Unfolding Emotions on Twitter: Navigating Sentiment Analysis with Advanced Neural Architectures

Pravesh Pradheep*
Department of Computer Science
University College Dublin
Dublin, Ireland
pravesh.pradheep@ucdconnect.ie

Abstract

Sentiment analysis applies natural language processing to understand user sentiments. Twitter, a golden repository with user opinions, offers a rich dataset for analysis. This paper compares the efficiency of different deep learning models for sentiment analysis: a Multi-Layer Perceptron (MLP), a Convolutional Neural Network (CNN), a Bi-directional Long Short-Term Memory (Bi-LSTM), and a Bi-LSTM with Multi-Head Attention. Our findings reveal that while the MLP lay a solid foundation, the CNN and Bi-LSTM models, particularly with attention, excel in contextual understanding. The attention-enhanced Bi-LSTM model emerges as a better model for sentiment analysis, providing insights for future developments in sentiment analysis algorithms.

1 Introduction

In the digital age, social media platforms like Twitter have become central to the spread of immediate public opinion. Sentiment analysis, a subfield of natural language processing (NLP), aims to automate the interpretation of emotions expressed in text, offering insights into the collective mood on a wide range of topics. This technology not only aids businesses in understanding consumer responses but also assists policymakers and researchers in tracking public sentiment toward various issues.

Traditional methods for sentiment analysis have heavily relied on machine learning algorithms such as Support Vector Machines (SVM), Naive Bayes, and Decision Trees, coupled with feature extraction techniques like TF-IDF (Term Frequency-Inverse Document Frequency). While these are effective in structured and expansive texts, these methods often fail with the unstructured, concise nature of tweets, which contains slang, abbreviations, and emoticons. The necessity for well-defined features and the limitation of traditional algorithms in handling the semantic complexity of language present significant challenges.

The emergence of deep learning marked a paradigm shift in this field. Deep learning models excel at capturing the various complexity and semantic linkages found in natural language because they use structures capable of learning sequence representations of data. Notably, neural networks such as Multilayer Perceptrons (MLP), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNNs), as well as its derivatives such as Long Short-Term Memory (LSTM) networks, have made substantial contributions to the field.

This paper evaluates and compares the effectiveness of multiple deep learning models for sentiment analysis using Twitter data. We investigate a simple MLP, a CNN, and a bidirectional long short-term memory (Bi-LSTM) model. In addition, we improve the Bi-LSTM model with a Multi-Head Attention

*Student at University College Dublin. MSc Computer Science (Negotiated Learning). Student number 23200495

Preprint. Under review.

mechanism, believing that this enhancement will help the model to better focus on significant regions of text sequences, thus improving its explainability in text sentiment analysis.

Our comparison analysis tries to determine which model best reflects the complex dynamics of sentiment expressed in tweets, taking into account the unique constraints presented by Twitter's brevity and informal language. By rigorously assessing the performance of each model, this study hopes to shed light on the potential of sophisticated neural architectures in sentiment analysis, paving the way for future research and application improvements in real-world sentiment tracking.

2 Related Work

Historically, sentiment analysis was dominated by classical machine learning algorithms. Among these, Support Vector Machines (SVMs) and Naive Bayes were notable for their effectiveness in handling high-dimensional data derived from structured textual features. Early work by Pang et al. (2002)[1] utilized bag-of-words (BoW) and term frequency-inverse document frequency (TF-IDF) to capture the presence or absence of crucial words and phrases manually selected for their sentiment prediction ability. These models, however, struggled with the position of words in a sentence and nuances of social media text, which often departs significantly from formal language structures.

The field of sentiment analysis witnessed a significant transformation with the adoption of deep learning models, capable of end-to-end feature learning from data. This shift began with the deployment of neural networks such as multi-layer perceptrons (MLPs), which were able to discern non-linear relationships and interactions between words more effectively than linear models (Socher et al., 2013)[2].

Further progress was marked by the introduction of convolutional neural networks (CNNs) and long short-term memory networks (LSTMs) in sentiment analysis. CNNs, renowned for their feature extraction capabilities in image processing, were adapted to NLP tasks to identify local dependencies and dynamically learn salient text features (Kim, 2014)[3]. Simultaneously, LSTMs addressed the shortcomings of earlier recurrent neural networks by capturing long-range dependencies within texts, which is crucial for understanding context and semantic meanings across extended sequences (Hochreiter & Schmidhuber, 1997)[4].

The refinement of LSTM models into bidirectional versions (BiLSTMs) enabled the analysis of context from both past and future inputs symmetrically, which significantly enhanced the understanding of contextual nuances in sentiment analysis (Schuster & Paliwal, 1997)[5]. The subsequent integration of attention mechanisms with BiLSTMs marked a significant innovation, allowing models to concentrate on specific text segments that are particularly informative for sentiment analysis. This focus notably improved both the accuracy and interpretability of the analysis results (Bahdanau et al., 2015)[6].

3 Experimental Setup

3.1 Pre-processing Approach

The pre-processing of textual data is a critical step in the workflow of sentiment analysis, particularly when dealing with informal and unstructured text such as tweets. The uniqueness of Twitter data, characterized by its briefness and non-adherence to grammatical rules, necessitates a robust pre-processing strategy to effectively categorize sentiments or emotions. Here, we outline the data collection and pre-processing methods employed in this study.

3.1.1 Data Collection

For this paper, we sourced our data from a publicly available dataset on Kaggle, which comprises 916,575 tweets tagged with specific emotions such as "angry," "disappointed," and "happy"(https://www.kaggle.com/datasets/kosweet/cleaned-emotion-extraction-dataset-from-twitter). This dataset contains 3 columns:

- *Emotion*: This column describes the emotion of the tweet.

- *Content*: The content of the tweet.
- *Cleaned Content*: The tweet content cleaned by the author.

Tweets pose distinct challenges for sentiment analysis:

- *Length*: Tweets are inherently brief, making it difficult to extract context or the full mood.
- *Grammar and Syntax*: The vocabulary used in tweets frequently deviates from traditional grammatical standards. This includes using unorthodox punctuation, capitalization, and sentence patterns.
- *Slang and Abbreviations*: Twitter language comprises a lot of slang, abbreviations, and acronyms that are not often used in normal language datasets.

Given the nature of Twitter data, our pre-processing approach involves several techniques designed to standardize and clarify the text for analysis:

(a) *Case Normalization*: All tweet content is converted to lowercase to standardize the dataset, as capitalization typically does not influence the sentiment conveyed by words.

(b) *URL Removal*: URLs are extracted and removed using regular expressions, as they do not contribute meaningful sentiment-related information.

(c) *Hashtags and Numbers*: Hashtags and numerical data, which generally do not add value to sentiment analysis, are removed from tweets via regular expression matching.

(d) *HTML Entities*: Instances of HTML are decoded and removed to ensure that only plain text is analyzed. This is done using the Beautiful Soup library (‘bs4‘), which parses and cleans data from HTML content.

(e) *Acronym Expansion*: Due to the briefness of tweets, which often contain acronyms, common acronyms are expanded to their full expressions to clarify their sentiment implications.

(f) *Emoticons*: Emoticons are converted into their corresponding textual descriptions, as they play a crucial role in expressing sentiments in tweets.

(g) *RT Mentions Removal*: User mentions, indicated by the ’@’ symbol, are removed as they primarily serve to tag or address specific users and do not contribute to the overall sentiment.

(h) *Contraction Expansion*: Twitter’s character limit encourages the use of contractions (e.g., ’would’ve’, ’shouldn’t’). All such contractions are expanded to their full forms to ensure clarity in sentiment analysis.

(i) *Tokenization*: The text in tweets is tokenized using the WordPiece tokenization approach. This approach is very useful for Twitter data since it efficiently handles a wide range of phrases seen in tweets, such as slang expressions and misspellings. WordPiece tokenization divides words into sub-word units to better capture the meaning of prefixes, suffixes, and infixes inside words, improving the model’s capacity to recognise and process the numerous linguistic intricacies of social media content. We restrict each sentence to only have a maximum of 120 tokens.

3.2 Modelling

3.2.1 Word Embeddings

Word embeddings are a type of word representation that allows words with similar meanings to have similar representations. They are essential in natural language processing (NLP) tasks, offering a dense low-dimensional representation of words.

Custom Embedding Layer: Unlike utilizing pre-trained models such as Word2Vec, GloVe, or FastText, this study employs randomly initialized embeddings that are learned during the model training process. The vocabulary for the embedding layer consists of 30,522 tokens and input length of 120 and an embedding space of 128 dimensions are chosen.

Application in This Study: For our sentiment analysis, the embeddings layer starts with random weights and learns an optimal representation of words directly from the corpus during training.

3.2.2 Multi-Layer Perceptron (MLP)

A Multi-Layer Perceptron (MLP) is a feedforward artificial neural network that includes multiple layers of nodes, each fully connected to the next. MLPs are extensively utilized in machine learning for modeling complex non-linear relationships through their layered structure.

Architecture: The MLP used in this study consists of the following layers:

- **Input Layer**: Receives input features from the embeddings obtained from the processed tweets.
- **Hidden Layers**:The Multi-Layer Perceptron (MLP) comprises two hidden layers that apply non-linear transformations to the inputs. Each layer l computes its outputs as follows:

  $$h_l = ReLU(W_l h_{l-1} + b_l)$$

  where W_l denotes the weight matrix, b_l is the bias vector, and h_{l-1} represents the output from the previous layer or the input for $l = 1$. The ReLU (Rectified Linear Unit) function is employed as the activation function to introduce non-linearity. .
- **Output Layer**: Produces the final output of the network. For classification, this layer typically uses a softmax function, given by $y_k = \frac{e^{z_k}}{\sum_j e^{z_j}}$ for each class k, where z_k is the input to the output neurons.

3.2.3 1D Convolutional Layer(CNN):

Convolutional Neural Networks have traditionally proved to be effiective used in the field of image processing, image recognition. Convolutional Neural Networks (CNNs) have been recently adapted for text processing tasks, leveraging their ability to capture local features through the application of convolutional filters.

Architecture and Configuration: In this study, the CNN architecture includes one-dimensional (1D) convolutional layers specifically designed to process textual data. These layers apply filters of varying sizes to capture different n-gram features from the text. The configuration of the convolutional layers is as follows:

- Filters of size 3 with 64 feature maps.
- Filters of size 4 with 128 feature maps.
- Filters of size 5 with 256 feature maps.

Each filter size corresponds to capturing the semantic information from n-grams of different lengths, allowing the model to learn a diverse set of textual features.

Functionality: The convolutional layers work by sliding filters over the input text data to generate a feature map that captures the presence of specific patterns. These features are then pooled together to reduce the dimensionality and helps model learn high-level features.

Output and Classification: The pooled features from all convolutional layers are concatenated and fed into a fully connected layer, which then outputs to a softmax function given by,$y_k = \frac{e^{z_k}}{\sum_j e^{z_j}}$ for each class k, where z_k is the input to the output neurons.

3.2.4 Long Short Term Memory(Bi-LSTM):

The LSTM architecture, introduced by Hochreiter and Schmidhuber[4], is designed to overcome the limitations of traditional recurrent neural networks by incorporating a gating mechanism. These gates control the flow of information, allowing the network to retain longer dependencies and mitigate the vanishing gradient problem. Bi-directional Long Short-Term Memory (BiLSTM) networks enhance the traditional LSTM architecture by processing data in both past and future contexts.

3.2.5 Bi-directional Long Short Term Memory(BiLSTM):

Architecture and Configuration: In this study, the BiLSTM architecture is configured as follows:

- **Input Dimension**: Each input sequence consists of 128-dimensional vectors, reflecting the rich feature set designed to capture the nuances of textual data.
- **Sequence Length**: The model processes sequences that are 120 tokens long, allowing it to handle most lengths typical for the content under analysis, such as tweets or short comments.
- **Hidden Units**: Each direction of the BiLSTM layers contains 128 hidden units. This size is chosen to balance the model's complexity and computational efficiency while providing sufficient capacity to learn detailed dependencies.
- **Number of Layers**: The network includes 6 stacked BiLSTM layers, enhancing its ability to model more complex patterns. Stacking multiple layers allows the network to learn a hierarchy of features at different levels of abstraction.

Functionality: The Bi-directional LSTM (BiLSTM) architecture enhances the capability of traditional LSTMs by processing data in both forward and backward directions, allowing it to capture information from both past and future contexts within a sequence.

Forward LSTM: For the forward LSTM pass, the calculations at each time step t are as follows:
$\mathrm{i}_t^f = \sigma(W_{xi}^f x_t + W_{hi}^f h_{t-1}^f + W_{ci}^f c_{t-1}^f + b_i^f)$
$f_t^f = \sigma(W_{xf}^f x_t + W_{hf}^f h_{t-1}^f + W_{cf}^f c_{t-1}^f + b_f^f)$
$c_t^f = f_t^f \cdot c_{t-1}^f + i_t^f \cdot \tanh(W_{xc}^f x_t + W_{hc}^f h_{t-1}^f + b_c^f)$
$o_t^f = \sigma(W_{xo}^f x_t + W_{ho}^f h_{t-1}^f + W_{co}^f c_t^f + b_o^f)$
$h_t^f = o_t^f \cdot \tanh(c_t^f)$

Backward LSTM: For the backward LSTM pass, the calculations at each time step t are as follows:
$\mathrm{i}_t^b = \sigma(W_{xi}^b x_t + W_{hi}^b h_{t+1}^b + W_{ci}^b c_{t+1}^b + b_i^b)$
$f_t^b = \sigma(W_{xf}^b x_t + W_{hf}^b h_{t+1}^b + W_{cf}^b c_{t+1}^b + b_f^b)$
$c_t^b = f_t^b \cdot c_{t+1}^b + i_t^b \cdot \tanh(W_{xc}^b x_t + W_{hc}^b h_{t+1}^b + b_c^b)$
$o_t^b = \sigma(W_{xo}^b x_t + W_{ho}^b h_{t+1}^b + W_{co}^b c_t^b + b_o^b)$
$h_t^b = o_t^b \cdot \tanh(c_t^b)$

In these equations, σ represents the sigmoid activation function, $\tanh$ is the hyperbolic tangent function, W and b denote the weights and biases specific to each gate and each LSTM direction (forward f and backward b), x_t is the input vector at time step t, h_t is the hidden state, and c_t is the cell state.

Output and Classification: The output from both the forward and backward pass are concatenated using element wise addition and fed into a fully connected layer,the softmax function, which is used to normalize the outputs of the fully connected layer into a probability distribution over predicted output classes. The softmax function for each class k is given by the following equation:

$$y_k = \frac{e^{z_k}}{\sum_j e^{z_j}}$$

where y_k represents the probability that the input belongs to class k, and $z_k = h_t^f + h_t^b$ denotes the input to the output neurons from the fully connected layer.

3.2.6 Bi-directional Long Short Term Memory with Multi-Head Attention(Bi-LSTM-MHAM):

Integration of Multi-Head Attention: The Multi-Head Attention Mechanism, inspired by the Transformer architecture[7], is made up of many attention heads. Each head individually attends to input from various representation at different places.

Attention: The attention mechanism computes a set of attention scores, which indicate the importance of each part of the input data. This is done by:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where Q (Query), K (Key), and V (Value) are matrices produced by linear transformations of the input, and d_k is the dimension of the keys. Each "head" in the MHAM processes the input independently using this formula, allowing the model to capture different types of dependencies in the data.

Integration Bi-LSTM with Attention: The output from the Bi-directional LSTM (Bi-LSTM) layer, represented as $z_k = h_t^f + h_t^b$, encapsulates both forward and backward contextual information at each time step. This output is subsequently fed into a Multi-Head Attention layer, which enhances the model's ability to focus on relevant features across different representation subspaces. The following transformation taking place at each head:

1. Compute the query Q_i, key K_i, and value V_i vectors by applying linear transformations followed by a dropout operation:
   $Q_i = Dropout(W_{Q_i} \cdot lstm\_outputs + b_{Q_i})$
   $K_i = Dropout(W_{K_i} \cdot lstm\_outputs + b_{K_i})$
   $V_i = Dropout(W_{V_i} \cdot lstm\_outputs + b_{V_i})$ where W_{Q_i}, W_{K_i}, and W_{V_i} are the weight matrices, and b_{Q_i}, b_{K_i}, and b_{V_i} are the bias vectors for the i^{th} attention head.
2. Calculate the attention scores by taking the dot product of Q_i and the transpose of K_i, followed by scaling with the dimension of the key vectors and applying the softmax function: $\text{scores}_i = Softmax\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right)$ where d_k is the dimensionality of the key vectors.
3. Generate the output of the attention head by performing a weighted sum of the value vectors V_i, using the attention scores: $\text{output}_i = scores_i V_i$
4. The final output from all heads is concatenated to form the overall output of the Multi-Head Attention layer: final_output = Concatenate($\text{output}_1, output_2, \ldots, output_{num\_heads}$)

Output and Processing: The outputs from each attention head are concatenated and passed through a final linear layer to combine the different learned aspects into a cohesive output, which is then fed into a softmax layer for classification. This configuration enables the Bi-LSTM with MHAM to leverage both the sequential depth provided by the LSTM layers and the focused attention provided by the MHAM, offering a enhanced understanding of textual data.

3.2.7 Training

The models are trained to minimize the cross-entropy loss , an effective measure of prediction error for classification tasks.

Backpropagation: Optimization during training is achieved through backpropagation[8], a method that computes the gradient of the loss function with respect to each weight and bias by applying the chain rule. This process is essential for updating the parameters in the direction that minimally reduces prediction error.

Adadelta Optimizer: To optimize these parameters, the MLP employs Adadelta[9], an advanced form of AdaGrad. Adadelta addresses the rapid decrease in learning rates associated with AdaGrad by limiting the accumulation of all past squared gradients to a fixed-size window. It calculates updates as:

$$\Delta x_t = -\frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \cdot g_t,$$

where g_t represents the gradient at time step t, $E[g^2]_t$ is the moving average of the squared gradients, and ϵ is a small constant to prevent division by zero. This approach helps maintain an adaptive learning rate that improves convergence behaviors, particularly suitable for handling the sparse and noisy data typically seen in tweets.

Table 1: Dataset Distribution

Category	Disappointed	Angry	Happy	Total
TRAIN	219864	210280	211458	641602
VALID	46830	45402	45254	137486
TEST	47020	45308	45159	137487

Table 2: Model Performance Metrics

	Class 1: angry %			Class 2: disappointed %			Class 3: happy %			
Model	p	r	f1	p	r	f1	p	r	f1	acc /%
MLP	70.39	81.6	75.58	70.45	63.93	67.07	81.54	76.32	78.84	73.85
CNN	81.48	91.01	85.99	89.14	82.57	85.73	96.51	92.54	94.48	88.65
BiLSTM	81.81	91.25	86.28	88.82	83.86	86.27	98.59	92.88	95.65	89.28
BiLSTM-MHAT	82.32	91.50	86.67	89.18	84.14	86.58	98.11	92.87	95.41	89.45

4 Results:

The evaluation of classifiers were performed on 137,487 tweets from the dataset and we compare it against 4 models that was developed. Table 1 depicts the number of samples in each class used for testing.

Figure 2 shows the training and validation losses for each model. The convergence of training and validation losses across all models shows effective generalization with no evidence of overfitting.

We evaluated the performance of various models on the sentiment analysis task, including Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), Bi-directional Long Short-Term Memory (BiLSTM), and BiLSTM with Multi-Head Attention (BiLSTM-MHAT). The models were assessed across three sentiment classes—angry, disappointed, and happy—using precision (p), recall (r), F1 score (f1), and overall accuracy (acc) as shown in Table 2.

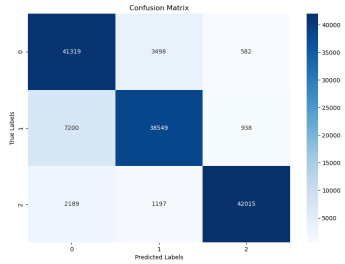
The MLP model showed reasonable performance with a balance between recall and precision across the classes. Specifically, the MLP achieved a precision of 70.39% and recall of 81.6% for the angry class, resulting in an F1 score of 75.58%. For the disappointed class, the precision dropped to 70.45% with a recall of 63.93%, leading to an F1 score of 67.07%. The model performed better in identifying the happy class with a precision of 81.54%, recall of 76.32%, and an F1 score of 78.84%. Overall, the MLP model attained an accuracy of 73.85%.

The CNN model demonstrated superior performance compared to the MLP, especially in terms of precision. It reached a high precision of 81.48% and recall of 91.01% for the angry class, with an F1 score of 85.99%. The model was consistent across the disappointed class with precision of 89.14%, and an F1 score of 85.73%, alongside a recall of 82.57%. For the happy class, the CNN model achieved its best performance, recording a precision of 96.51%, recall of 92.54%, and an impressive F1 score of 94.48%. Overall accuracy for the CNN was 88.65%.

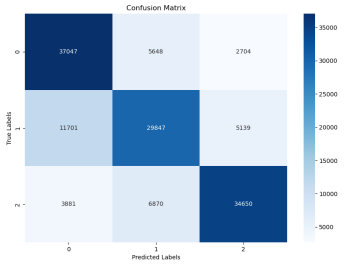
BiLSTM models performs better in sequence processing, achieving F1 scores of 86.28% for angry, 86.27% for disappointed, and 94.48% for happy, with an overall accuracy of 89.28%. Adding the Multi-Head Attention mechanism to the BiLSTM-MHAT model improved results, increasing F1 scores to 86.67%, 86.58%, and 95.41% for angry, disappointed, and happy classes, respectively, with an overall accuracy of 89.45%. The BiLSTM-MHAT model outperforms the BiLSTM model slightly while improving model explainability through attention weights.

5 Conclusions Future Work

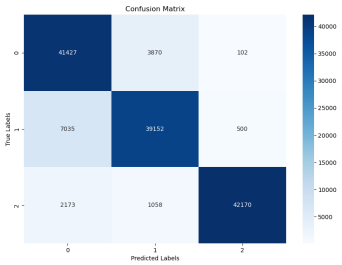
In conclusion, this study presented a comprehensive evaluation of different deep learning architectures for sentiment analysis. Our results indicate that while conventional models like MLP and CNN are effective for such tasks, advanced architectures like BiLSTM enhance the ability to capture sequential information, thereby improving performance. Notably, the integration of Multi-Head Attention with



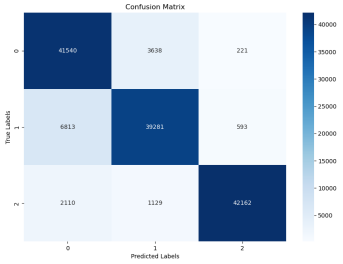
(a) Confusion Matrix for CNN Model



(b) Confusion Matrix for MLP Model



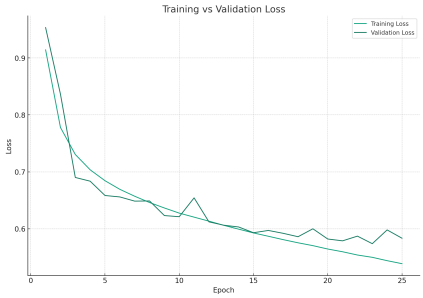
(c) Confusion Matrix for BiLSTM Model



(d) Confusion Matrix for BiLSTM-MHAM Model

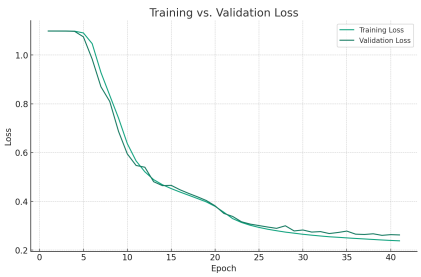
Figure 1: Comparative confusion matrices for different neural network architectures used in sentiment analysis of Twitter data.

(a) Training vs Validation Loss - CNN



(b) Training vs Validation Loss - MLP

(c) Training vs Validation Loss - BiLSTM Model



(d) Training vs Validation Loss - BiLSTM-MHAH Model

Figure 2: Training vs Validation loss for models used in the study

BiLSTM (BiLSTM-MHAT) provided the best overall performance, highlighting the effectiveness of attention mechanisms in identifying important features across different sentiment contexts.

Future research will explore the incorporation of pre-trained word embeddings and more sophisticated transformer-based models that can capture nuanced emotional expressions. Additionally, investigating cross-lingual and multi-modal sentiment analysis could extend the applicability of our findings. The adaptability of models to low-resource languages and the integration of unsupervised learning techniques also present promising avenues for exploration.

References

[1] Thumbs up? Sentiment Classification using Machine Learning Techniques(https://aclanthology.org/W02-1011) (Pang et al., EMNLP 2002).

[2] Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank(https://aclanthology.org/D13-1170) (Socher et al., EMNLP 2013).

[3] Kim, Y. (2014) Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, 1746-1751. arXiv: 1408.5882 https://doi.org/10.3115/v1/D14-1181.

[4] Hochreiter, S. and Schmidhuber, J. (1997) Long Short-Term Memory. Neural Computation, 9, 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

[5] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," in IEEE Transactions on Signal Processing, vol. 45, no. 11, pp. 2673-2681, Nov. 1997, doi: 10.1109/78.650093. keywords: Recurrent neural networks;Artificial neural networks;Training data;Databases;Probability;Shape;Parameter estimation;Speech recognition;Control systems;Telecommunication control,

[6] Dzmitry Bahdanau, KyunghyunCho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In ICLR.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[8] Rumelhart, D., Hinton, G. Williams, R. Learning representations by back-propagating errors. Nature 323, 533–536 (1986). https://doi.org/10.1038/323533a0

[9] Zeiler, Matthew. (2012). ADADELTA: An adaptive learning rate method. 1212.