Exploration of Natural Language Processing for Sentiment Analysis

1 Introduction

With the rapid rise in the spread of information through social media platforms such as Twitter, the tweets have earned significant attention as a rich source of information to guide important decision-making processes. However, their inherent nature of informal and noisy linguistic style poses a challenge to NLP tasks, including sentiment analysis. This work focuses on utilizing traditional machine learning, deep learning, and transformer-based architectures along with various word embedding techniques to build a robust twitter sentiment analysis model using the Sentiment140 dataset.

2 Literature Survey

Sentiment analysis is an automated process used to predict people's opinions, sentiments and emotion towards entities and their attributes expressed in written text. The main task in this domain is to build a deep understanding of the natural text to deal effectively with the informal structure of the tweets. The Vector space models (VSMs) are one of the earliest and the most common strategies adopted in text classification literature. These models represent each document in a corpus as a point in a vector space and points that are close together are classified as semantically similar documents. The initial VSMs rlied on count-based methods such as Bag-of-words(BoW) and BoW with TF-IDF. Although these models have performed well, they cannot deal with the curse of dimensionality. Learning dense vectors to represent words and texts, the embeddings, have been used to tackle the problem of growing dimensionality. The Word2Vec, Fast-Text, and Glove embeddings have been extensively used but the performance of such techniques are still poor due to their inability to capture contextual meaning of tweets in a limited space. Contextual embedding techniques including transformer based autoencoder methods such as BERT, RoBERTa, and BERTweet capture not only complex characteristics of word usage, including syntax and semantics, but also the variance of word uage across linguistic contexts. These methods have achieved state-of-the-art results on various NLP tasks, including sentiment analysis.

The authors in [1] introduce a method for automatically classifying the sentiment of the twitter messages as positive or negative using distant supervision. The tweets are labeled with emoticons as noisy labels eliminating the need for hand-labeled data. This approach allows for large-scale training leveraging the Twitter API. The study explores Naive Bayes, Maximum Entropy, and SVM ML algorithms with different extraction techniques such as unigrams, bigrams, and part-of-speech (POS) tags. The classifiers achieved an accuracy of over 80% on test set of tweets with unigrams providing the most effective feature set, and combination of unigram and bigram resulting in best performance. The limitations of the paper include a study that only focused on binary classification, data sparsity due to bigram features, and POS tagging leading to reduced performance, and missing emoticon handling which can provide important sentiment of a tweet.

The sentiment analysis using Sentiment140 dataset was carried out in [2] with the help of Very-Deep Convolutional Neural Networks(VD-CNN) and BERT models. The architectures for the analysis included a baseline CNN with four parallel 1-d convolutional layers and max-pooling, a VD-CNN with Glove embeddings (trained on Wikipedia + Gigaword and Twitter) comprising of two and four convolutional blocks, and a pretrained BERT model. The two convolutional block CNN using Glove Twitter embeddings achieved the best performance with and F1 score of 85.3% and an accuracy of 80.7%. The BERT model performed similarly with an accuracy of 80.8% but was outperformed by CNN in terms of F1 score due to limited fine-tuning. The deeper CNN model did not improve performance over the shallower model due to short length of tweets. The models showed a tendency to overfit the training data, the deeper models required more training time and memory but did not show substantial performance gain and limiting the tweets to a fixed size may have led to misclassification of longer tweets.

The authors in [3] conduct a comprehensive evaluation of various text representation models for sentiment analysis across 22 diverse Twitter datasets. The paper provides a comparison of traditional representations such as Bag-of-Words, TF-IDF, static word embeddings such as Word2Vec, FastText, Glove, Emov2Vec, DeepMoji, Sentiment-Specific Word Embeddings (SSWE), contextual embeddings such as BERT, RoBERTa, BERTweet with classifiers including SVM, Logistic Regression, Random Forest, XGBoost, and MLP. The RoBERTa with SVM and MLP classifiers performed the best overall in terms of accuracy and F1-macro. The Emo2Vec and w2v-Edin(a variant of Word2Vec trained on tweets) also achieved strong performance, especially when paired with Random Forest and XGBoost. Fine-tuning models like BERT and RoBERTa improved their sentiment classification performance. The transformer-based models require significant computational resources and memory, the static embedding like Word2Vec and FastText underperformed compared to the contextual embeddings, fine-tuning contextual models improved results, but the performance gains did not always justify the additional computational costs.

The multi-class sentiment classification task was carried out by authors in [4]. CNN and LSTM models were trained with Word2Vec, FastText and Glove embeddings, and an ensemble was proposed combining results of CNN and LSTM models to boost performance. The ensemble model yielded a 74.8% score on a test set containing 1853 tweets. The system was not able to perform well particularly in handling neutral tweets, and given high computational complexity of combining CNN and LSTM models.

The authors in [5] proposed a novel architecture that combines hidden layers from BERT using Gated Recurrent Units (GRUs) for twitter sentiment analysis. The GRUBERT dynamically combines hidden layers capturing different linguistic aspects, syntactic information in middle layers, semantic features in later layers. After combining the hidden layers, the embeddings are passed through a downstream GRU, a fully connected layer with ReLU, dropout, and finally a softmax classifier for sentiment classification. The model achieved 90.94% accuracy using an ensemble of BERT and RoBERTa embeddings, and outperformed GloVe, ELMo, and standard BERT-based approaches by dynamically learning which layers to use for different tweets. The training of the model is computationally expensive and while powerful, the complexity of using multiple GRUs for layer fusion can limit scalability on larger datasets.

A multi-view ensemble approach for sentiment analysis on twitter data was proposed in [6]. The architecture comprised of soft-voting ensemble where the classifiers (SVM and Logistic Regressor) are trained separately on text representations including bag-of-words model, averaged word embeddings, and TF-IDF weighted word embeddings. The model achieved an accuracy of 61.7% on the Twitter2017 test dataset. The model struggles with datasets from different origins, and sarcasm detection due to the lack of sequence modeling. A decision stream algorithm was proposed in [7] that merges nodes in decision trees based on statistical similarity, creating a deep directed acyclic graph (DAG) instead of a traditional tree structure leading to reduced model complexity and overfitting. The Decision Stream model achieved better results than standard decision trees on twitter sentiment analysis, leading to 35% reduction in prediction error. One drawback of the model is that the merging process can increase computational time, especially in large datasets. A multitask learning framework using a bidirectional LSTM (biLSTM) to jointly learn ternary (3-class) and fine-grained (5-class) sentiment classification tasks on twitter data was proposed in [8]. The multitask learning model achieved a mean absolute error score of 0.685 using the multitask biLSTM approach, outperforming the single-task biLSTM model, achieving 0.694 MAE. The proposed model requires large computational resources due to the neural network architecture, and the performance of the model can depend heavily on the availability of multitask data.

3 Proposed Model

This work includes classifier models including Bernoulli Naive Bayes, SVM, Logistic Regressor, and Random Forest with TF-IDF word embeddings for twitter sentiment analysis. Custom CNN and LSTM models with different word embeddings such as Glove (trained on Wikipedia + Gigaword and Twitter) and

FastText are trained on the Sentiment140 dataset. Finally, transformer-based autoencoder models including BERT, RoBERTa, BERTweet, and DistilBERT are fine-tuned for twitter sentiment analysis.

3.1 Dataset Description and EDA

The Sentiment 140 dataset is a collection of 1.6 million tweets labeled for sentiment analysis. Each tweet is automatically labeled as positive(labeled as 4) or negative(labeled as 0) based on the emoticons present in the text. The dataset has six features including tweet ID, the sentiment label, date of the tweet, the query (if present), username, and the text of the tweet. The sentiment label distribution and the distribution of the tweet length can be seen in the figure 1 and 2.

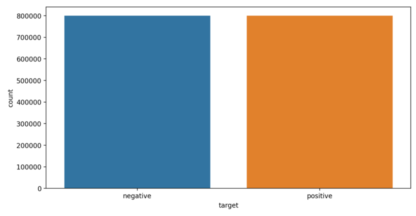

Figure 1: Sentiment Label Distribution

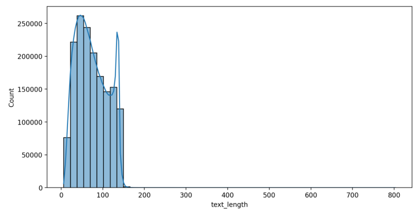

Figure 2: Distribution of Tweet Lengths

3.2 Preprocessing Data

Firstly, the positive label of the tweets are replaced with 1 instead of 4 to ensure proper binary classification structure. The unnecessary features apart from the text and the sentiment of the tweets have been stripped as they do not add useful information for sentiment analysis. Various experimentation including effect of stop-word removal and checking model performance with stemming compared to lemmatization was carried out to preprocess data effectively.

3.2.1 Tokenization

The first step in the cleaning of the data includes breaking the entire sentence into smaller words or tokens that can be further processed in order to remove the casual semantic language that introduces noise in the text.

3.2.2 Converting to Lowercase

This step converts all text to lowercase to ensure uniformity. It avoids treating ”Happy” and ”happy” as different words, improving model consistency.

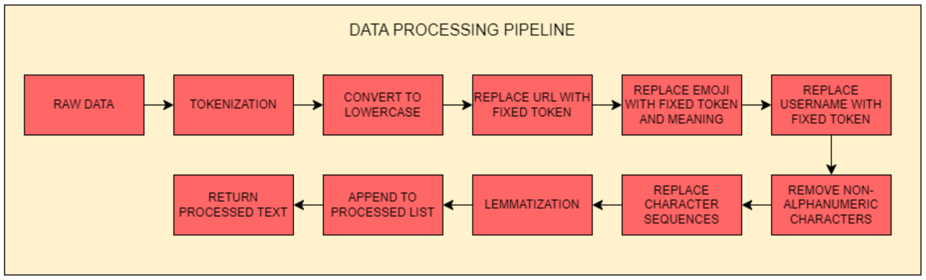
DATA PROCESSING PIPELINE
RAW DATA
TOKENIZATION
CONVERT TO LOWERCASE
REPLACE URL WITH FIXED TOKEN
REPLACE EMOJI WITH FIXED TOKEN AND MEANING
REPLACE USERNAME WITH FIXED TOKEN
REMOVE NON-ALPHANUMERIC CHARACTERS
REPLACE CHARACTER SEQUENCES
LEMMATIZATION
APPEND TO PROCESSED LIST
RETURN PROCESSED TEXT

Figure 3: The preprocessing pipeline for cleaning data

3.2.3 Replace URLs

The URL links are replaced with a placeholder [URL] to standardize web links without needing to analyze the specific content of the links.

3.2.4 Replace Emojis

This step converts emojis into their textual meanings (e.g., ":)" $\rightarrow$ "smile") to retain sentiment information that emojis convey, which may be important for sentiment analysis. Although, this can be useful for the models that are not trained on twitter dataset specifically, it can also reduce efficiency compared to the word embeddings like DeepMoji and transformer models like BERTweet that have been trained on the twitter dataset and have mastered the technique of representing emojis.

3.2.5 Replace Usernames

The @Usernames are replaced with a placeholder [USER] to anonymize and generalize mentions, as usernames are not relevant for sentiment analysis.

3.2.6 Remove Non-Alphanumeric Characters

This step removes punctuation and other special characters to reduce noise, as they generally do not contribute to sentiment and can complicate text processing.

3.2.7 Replace Character Sequences

Replacing the character sequences condenses repeated characters (eg. "sooo good" $\rightarrow$ "so good") to a normalized form, reducing redundancy and improving text uniformity.

3.2.8 Lemmatization

This is a process that reduces words to their base form (eg. "running" $\rightarrow$ "run") to ensure that different forms of a word are treated as the same, aiding in better feature extraction for analysis.

Finally, the tokens are appended together and returned as a sequence of cleaned and processed text free of the informal and noisy input inherent to a tweet. The entire data processing pipeline is shown in figure 3.

3.3 Proposed Model Description

3.3.1 Machine Learning Models with TF-IDF Vectorizer

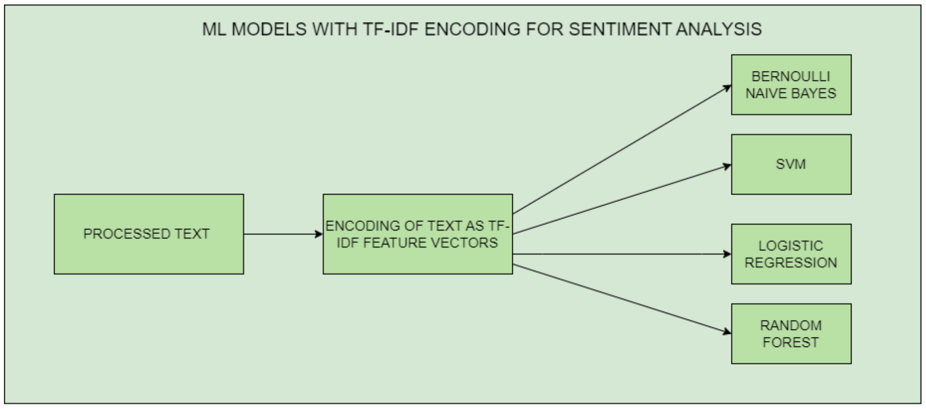
ML MODELS WITH TF-IDF ENCODING FOR SENTIMENT ANALYSIS
PROCESSED TEXT
ENCODING OF TEXT AS TF-IDF FEATURE VECTORS
BERNOULLI NAIVE BAYES
SVM
LOGISTIC REGRESSION
RANDOM FOREST

Figure 4: ML models with TF-IDF vectoriser for twitter sentiment analysis

The TF-IDF (Term Frequency-Inverse Document Frequency) is used to evaluate the importance of a word in a document relative to a collection of documents (corpus). The term frequency measures the number of times a term appears in a document and the inverse document frequency measures the importance of a term by considering its frequency in various documents. The TF-IDF vectorizer converts a collection of raw text into a matrix of TF-IDF features. Feature extraction using both unigrams and bigrams has been considered in this work.

The Bernoulli Naive Bayes classifier is a probabilistic model based on Bayes theorem working on the assumption that each feature is independent of the others, and the features follow a Bernoulli distribution. The posterior probability of a class given a document is given by:

$$P(c|D) = \frac{P(D|c) \cdot P(c)}{P(D)} \tag{1}$$

The Support Vector Classifier (SVC) is a type of Support Vector Machine (SVM) that works by finding the optimal hyperplane that separates classes in a high-dimensional space. The classifier maximizes the margin between the closest points of the two classes (support vectors) for sentiment analysis. The classifier can use different kernel functions to handle non-linear data by transforming it into a higher-dimensional space.

The Logistic Regression is a statistical model that uses the logistic function to model the probability of the sentiment based on the encoded TF-IDF features. The probability can be modeled as:

$$P(y = 1 \mid X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n)}} \tag{2}$$

The Random Forest Classifier is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of their predictions for classification.

3.3.2 Deep Learning Models with static embeddings

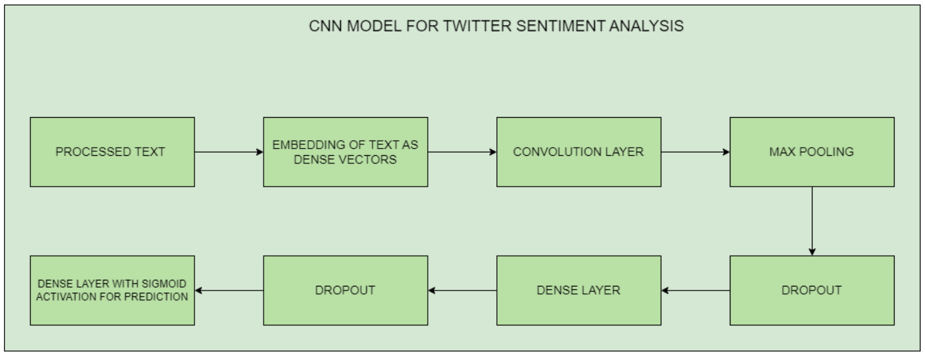
CNN MODEL FOR TWITTER SENTIMENT ANALYSIS
PROCESSED TEXT
EMBEDDING OF TEXT AS DENSE VECTORS
CONVOLUTION LAYER
MAX POOLING
DROPOUT
DENSE LAYER
DROPOUT
DENSE LAYER WITH SIGMOID ACTIVATION FOR PREDICTION

Figure 5: CNN model for twitter sentiment analysis

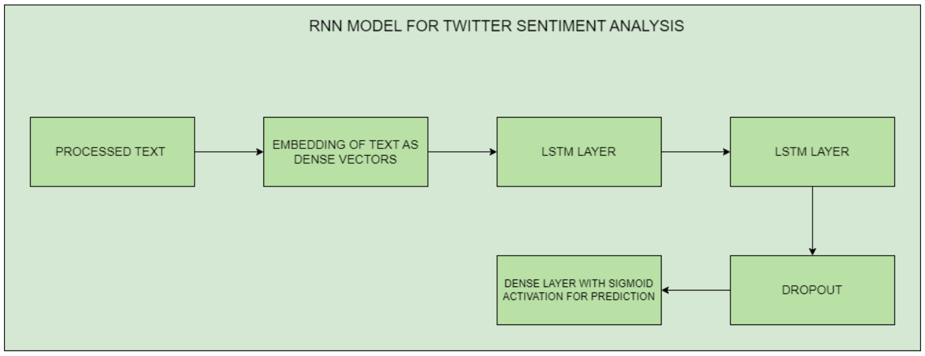
RNN MODEL FOR TWITTER SENTIMENT ANALYSIS
PROCESSED TEXT
EMBEDDING OF TEXT AS DENSE VECTORS
LSTM LAYER
LSTM LAYER
DROPOUT
DENSE LAYER WITH SIGMOID ACTIVATION FOR PREDICTION

Figure 6: RNN model for twitter sentiment analysis

Word-Index Embedding : The processed text are converted into sequences of integers, where each integer represents a word in the vocabulary. The embedding layer transforms the integer-encoded words from the tokenization process into dense vector representations. Each unique word in the vocabulary is mapped to a fixed-size vector. These embedding capture semantic relationships between words and are trained to the specific patterns and relationships relevant to the sentiment classification task.

Glove Embedding: The Glove embedding uses word co-occurrence statistics from a corpus to capture semantic relationships between words. It learns word vectors by factorizing a matrix that represents how frequently words co-occur in a given context. Glove provides pre-trained embeddings on large corpora like Wikipedia with vectors that can capture both syntactic and semantic information.

Glove Twitter Embedding: The GloVe Twitter embeddings are a specialized version of GloVe embeddings trained specifically on a large corpus of Twitter data. The pre-trained embeddings on a massive corpus of tweets help in understanding the specific language patterns and jargon used on Twitter, making them particularly useful for Twitter-based sentiment analysis.

FastText Embedding: The FastText embedding breaks down each word into subword units that allows to handle out-of-vocabulary words or rare words more effectively by constructing the word vector from its subword components. It captures both morphological information and context, making it particularly robust for handling typos, rare words, or word variations.

CNN Model: The CNN model uses convolutional filters on the embedding layer to detect local patterns and relationships in text data and the spatial hierarchies in the CNN model help capture not only the meanings of individual words but also the context provided by neighboring words. The visualization of the custom CNN model used for sentiment analysis is provided in figure 5.

RNN Model: The RNN model is designed to process sequences of data by maintaining a hidden state that captures information about the previous inputs in the sequence making them particularly effective for tasks where the order of inputs matters. Each word embedding in the sequence is used to update the hidden state and carry forward the information of the past inputs. After processing all words in the sequence, the final hidden state is used to make the sentiment prediction. A fully connected layer with a sigmoid activation function produces the final output. The visualization of the custom CNN model used for sentiment analysis is provided in figure 6.

3.3.3 Transformer-based autoencoder models

BERT: BERT(Bidirectional Encoder Representations from Transformers) [12] utilizes WordPiece tokenization to convert text into subwords, adding [CLS] at the start and [SEP] to separate segments. The input sequence is padded or truncated to maintain a uniform length. The architecture of BERT consists of 12 transformer layers, allowing it to capture deep contextual relationships within the text. The self-attention mechanism enables the model to weigh the importance of different words in context, leading to rich embeddings for each token. The output embedding from the [CLS] token is fed into a classification layer that computes the final logits corresponding to different classes.

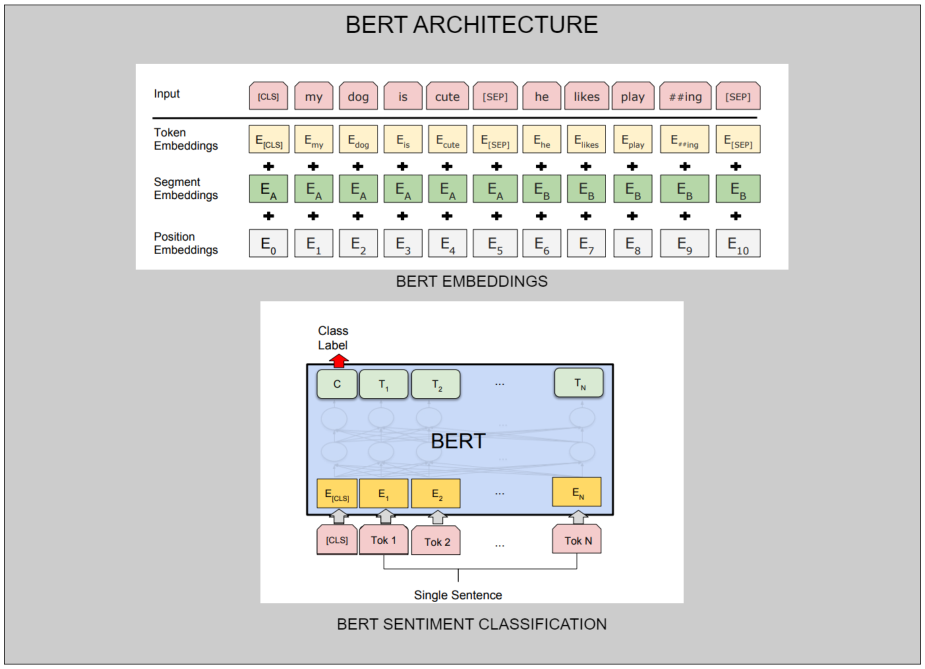

Figure 7: BERT model for twitter sentiment analysis [12]

DistilBERT: DistilBERT [10] is a distilled version of BERT that retains 97% of its language understanding while being 60% faster and smaller. It employs WordPiece tokenization, which breaks text into subword units to effectively manage out-of-vocabulary (OOV) words and also adds special tokens such as [CLS] at the beginning of the input for classification tasks and [SEP] to separate sentences or segments within the input. To maintain uniform input lengths, the tokenizer handles padding by appending [PAD] tokens to shorter sequences and truncating longer sequences to a specified maximum length. The architecture of DistilBERT features 6 transformer layers, utilizing self-attention mechanisms to capture contextual relationships in the input text. The model generates contextual embeddings, with particular emphasis on the [CLS] token for classification tasks.

RoBERTa: RoBERTa (A Robustly Optimized BERT Pretraining Approach) [11] uses Byte-Pair Encoding (BPE) for tokenization, which allows it to handle a wide range of linguistic phenomena effectively. This method enables the tokenizer to create tokens for common phrases and constructs that may not be present as single words. RoBERTa includes special tokens including [CLS] for classification and [SEP] to separate text segments. The padding and truncation mechanisms are similar to those in BERT, ensuring that input sequences meet a specified maximum length. The architecture of RoBERTa features 12 transformer layers but also implements dynamic masking during training, meaning that different words are masked in each training epoch. This technique increases the diversity of training data, and helps to improve the model's understanding of the context. RoBERTa eliminates the next sentence prediction objective found in BERT, leading to better contextual embeddings primarily derived from the [CLS] token.

BERTweet: BERTweet [9] is a model specifically designed for processing Twitter data, incorporating a specialized WordPiece tokenizer tailored to handle unique social media elements, such as hashtags, mentions, and emojis. The BERTweet model adds special tokens including [CLS] for classification tasks and [SEP] to separate different segments. Padding and truncation are applied to maintain consistent input lengths across different sequences. The BERTweet architecture is based on BERT, featuring 12 transformer layers. It captures the nuances of social media language and context, producing contextual embeddings primarily from the [CLS] token.

4 Experimentation and Results

The experiments were carried out using T4 and P100 GPUs. The results of the built ML, DL, and transformer models for the task of twitter sentiment analysis is shown in table 1. The BERTweet model achieved highest KPI metrics due to the specialized training of its embeddings and model on the Twitter corpus. The simpler ML and DL models have also achieved comparative results indicating the robustness of the models for effective twitter sentiment analysis.

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Bernoulli Naive Bayes(TF-IDF)	80.18%	79.67%	81.07%	80.36%
SVM(TF-IDF)	81.84%	81.30%	82.72%	82.01%
Logistic Regression(TF-IDF)	82.56%	81.94%	83.55%	82.74%
Random Forest(TF-IDF)	72.36%	70.63%	76.59%	73.49%
CNN(word index embedding)	81.92%	83.36%	79.91%	81.6%
CNN(Glove embedding)	79.04%	80.81%	76.35%	78.51%
CNN(Glove Twitter embedding)	81.3%	82.95%	78.94%	80.9%
CNN(FastText embeddings)	80.89%	82.62%	78.4%	80.45%
RNN(word index embedding)	82.47%	82.5%	82.56%	82.53%
RNN(Glove embedding)	82.58%	82.34%	83.1%	82.72%
RNN(Glove Twitter embedding)	83.45%	83.36%	83.71%	83.54%
RNN(FastText embeddings)	83.53%	84.44%	82.35%	83.38%
BERT(Fine-Tuned)	85.8%	86.92%	84.33%	85.61%
RoBERTa(Fine-Tuned)	86.26%	88.59%	83.28%	85.85%
DistilBERT(Fine-Tuned)	85.32%	86.98%	84.14%	85.54%
BERTweet(Fine-Tuned)	**87.4%**	**88.35%**	**86.19%**	**87.25%**

Table 1: Comparison of built models for twitter sentiment analysis

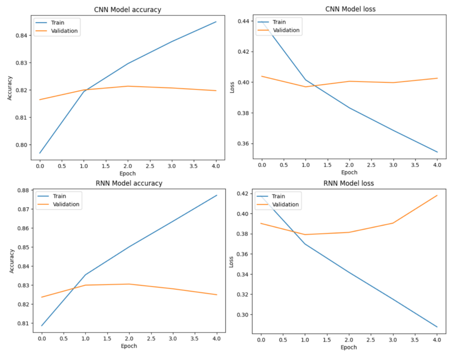

Figure 8: Training graphs for CNN & RNN Models with word-index embeddings

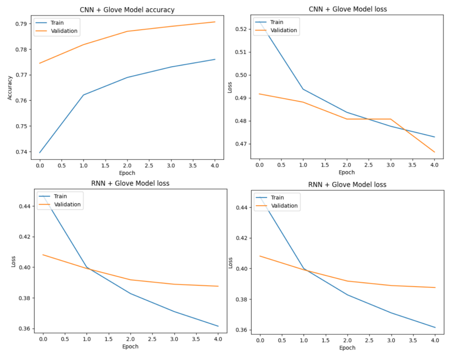

Figure 9: Training graphs for CNN & RNN Models with Glove embeddings

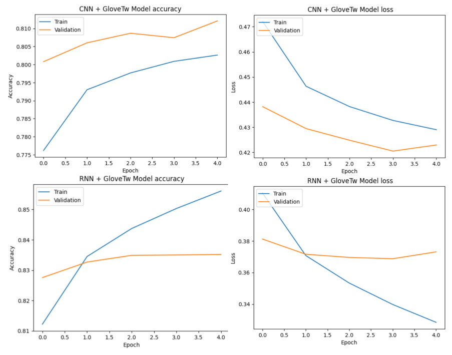

Figure 10: Training graphs for CNN & RNN Models with Glove Twitter embeddings

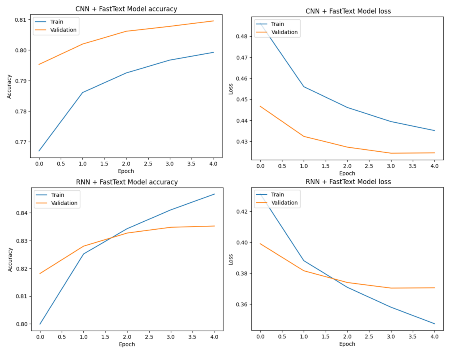

Figure 11: Training graphs for CNN & RNN Models with FastText embeddings

The description of the training hyperparameters for the DL and transformer based models is provided in the table 3 and 4 respectively. Also, the details regarding the parameters and the architectures of the transformer models is provided in table 2. The training graphs of DL models with various embedding techniques is shown in figures 8, 9, 10, and 11.

Model	**No.of Layers**	**Hidden Size**	**No. of Attention Heads**	**Total Parameters**
BERT	12 (Base)	768 (Base)	12 (Base)	109483009
DistilBERT	6	768	12	66363649
RoBERTa	12 (Base)	768 (Base)	12 (Base)	124646401
BERTweet	12	768	12	134900737

Table 2: Summary of Transformer Models and Their Parameters

Hyperparameter	**Value**
Epochs	5
Batch Size	64
Optimizer	Adam
Loss	BinaryCrossentropy

Table 3: Training Hyperparameters for CNN and RNN Models

Hyperparameter	**Value**
Epochs	3
Batch Size	64
Learning Rate	1e-5
Optimizer	Adam
Loss	BinaryCrossentropy

Table 4: Training Hyperparameters for Transformer Models

5 Conclusion and Future Work

To conclude, this work explores various NLP techniques and models to build a robust twitter sentiment analysis model. The transformer models provided the best KPI metrics given the constrained computational power available for this work. The future work includes:
1. Comparing the model performance without aggressive preprocessing step carried out in this work. Efficient embedding techniques such as DeepMoji and models such as BERTweet are trained on a huge Twitter corpus allowing them to effectively handle the noise in the tweets.
2. Incorporating ML and DL classifiers with transformer embeddings to avoid higher computation for the LLMs such as BERT.
3. Using the Llama model for the task of twitter sentiment analysis as it has proven to provide better accuracy but it comes with the downside of billions of trainable parameters.
4. Comparing the efficiency of the models across diverse twitter datasets that contain neutral sentiment.

References

[1] Go, A., Bhayani, R., & Huang, L. (2009). Twitter sentiment classification using distant supervision. CS224N project report, Stanford, 1(12), 2009.

[2] Cai, M. (2018, December). Sentiment analysis of tweets using deep neural architectures. In Proceedings of the 32nd Conference on Neural Information Processing Systems (pp. 1-8).

[3] Barreto, S., Moura, R., Carvalho, J., Paes, A., & Plastino, A. (2023). Sentiment analysis in tweets: an assessment study from classical to modern word representation models. Data Mining and Knowledge Discovery, 37(1), 318-380.

[4] Cliche, M. (2017). BB_twtr at SemEval-2017 task 4: Twitter sentiment analysis with CNNs and LSTMs. arXiv preprint arXiv:1704.06125.

[5] Horne, L., Matti, M., Pourjafar, P., & Wang, Z. (2020, December). GRU-BERT: A GRU-based method to fuse BERT hidden layers for Twitter sentiment analysis. In Proceedings of the 1st conference of the Asia-Pacific chapter of the association for computational linguistics and the 10th international joint conference on natural language processing: Student research workshop (pp. 130-138).

[6] Júnior, E. A. C., Marinho, V. Q., & dos Santos, L. B. (2017, August). NILC-USP at semeval-2017 task 4: A multi-view ensemble for twitter sentiment analysis. In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017) (pp. 611-615).

[7] Ignatov, D., & Ignatov, A. (2017, November). Decision stream: Cultivating deep decision trees. In 2017 ieee 29th international conference on tools with artificial intelligence (ictai) (pp. 905-912). IEEE.

[8] Balikas, G., Moura, S., & Amini, M. R. (2017, August). Multitask learning for fine-grained twitter sentiment analysis. In Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval (pp. 1005-1008).

[9] Nguyen, D. Q., Vu, T., & Nguyen, A. T. (2020). BERTweet: A pre-trained language model for English Tweets. arXiv preprint arXiv:2005.10200.

[10] Sanh, V. (2019). DistilBERT, A Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. arXiv preprint arXiv:1910.01108.

[11] Liu, Y. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

[12] Devlin, J. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[13] Vaswani, A. (2017). Attention is all you need. Advances in Neural Information Processing Systems.

[14] Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. arXiv preprint arXiv:1801.06146.

[15] Brown, T. B. (2020). Language models are few-shot learners. arXiv preprint arXiv:2005.14165.