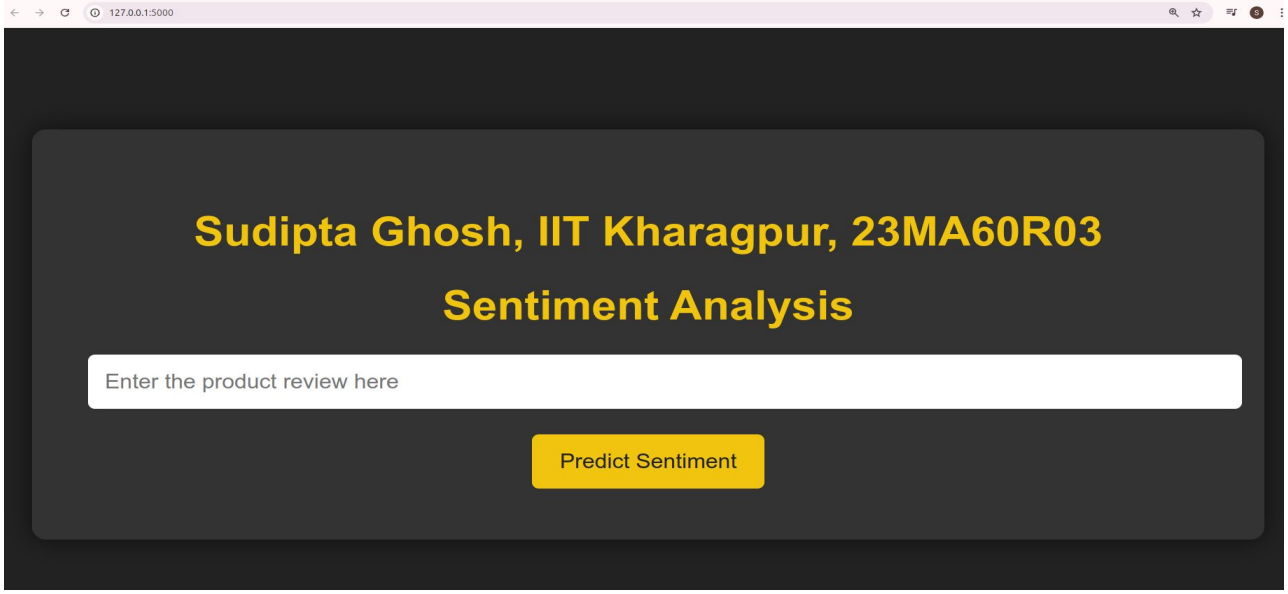
Comparison of different Models

I have used my dataset on different deep learning models like Simple **RNN**, **LSTM**, **BiLSTM**, **GRU**, **BERT**

Model	Accuracy	Precision	Recall	F1 Score
LSTM	0.8158	0.8153	0.8181	0.8167
BiLSTM	0.8184	0.8211	0.8142	0.8177
GRU	0.8183	0.8305	0.8000	0.8149
BERT	0.8737	0.8627	0.8849	0.8795

Implementation

I have deployed the Sentiment Analysis model using Flask. For the user interface, I designed an index.html file using HTML and CSS. A screenshot of the web application is attached below. The model is running on localhost 127.0.0.1.5000



Challenges of sentiment analysis

- **Sarcasm:** It is extremely difficult for a computer to analyze sentiment in sentences that comprise sarcasm. Consider the following sentence, *Yeah, great. It took three weeks for my order to arrive*. Unless the computer analyzes the sentence with a complete understanding of the scenario, it will label the experience as positive based on the word *great*.

- **Negation** : Negation is the use of negative words to convey a reversal of meaning in the sentence. For example, I wouldn't say the subscription was expensive. Sentiment analysis algorithms might have difficulty interpreting such sentences correctly, particularly if the negation happens across two sentences, such as, *I thought the subscription was cheap. It wasn't.*

- **Multipolarity** : Multipolarity occurs when a sentence contains more than one sentiment. For example, a product review reads, "I'm satisfied with the phone's performance but disappointed with its battery life.". It becomes difficult for the software to interpret the underlying sentiment. You'll need to use aspect-based sentiment analysis to extract each entity and its corresponding emotion.

References

- Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011, June). Learning word vectors for sentiment analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1 (pp. 142-150). Association for Computational Linguistics.

- Prerana Singhal, Pushpak Bhattacharya. Sentiment Analysis and Deep Learning: A survey. Dept. of Computer Science and Engineering , Indian Institute of Technology, Powai Mumbai, India.

- Jan Deriu, Mark Cieliebak. Sentiment Analysis using Convolutional Neural Networks with Multi-Task Training and Distant Supervision on Italian Tweets. Zurich University of Applied Sciences, Switzerland. deri@zhaw.ch, ciel@zhaw.ch.

- N. Girdhar and K. K. Bharadwaj, “Social status computation for nodes of overlapping communities in directed signed social networks,” Integrated Intelligent Computing, Communication and Security, pp. 49–57, 2019.

- M. Anand Kumar, S. Sachinkumar. Sentiment Analysis of Tweets in Malayalam Using Long Short-Term Memory Units and Convolutional Neural Nets. Mining Intelligence and Knowledge Exploration: 5th International Conference, MIKE 2017, Hyderabad, India, December 13–15, 2017.

Thank You