**Machine Learning vs Deep Learning Approach for Sentiment Analysis on Twitter Data**

**Swapnil sonawane Alisha Gaikwad Sneha Thakur**

[swapnils0098@gmail.com](mailto:swapnils0098@gmail.com) alishagaikwad03@gmail.com [thakursneha578@gmail.com](mailto:thakursneha578@gmail.com)

**ABSTRACT-** Sentiment Analysis is a process of categorizing whether the text is positive, negative, or neutral. Not only this but it also includes the emotions like happiness, sadness, anger, fear, and surprise. Sentiment analysis can be used in various fields, some well-known fields including online shopping. Sentiment analysis can solve real-time issues and is a crucial task in Natural Language Processing (NLP). We can use traditional Machine Learning algorithms, such as Support Vector Machine (SVM), Tree-based technique or Naïve Bayes have been widely used for sentiment analysis. The advent of deep learning (DL) techniques, like CNN, RNN, or state-of-the-art methods changed this field to capture the more complex pattern in data. This paper presents the comparative study of sentiment analysis using ML and DL techniques. We used ML and DL algorithms against Twitter Sentiment Analysis data and compared the algorithms based on accuracy, computational efficiency, and ability to adopt complex patterns in large datasets. This paper provides insights into the trade-off between ML and DL approaches for Sentiment Analysis, further guiding researchers and practitioners in choosing the appropriate approach for their specific tasks.

**Key Words:** Sentiment Analysis, Machine Learning, Deep Learning, RNN, LSTM, GRU

1. INTRODUCTION

As the whole world connects to the internet, data is everywhere and the famous quote “Data is the new Oil” is relevant to the current world. This data includes customer feedback, reviews on the products they buy, and people’s opinions on various topics on different social media platforms. For e-commerce websites like Amazon, Flipkart, and Walmart they must analyze the customer's feedback, and review the product to increase the product sales. Also, the manufacturer can improve and address the customer's concerns, to enhance the customer's experience and satisfaction by analyzing the sentiment of customers' reviews and feedback. Sentiment analysis can help business monitor their reputation by tracking the comments, and social media reviews. Understanding the users review product development can improve their products. Companies like Twitter, Facebook, and Instagram can analyze people’s opinions on current trending topics like elections, and wars to keep the platform safe and make sure that the platform stays neutral for all users and does not get biased in a specific direction. These companies also used sentiment analysis to monitor the tweets and posts to make sure that it is appropriate and follow all the community guidelines, and if someone goes against the guidelines, they remove their content from the platform. We can use Machine Learning and Deep Learning algorithms to classify the sentiment. However, which technique is suitable for problem statements depends on the data size and ability to adapt to new contexts.

There are a few observations on which the system architecture is proposed.

1. LITERATURE REVIEW
2. **Lexical or Rule-Based Approach:**

In “Twitter Sentiment Analysis Using Lexical or Rule-Based Approach: A Case Study” [1] Sheresh Zahoor and Rajesh Rohila use Lexical or Rule Based (unsupervised technique) for Twitter sentiment analysis. Using Twitter API they create 4 different datasets. 1. Haryana Assembly Polls 2. ML Khattar 3. The sky is pink (movie) 4. United Nations General Assembly (UNGA). The steps they follow to collect the data and analyze the sentiments are:

1. Data Collection
2. Data pre-processing
3. Part of Speech tagging (POS)
4. Sentiment analysis using an in-built dictionary
5. Data Collection:

To collect the data from Twitter they use Twitter API and collect the tweet and save it in CSV format. CSV file contains the date, text, retweet, hashtag, and followers.

1. Data Pre-Processing:

To prepare data for sentiment analysis they perform various operations on data including tokenization or Bag-of-words, N-gram Extraction, Stemming and Lemmatization, and StopWords removal.

1. Part-of-Speech (POS):

Process of automatically tagging each word by their grammatical feature such as Noun, Pronoun, verb, adverb, etc.

1. Model Evaluation:

They used TextBlob and VADER built-in libraries available in Python. TextBlob is an open-source NLTK-based library whereas VADER (Valence Aware Dictionary and sentiment Reasoner) is used for lexicon-based sentiment analysis. The result they conclude is:

Case 1: Haryana Assembly Polls

Case 2: ML Khattar

Case 3: The sky is pink

Case 4: United Nations General Assembly

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sentiment | Case 1 | | Case 2 | |
| Text Blob | VADER | Text Blob | VADER |
| Positive | 29.7% | 44% | 58.5% | 58.5% |
| Negative | 12.0% | 17.6% | 9.6% | 9.6% |
| Neutral | 58.3% | 38.5% | 32% | 20.5% |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sentiment | Case 3 | | Case 4 | |
| Text Blob | VADER | Text Blob | VADER |
| Positive | 64.1% | 62.8% | 36.2% | 33.4% |
| Negative | 12.1% | 12.7% | 12.6% | 40.1% |
| Neutral | 23.8% | 24.5% | 251.2% | 26.5% |

The conclusion of this case study found that the results obtained from unsupervised techniques are not accurate and subject to change.

1. **Deep Learning Approach:**

Vasily D. Derbentsev and Vitalii S. Bezkorovainyi et al. published “A Comparative Study of Deep Learning Models for Sentiment Analysis of social media texts” [2] paper. The author of this paper presents a comparative study of a deep learning model for sentiment analysis of social media text. They used Deep Neural Network (DNN), Convolutional Neural Network (CNN) Long-Short Term Memory (LSTM) architecture, and Logistic Regression classifier as a baseline. They chose 2 datasets for their study one is IMDB Movie Reviews and the other is Twitter Sentiment 140.

For Feature Extraction, they follow Bag of Words (BOW), N-grams, TF-IDF, word embedding

1. Pre-processing and word embeddings:

For the text-preprocessing task, they used the NLKT library and this task includes removing punctuations, markup tags, HTML and Tweet addresses, removing stopwords, and converting all words into lowercase words.

1. DNN models design and hyperparameters settings:
2. Used pre-trained GloVe embeddings of size 100 in the first layer (embedding layer)
3. First model CNN with three convolutional layers with different kernel sizes and used Maxpooling layers between them and then flatten and Dense layer.
4. In the second approach they combine the CNN+LSTM
5. Third, CNN + BiLSTM (forward and Backward LSTM)
6. To obstruct overfitting Dropout layers are used
7. Evaluation:

IMDB Dataset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Models | LR | CNN | CNN-LSTM | CNN-BiLSTM |
| Precision | 86.62% | 90.04% | 90.90% | 83.08% |
| Recall | 85.54% | 90.31% | 84.84% | 93.25% |
| F1-Score | 86.08% | 90.18% | 87.76% | 87.87% |
| Accuracy | 85.90% | 90.09% | 88.08% | 87.03% |

Twitter-140 dataset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Models | LR | CNN | CNN-LSTM | CNN-BiLSTM |
| Precision | 71.61% | 76.17% | 78.98% | 79.54% |
| Recall | 74.63% | 79.47% | 77.47% | 84.41% |
| F1-Score | 73.09% | 77.78% | 78.23% | 81.91% |
| Accuracy | 79.54% | 77.24% | 78.37% | 82.10% |

The experiment showed that LR (baseline) achieved 85.9% (74.23%), CNN achieved 90.09% (77.24%), CNN-LSTM reached 88.01% (78.36%), and BiLSTM-CNN attained 87.03% (82.10%).

1. **Machine Learning Algorithms:**

In “Sentiment Analysis of Twitter Data: A Survey of Techniques” [3] paper authors Vishal and S. Sonawane use machine learning algorithms on a comparatively small dataset publicly made available by Stanford University. They studied Naïve Bayes, Max Entropy, and Support Vector Machine algorithms. Also, they compared the result with the various data pre-processing techniques like stopwords removal, Unigram, and Trigram.

A. Data Processing:

StopWords: Words like I, am, you, your, etc. are removed during the data processing step because these words don’t add much information to the text.

Bigram: It uses a combination of two words, eg “Not happy” clearly indicating the negative sentiment.

B. Evaluation:

1. They achieve 73.56% accuracy without using StopWords removal, Unigram, and Bigram techniques.
2. Naïve Bayes Algorithm

|  |  |
| --- | --- |
| Algorithm | Accuracy |
| Naïve Bayes (Unigram) | 74.56% |
| Naïve Bayes (Bigram) | 76.44% |
| Naïve Bayes (trigram) | 75.41% |

1. Support Vector Machine (SVM)

|  |  |
| --- | --- |
| Algorithm | Accuracy |
| SVM with unigram | 76.68% |
| SVM with bigram | 77.73% |

The paper concludes that Support Vector Machine and Naïve Bayes these machine learning algorithms that give accuracy.

1. SYSTEM ARCHITECUTRE

Dataset:

We used the Sentiment140[4] dataset for our experiment. The dataset contains 1.6 million tweets. The dataset has a target, ids, date, flag, user, and text columns.

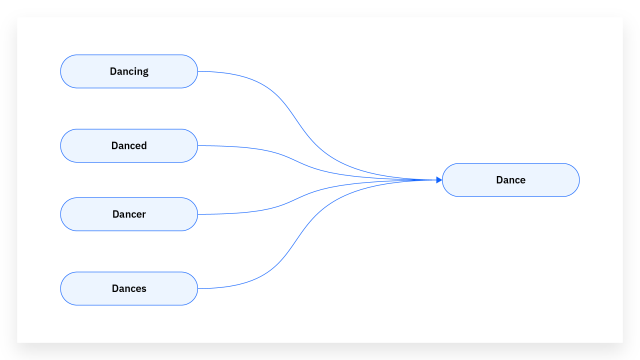
1. Target: the polarity of tweets (0: Negative, 4: Positive)
2. Ids: id of the tweet
3. Date: date of the tweet
4. Flag: query
5. User: The user who tweeted
6. Text: text of the tweet.

The dataset has **0.8M** **positive** tweets and **0.8M negative** tweets.

DATA PROCESSING

This dataset has no column names so we manually give names to each column for our better understanding to make the process easy to understand the data. The column names are target, ids, date, flag, user, and text. We then check if any column contains any null values or not and find that there are no null values present in any of the columns. The dataset has an equal number of Positive and negative Tweets i.e. 0.8M tweets in each category. For the target column, 0 represents Negative sentiment and 4 represents Positive sentiment. We replaced 4 with 1 as it gives more context to the data. We apply the following data pre-processing and data cleaning operations on data.

1. StopWords remove: Use the NLTK library to remove all the stopwords from the data as these words don’t contribute much to prediction.
2. Removal of Special characters and converting each word to a lowercase word: to remove any special characters such as @, // etc. from the data we use a regular expression library to find these characters and remove them. We convert all words into lowercase words.
3. Stemming: Stemming is a text-preprocessing technique in natural language processing (NLP). Specifically, it is the process of reducing the inflected form of a word to one so-called “stem,” or root form, also known as a “lemma” in linguistics.[4]



We used the PortStemmer function from the NLTK library to perform the stemming operation. Then we add the “stemmed\_text” column to our dataset after performing the stemming step.

(The above operation is performed on “text” columns)

FEATURE EXTRACTION

After data processing, we extract only the target and the newly added stemmed\_text column for further processing. All other columns are not required for sentiment analysis. After this, the next step is to split the data into train test split. For this, we used scikit-learn train\_test\_split functions with the splitting ratio of 80:20 i.e. 80% data for training purposes and 20% data for testing purposes. To convert the text into vectors we used the TF-iDF technique.

TF-IDF: Term Frequency Inverse Document Frequency (TF-IDF) is an algorithm to transfer text into a meaningful representation of numbers (vectors)

1. Term Frequency (TF):

To measure how frequently a word (term) appears in text.

Formula:

1. Inverse Document Frequency (IDF):

It measures how important a word is with entire text corpus.

Formula:

1. TF-IDF Score:

Model Training and Result:

Machine Learning Algorithms:

1. Logistic Regression

Logistic Regression is a supervised machine learning algorithm used for classification problems. It is a statistical algorithm. For predicting the output, it uses Sigmoid functions, which take inputs and produce probability values between 0 and 1.

Sigmoid Function:

1. NAÏVE BAYES

Naïve Bayes is a classification algorithm based on Baye’s theorem. The name “Naïve” indicates the assumptions made by the algorithm. The algorithm assumes that features(columns) are independent, given class labels.

Bayes Theorem:

Where, y=Class Labels and X= dependent features(columns)

We achieve 80% training accuracy and 75.5% testing accuracy by using Multinomial Naïve Bayes.

1. XGBOOST CLASSIFIER

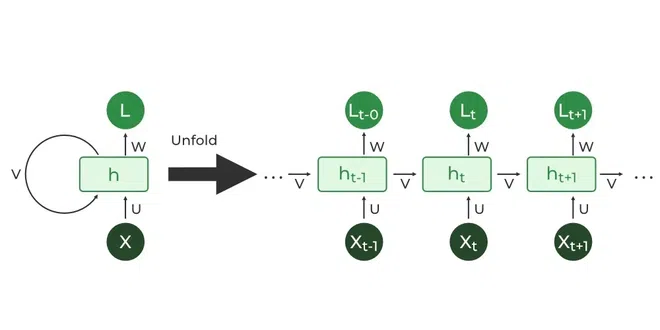
Extreme Gradient Boosting (XGBoost) is an ensemble learning method that combines predictions of multiple weak models to produce a stronger prediction (weaker → stronger). It is widely used because it can handle a large dataset and handling of missing values without requiring significant pre-processing. It is an implementation of the Gradient Boosted Decision Tree.

Deep Learning Architecture:

1. RNN ARCHITECTURE

RNNs are used for tasks that involve sequential data, such as time series prediction, natural language processing (NLP), and speech recognition.

RNNs are like networks that have a memory. They process data one step at a time and remember information from previous steps. Imagine reading a sentence word by word. An RNN processes each word in order and keeps track of the context from previous words to understand the sentence better. This memory aspect helps RNNs make decisions based on the sequence of data, such as predicting the next word in a sentence or recognizing spoken words over time. RNNs are very similar to feedforward neural networks, except is also have a connection pointing backward.

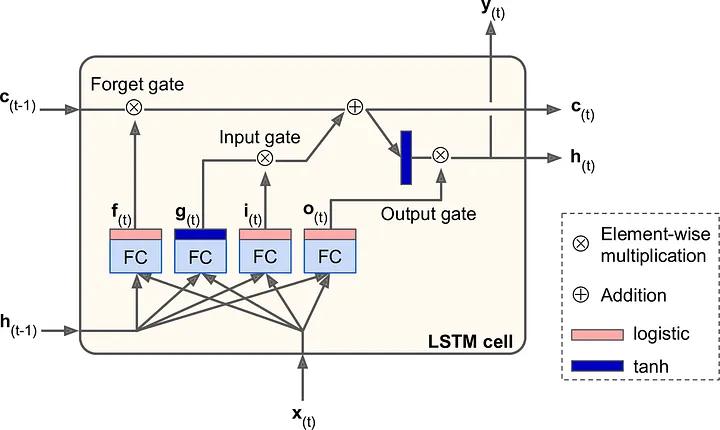


Because of unstable gradients, RNN suffers from two major problems: 1) Problem of long-term dependency and(Vanishing gradient problem) 2) Stagnated Training(Exploding gradient problem).

As sequence length increases, RNN struggles to remember the initial time-step context and this issue arises because of the vanishing gradient problem, and because of the exploding gradient the unstable gradient problem occurs.

1. LSTM AND BILSTM ARCHITECTURE

To tackle the Long-Term dependency problem Sepp Hochreiter and Jurgen Schmihuber introduced “Long-Short-Term Memory (LSTM)” [5] architecture.



From c(t-1) to c(t) the LSTM cell decides which part has to remove (forget) based on input(x(t)) and what to add based on input (x(t))

LSTM Cell has two states:

1. c(t) (Cell state) → Long Term State, for remembering information for a longer duration
2. h(t) (Hidden state) → Short Term State, for remembering information for short durations

The key idea behind these two states is what to keep and what to discard.

Type of gate in LSTM:

1. Forget gate (f(t)): it controls which parts of the long-term state should be erased.
2. Input gate (i(t)): it controls which parts of g(yt) should be added to long-term state.
3. Output gate(o(t)): it controls which part of the long-term state should be read and output at this time step, both to h(t) and y(t).

LSTM computations:

Here,

wxi, wxf, wxo, and wxg are the weight matrices of each of the four layers for their connections to the input vector x(t).

whi, whf ,who, and whg are the weight matrices of each of the four layers for their connection to the previous short-term state h(t-1).

bi, bf, bo, and bg are the biases for each of the four layers.

BiLSTM

To improve the performance of LSTM, BiLSTM architecture is introduced in which it captures the dependencies in both forward and backward directions of sequence. It uses two LSTM layers:

1. Forward LSTM: It processed the sequence from left to right (Start to End) direction.
2. Backward LSTM: It processes the sequence from right to left (end to start)

Then, output from both LSTMs is combined, which allows the model to consider the context from both directions at each time step.

1. GRU ARCHITECTURE

The Gated Recurrent Unit (GRU) cell was proposed by Kyunghyun Cho et al. in a 2014 paper titled “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation” [6]

GRU is a simplified version of the LSTM cell, and it seems to perform just as well. The simplifications are:

1. Both state vectors are merged into a single vector h(t).
2. A single gate controller x(t) controls the forget gate and input gate.
3. Full state vector h(t) is the output of every time step h(t)=y(t)
4. New gate controller r(t) that controls which part of the previous state will be shown to the layer g(t).

****

TRAINING AND RESULT

Machine Learning Algorithms

We used Logistic Regression (LR) as a baseline model to compare the accuracy of the other models.

|  |  |  |  |
| --- | --- | --- | --- |
| Models | LR | MNB | XGBoost |
| Precision | 75.62% | 75.28% | 75.62% |
| Recall | 78.96% | 75.12% | 78.96% |
| F1-Score | 77.25% | 75.2% | 77.25% |
| Accuracy | 76.75% | 75.23% | 76.75% |

Deep Learning Algorithms

For each architecture, we used the Adam optimizer with a learning rate of 0.001, the sigmoid activation function for the output (last) layer, binary cross-entropy as the loss function, accuracy as the metric, 50 epochs, and a batch size of 128.

For the LSTM, BiLSTM, and GRU architectures, we used the ReLU activation function in the Dense layer. To avoid overfitting, we applied L2 regularization.

Model Summary

|  |  |  |
| --- | --- | --- |
| Models | Layers | Parameters |
| Simple RNN | Embedding | input\_dim=5000, output\_dim=100, input\_length=50 |
| simpleRNN | 128, return\_sequence=True |
| Dropout | 0.5 |
| simpleRNN | 64, return\_sequence=True |
| Dropout | 0.5 |
| simpleRNN | 32 |
| dense | 1, activation=” Sigmoid” |
| LSTM | Embedding | input\_dim=5000, output\_dim=100, input\_length=50 |
| LSTM | 128, return\_sequence=True  kernel\_regularizer=l2(0.001) |
| Dropout | 0.4 |
| LSTM | 64 |
| Dropout | 0.4 |
| Dense | 64, activation='relu', kernel\_regularizer=l2(0.001) |
| dropout | 0.4 |
| dense | 1, activation=’sigmoid’ |
| BiLSTM | Embedding | input\_dim=5000, output\_dim=100, input\_length=50 |
| Bidirectional (LSTM) | 128, return\_sequence=True  kernel\_regularizer=l2(0.001) |
| Dropout | 0.4 |
| Bidirectional (LSTM) | 64 |
| Dropout | 0.4 |
| Dense | 64, activation='relu', kernel\_regularizer=l2(0.001) |
| dropout | 0.4 |
| dense | 1, activation=’sigmoid’ |
| GRU | Embedding | input\_dim=5000, output\_dim=100, input\_length=50 |
| Bidirectional (LSTM) | 128, return\_sequence=True  kernel\_regularizer=l2(0.001) |
| Dropout | 0.4 |
| Bidirectional (LSTM) | 64 |
| Dropout | 0.4 |
| Dense | 64, activation='relu', kernel\_regularizer=l2(0.001) |
| dropout | 0.4 |
| dense | 1, activation=’sigmoid’ |

Result

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Models | RNN | LSTM | BiLSTM | GRU |
| Precision | 76.47% | 77.41% | 76.52% | 77.84% |
| Recall | 75.75% | **78.27%** | **77.96%** | **78.01%** |
| F1-Score | **77.6%** | 77.84% | 77.55% | 77.92% |
| Accuracy | 77.19% | 77.64% | 77.5% | 77.83 |

**CONCLUSION**

In this paper, we apply both machine learning and deep learning techniques to the sentiment140 dataset. This large dataset (1.6M tweets) is best for comparing the ML and DL algorithms. We found that the machine learning algorithm achieved a maximum of 76.75% accuracy whereas with only 50 epochs we crossed 77% accuracy with a simple RNN model with a limited number of parameters, and with LSTM we crossed 82% training accuracy. Hence, we can conclude that deep learning architecture outperforms machine learning techniques. As with limited resources we are not able to do much experiment. For future studies, we plan to increase the epoch size and will also increase the LSTM and GRU layers and compare the results.

**REFERENCES**

[1]Sheresh Zahoor, Rajesh Rohilla “Twitter Sentiment Analysis Using Lexical or Rule-Based Approach: A Case Study”.

[2] Vasily D. Derbentsev, Vitalii S. Bezkorovainyi et al. “A comparative study of deep learning models for sentiment analysis of social media texts.”

[3] Vishal A. Kharde, S.S. Sonawane “Sentiment Analysis of Twitter Data: A Survey of Techniques.”

[4] <https://www.ibm.com/topics/stemming>

[5] Sepp Hochreiter and Jurgen Schmihuber “Long-Short-Term Memory.”

[6] Kyunghyun Cho et al. “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”

[7] Mayur Wankhade, Annavarapu Chandra Sekhara Rao & Chaitanya Kulkarni “A survey on sentiment analysis methods, applications, and challenges”

[8] Dr. Manjula Bairam, R Lakshman Naik “A Study of Sentiment Analysis: Concepts, Techniques, and Challenges”

[9] Xing Fang & Justin Zhan “Sentiment analysis using product review data”