Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control
Journal homepage: http://kinetik.umm.ac.id
ISSN: 2503-2267
Vol. 10, No. 3, August, Pp. 261-270

Analysis of mental health disorders via social media mining using LSTM and Bi-LSTM

Binti Kholifah*[1], Iwan Syarif[2], Tessy Badriyah[2]
Universitas Negeri Surabaya, Indonesia[1]
Politeknik Elektronika Negeri Surabaya, Indonesia[2]

Article Info
Keywords:
Mental Health, Text Pre-processing, Word Embedding, LSTM, Bi-LSTM

Article history:
Received: December 24, 2024
Accepted: March 19, 2025
Published: August 31, 2025

Cite:
B. Kholifah, I. Syarif, and T. Badriyah, "Analysis of Mental Health Disorders via Social Media Mining Using LSTM and Bi-LSTM ", *KINETIK*, vol. 10, no. 3, Aug. 2025.
https://doi.org/10.22219/kinetik.v10i3.2205

*Corresponding author.
Binti Kholifah
E-mail address:
bintikholifah@unesa.ac.id

Abstract
Mental health disorders are a growing global concern, with many individuals lacking early detection and appropriate treatment. Mental illness can impact a person's quality of life and often goes undetected until symptoms worsen. One contributing factor to this problem is the limited ways to detect mental disorders in their early stages. Social media, especially platform X, offers the potential to analyze users' emotional expressions that may indicate a mental disorder, such as depression or anxiety. Psychological symptoms can be explored more broadly using Natural Language Processing. This study optimizes several text preprocessing techniques to extract meaningful information from social media text. To convert words into numerical vectors, several word embedding methods are used, such as Word2Vec, FastText, and GloVe. Meanwhile, the classification process is carried out using LSTM and Bi-LSTM because they are considered capable of studying data sequence patterns, such as sentence structure, effectively. The results show that the addition of expanding contractions, emoticon handling, negation handling, repeated character handling, and spelling correction in the preprocessing text can improve the model performance. In addition, Bi-LSTM with pre-trained FastText shows better results than the other methods in all experiments, achieving 86% accuracy, 87.5% precision, 84% recall, and 85.71% F1-Score.

1. Introduction

Based on data collected by the World Health Organization (WHO), around 720,000 people die due to suicide every year. Suicide is one of the leading causes of death in the world, especially among teenagers and young adults aged 15 to 29 years [1]. According to the National Criminal Information Center of the Criminal Investigation Agency of the Indonesian National Police, there were 1,023 suicide cases during the period from January to October 2024 [2]. Mental health is influenced by a variety of complex and interrelated factors, including biological, genetic, psychological, cultural, and environmental aspects [3]. Loneliness has an impact on an individual's mental health [4] as lonely individuals often feel like they do not have any friend and safe space to express their feelings. Social media platforms like X is one of the media that can be a medium for people to share their mental health experiences [5]. Individuals who want to express their feelings deeply often show emotional expressions through social media [6].

The background of this research is that the widespread use of social media presents a unique opportunity to identify early signs of mental health disorders by analyzing patterns in user-generated content. Early detection of mental health issues and appropriate treatment can mitigate the problems they cause. However, early detection of mental health disorders remains a challenge, particularly in identifying individuals who show signs of anxiety or depression through social media, because emotional expressions can vary and are difficult to interpret without the right methods. Therefore, this study offers a computational approach that can identify hidden patterns in text to accurately detect indications of mental disorders. Statements on platform X can express the feelings or moods of the writer. Individuals with mental health disorders tend to write statements that reflect negative emotions and expressions such as feeling depressed, weak, useless and even suicidal. Social media offers unprecedented opportunities to explore hidden patterns in statements (text). The resulting patterns can be valuable for X users, mental health organizations, and researchers in determining the underlying factors influencing a person's behavior [7]. The features contained in a statement on the X platform can be extracted using several word embedding methods, such as Word2Vec, FastText, and GloVe. Word embedding aims to convert a statement into a vector of numbers so that computers and algorithms can understand the meaning of the statement [8].

This study aims to analyze mental health disorders through X user reviews using LSTM (Long Short-Term Memory) and Bi-LSTM (Bidirectional Long Short-Term Memory). These models have shown remarkable success in capturing long-term dependencies, sequential patterns, and contextual information in textual data [9]. The LSTM model shows good results in recognizing individual personality traits [10]. LSTM is able to overcome the gradient magnification problem that often occurs in RNNs, which improves the model's ability to handle long and complex word sequences with an accuracy of up to 94% [11]. Bi-LSTM is an extended variant of the LSTM model that addresses its limitations. Bi-LSTM processes incoming data both forward and backward, allowing it to gain a complete understanding of the context in which the data resides [12]. Meanwhile, Bi-LSTM, with its ability to process data bidirectionally (both from previous and subsequent contexts), has proven effective in detecting difficult-to-analyze sarcasm, achieving an accuracy of over 95% [13]. In the same case study, the LSTM model successfully detected mental health disorders with an accuracy of 70.89%, a precision of 50.24%, and a recall of 70.89%. However, the feature extraction process in this model has not been carried out optimally, which may affect detection performance [5]. Although both approaches have proven effective, further research on improving LSTM performance, especially in the context of text pre-processing optimization, remains very limited.

In this study, features from text are extracted using word embedding methods such as Word2Vec, FastText, and GloVe. Word embedding aims to convert text into a numerical representation so that it can be understood by computers and machine learning algorithms [8]. Additionally, this study optimizes text pre-processing to explore features contained in text data to improve model accuracy. Several pre-processing techniques, such as emoticon handling, are used in this study. Pre-processing plays a significant role in reducing the ambiguity of meaning in short texts. This process is not only important in short text classification but also plays a crucial role in data clustering and anomaly detection [14]. The selection of the right pre-processing technique must be adjusted to the task and model used. Optimal pre-processing techniques can significantly improve classification accuracy compared to using less effective techniques [15]. With this approach, this study not only contributes to the development of an automatic method for detecting mental health disorders based on text analysis on social media but also introduces a more accurate and contextual pre-processing strategy, ensuring that the classification results are more optimal and reliable.

2. Research Method

The system design in this study can be seen in Figure 1. The system consists of five main parts, including data collection, data pre-processing to clean noisy text, word embedding to convert words into numerical vectors (using Word2Vec, FastText, and GloVe), model training with LSTM and Bi-LSTM, and data classification using the model obtained from the training process.

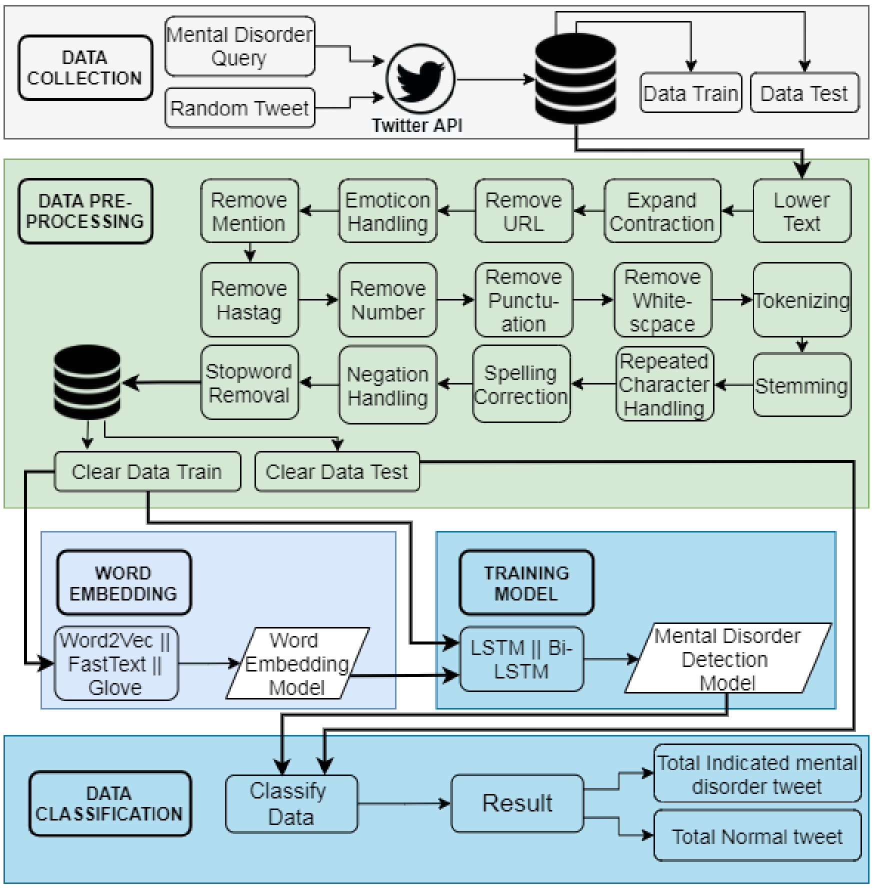

Figure 1. System Design

2.1 Data Collection

Data is collected from statements of X users using the X API. The data used is primary data obtained from X postings with a minimum of 75% in English. Data on X users who are indicated to have mental health disorders can be obtained by determining keywords—namely, words that are often spoken or words that indicate symptoms of mental disorders, such as depression, suicide, and feelings of loneliness and hopelessness [16]. The data obtained must be selected to match the symptoms shown by people with mental disorders. Examples of the data can be seen in Table 1.

Table 1. Dataset Sample

text	label
Damn I'm fully experiencing sadness again I'm so useless	1
Well played radio station. Playing the song 'Total Eclipse of the Heart' at the start of the eclipse!	0

The data consists of two features, namely text and label features. The text feature contains statements from tweets, while the label feature contains information on whether the statement in the text feature indicates a mental health disorder. Label 1 indicates data with indications of mental health disorders, while label 0 indicates normal user data.

2.2 Data Pre-processing

The data obtained still contains unnecessary characters and even interferes with the next process. Therefore, pre-processing is needed for both training data and testing data. Sample results before and after data pre-processing can be seen in Table 2.

Table 2. Pre-processing results

before	after
@apa I’m depressed and lonily in 2020 :(i am not happy............ www.img.com	I depress lonely sad i unhappy

In the first stage, all text is converted to lowercase. For example, the words ‘Hello’ and ‘hello’ are considered the same word after going through the lowercase stage, thus reducing data redundancy [17]. In the second stage, abbreviations have also been reverted to their original form such as “don’t” to “do not” and “I’m” to “I am”. Modern abbreviations like “idk” have also been reverted to their original form “I do not know”. We also use the emoticon and emoji library to convert the emoticons in the data into text according to their meaning. URLs, mentions, hashtags, numbers, punctuation, and whitespace are also removed so that the system only processes data in the form of letters [18].

The next stage is to tokenize or break the sentence into parts per word. Then, each word is changed into its root form. This can reduce the number of words and speed up computing [19]. People often express words in text with an emphasis. The computer does not know that “loooove” is “love” unless told. Repeated characters should be fixed so that a standard word can be obtained. In addition, we also perform spelling correction to check the spelling of words.

The statements "I am happy" and "I am not happy" have opposite meanings, where the word "not" changes a positive statement into a negative one [20]. Negation words can change the meaning of a sentence. Handling negation has been shown to improve deep learning performance [21]. The final stage is to remove stop words such as “a”, “an”, “the” as well as conjunctions and subject words. Stop words are removed because they do not provide relevant information to the text [22].

2.3 Word Embedding

In simple terms, word embedding is a technique for converting a word into a vector or array consisting of a collection of numbers. In this study, the word embedding techniques used are Word2Vec, FastText, and GloVe with Skip-gram architecture.

At the word embedding stage, a large amount of corpus data is needed to increase the effectiveness of word embedding. This study attempts to conduct training on corpus datasets and pre-trained word embedding provided by several sources. Training with corpus datasets means that the corpus used consists of words that only exist in the dataset [23]. Meanwhile, in pre-trained word embeddings, large corpora from several sources such as Google News, Wikipedia, and Facebook, are used with various sizes ranging from 25 dimensions to 300 dimensions that have been trained. These large corpora have been trained with one of the word embedding techniques and their pretrained files can be easily found on the official website of each word embedding technique in the format *.bin or *.vec [24]. The Word2Vec model can learn semantic knowledge from a large corpus without supervision. The Skip-gram architecture predicts the output from the current input. For example, if the training data used is "I feel so hopeless and useless", assume that all data has been pre-processed so that the one-hot encoded input data includes: feel (1 0 0), hopeless (0 1 0), and useless (0 0 1) [25].

FastText is one of the word-embedding methods developed by Facebook, building upon Word2Vec. FastText can handle words that have never been encountered before (Out Of Vocabulary word, or OOV). In the output layer, FastText uses hierarchical softmax to predict labels based on input. The use of hierarchical softmax can also speed up the training process [26]. Meanwhile, GloVe has the ability to make simple observations that the probability ratio of the co-occurrence of words can have several meanings. For example, in the probability of co-occurrence of the words “ice” and “steam”, the word “ice” appears more often in solid state than gas, while “steam” appears more often in gaseous state than solid [27].

2.4 Model Training

The LSTM architecture is shown in Figure 2. The LSTM model has a unique structure with three special gates. First, the forget gate determines how much old information is stored in the memory cell. Second, the input gate controls how much new information from the current input should be updated in the memory cell. Finally, the output gate determines whether the memory cell should affect the output at the current time step [28].

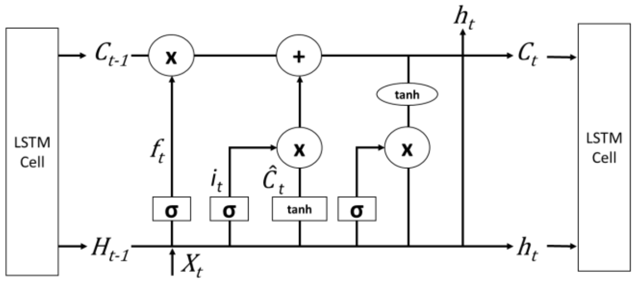

Figure 2. LSTM Architecture

The first stage is represented by Equation 1, where h_{t-1} and x_t pass through the sigmoid gate in Equation 2. The LSTM decides what information to remove from the cell state. The decision is made by the sigmoid layer, which can be called the “Forget Gate”. The forget gate processes h_{t-1} and x_t as input and produces an output of 0 or 1 in the cell state C_{t-1}. An output of 1 means approval to pass, while an output of 0 means disapproval to pass [29].

$$(x) = \frac{1}{1 + exp^{-1}} \tag{1}$$

$$f_t = \sigma(W_f . [h_{t-1}, X_t] + b_f \tag{2}$$

Where,

x	: input data
exp	: mathematical constant
f_t	: Forget gate
σ	: Sigmoid function
W_f	: Weight value for forget gate
h_{t-1}	: Output from previous cell
X_t	: Current input
b_f	: Bias value on forget gate

The next stage is to determine the new information to be stored in the cell state using Equation 3. The sigmoid layer, called the input gate, decides which values to update. Then, the tanh layer creates a new value vector, namely $\tilde{C}_t$, which can be added to the cell state using Equation 4. The two values are combined to create a new state [29].

$$i_t = \sigma(W_i . [h_{t-1}, x_t] + b_i) \tag{3}$$

$$\tilde{C}_t = tanh\ (W_c . [h_{t-1}] + b_c) \tag{4}$$

Where,

i_t	: Input gate
W_i	: Weight value for gate input
b_i	: Bias value at gate input
$\tilde{C}_t$	: New value that can be added to the cell state
W_c	: Weight value for cell state
b_c	: Bias value at cell state

The next stage is to update the old cell state, $C_{(t-1)}$, to the new cell state C_t using Equation 5. The old cell state is multiplied by the current cell state and then determined to be forgotten by the forget gate. Next, add $i_t * \tilde{C}_t$ to form a new value [29].

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{5}$$

Where,

C_t	: Cell state
C_{t-1}	: Cell state from the previous cell

The final stage is the output gate. The output must first match the cell state. Begin by running a sigmoid layer that determines which parts of the cell state are output, using Equation 6. Then, pass the cell state through a tanh function (to constrain the value between -1 and 1) and multiply it by the output of the sigmoid gate, thus only outputting the parts that were selected, as indicated by Equation 7 [29].

$$O_t = \sigma(W_0 . [h_{t-1}, x_t] + b_0) \tag{6}$$

$$h_t = O_t * tanh\,(C_t) \tag{7}$$

Where,

O_t	: Output gate
W_o	: Weight value for output gate
b_o	: Bias value at output gate
h_t	: Current output value

This study also uses another architecture of LSTM, namely Bi-LSTM. Bidirectional LSTM, commonly known as Bi-LSTM, is an LSTM that can perform two-way learning, as shown in Figure 3. The BiLSTM network structure consists of two single LSTM layers stacked vertically, where one is for the forward pass and the other is for the backward pass [30]. A recurrent bidirectional convolutional layer is used, employing two hidden layers. These two hidden layers allow for the simultaneous iteration of the forward and backward directions [31].

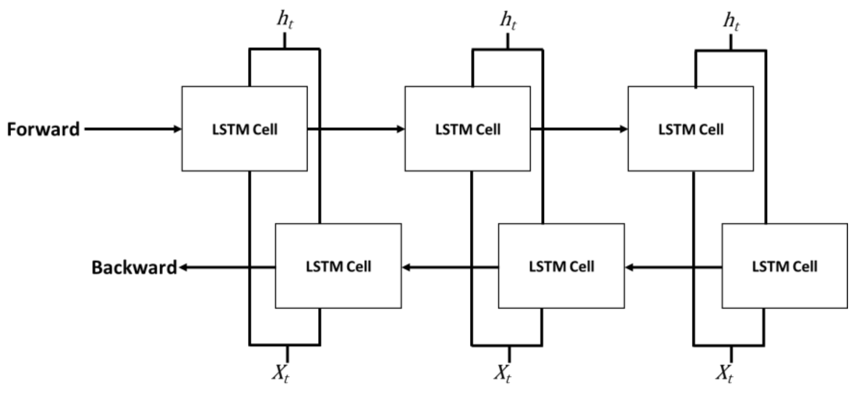

Figure 3. Bi-LSTM network layer

2.5 Data Classification

The model generated from the training process is used in the testing process. In this section, the model classifies data X that has been obtained from the data collection process using the username X. Data that has been successfully obtained through the crawling process using the X API enters the pre-processing stage with the aim of cleaning the data to ensure it does not contain unnecessary characters for the next process. The cleaned data is classified using the model built during the training process with word embedding and LSTM. From the classified data, the percentage of tweets indicating mental health disorders is calculated.

The current trial was conducted with 6,307 data points, containing 3,156 negative polarity data points and 3,151 neutral data points. Data with negative polarity, indicated as user data with a mental disorder, has the label "1", while positive data or normal user data has the label "0". The 6,307 data points were used as training data. In the classification process, 100 test data points were used, containing 50 data with the label "1" and 50 data points with the label "0". The experiment was conducted using sigmoid activation, Adam optimizer, and loss measurement using mean squared error, with an input size set to 300.

The first experiment was conducted to compare the performance of LSTM and Bidirectional LSTM. Word embedding uses the default embedding provided by LSTM. The first experiment also compared the effect of the text preprocessing techniques used. The second experiment was conducted using the word embedding method with the existing corpus dataset. The word embedding methods used were Word2Vec and FastText. This was followed by training the model using text preprocessing techniques and LSTM, which showed better performance based on the second experiment. In addition, experiments were also conducted using pre-trained word embeddings from Word2Vec, FastText, and GloVe with a size of 300 dimensions. From the second test, it was determined which word embedding method showed the best performance. Performance measurement results were carried out using precision, recall, accuracy, and F1-Score as evaluation criteria.

3. Results and Discussion

In this section, all the results of the experiments conducted using different techniques are displayed in several sections.

3.1 Experiments using LSTM and Bi-LSTM

The first experiment was conducted by applying several pre-processing techniques. In addition, we also tested two LSTM methods, namely unidirectional LSTM and bidirectional LSTM (Bi-LSTM). This study used hyperparameters commonly found in deep learning literature for text classification tasks. Specifically, we applied the sigmoid activation function, the Adam optimizer, and the mean squared error (MSE) as the loss function. The input size was set to 300, consistent with the default word embedding size commonly used in LSTM models [32]. The results of the experiment are shown in Table 3.

Table 3. Experimental Results using LSTM & Bi-LSTM

Pre-processing	LSTM			Bi-LSTM		
	7 stages	10 stages	15 stages	7 stages	10 stages	15 stages
Accuracy	73.00%	75.99%	77.99%	74.00%	76.99%	80.00%
Loss	20.11%	16.78%	15.71%	18.53%	15.87%	15.73%
Precission	71.69%	73.21%	72.58%	69.35%	73.68%	75.00%
Recall	76.00%	82.00%	90.00%	86.00%	84.00%	90.00%
F1 Score	73.78%	77.35%	80.35%	76.78%	78.50%	81.81%

Table 4 shows the text pre-processing results with 7 stages, 10 stages, and 15 stages. Text pre-processing with 7 stages can only remove punctuation, so it can be seen that the hashtags in the text are still there and only the hashtags are removed. In addition, the negation words also remain in their original form. In the 3rd text, the negation phrase "can't believe" remains unchanged, with only the existing stop words "can" and "not" removed. Text pre-processing with 10 stages can eliminate other unnecessary characters, such as hashtags and links. Text pre-processing with 15 stages is achieved by adding repeated character handling, spelling correction, and negation handling. Pre-processing with 7 and 10 stages cannot handle negation. In pre-processing with 15 stages, the negation phrase "can not believe" is changed to “disbelieve”, preserving its meaning. This technique has also been able to convert several emoticons into text form according to their meanings, such as the emoticon ":(" being changed to "frown/sad".

Table 4. Text Pre-processing Results

No	Text	7 stages	10 stages	15 stages
1	Today in Selfcare: beauty &amp; laughs Kung Fu Panda 3 #Wellness #joy #laughter #selfcare #therapist #philadelphia	today selfcare beauty amp laugh kung fu panda wellness joy laughter selfcare therapist philadelphia	today selfcare beauty laugh kung fu panda	today self care beauty laugh gunk panda

No	Text	7 stages	10 stages	15 stages
2	Depressed and lonely /: Stuck in a deep, never ending hole :(	depress lonely stick deep never end hole	depress lonely stuck deep never end hole	depress lonely stuck deep never end hole worn sad andry pout
3	Can't believe I've had 9 days off already and that I'm back to work tomorrow!	believe i days already i back work tomorrow	believe i days already i back work tomorrow	disbelieve i days already i back work tomorrow

The results show that pre-processing with 15 stages can produce cleaner and more meaningful data. Additionnally, the results show that Bi-LSTM has better performance than one-way LSTM. In the results of the LSTM experiment using 15-stage text preprocessing, an accuracy of 77.99% was achieved, while Bi-LSTM achieved an accuracy of 80%. The accuracy obtained did not show a significant difference. However, the precision and recall results of Bi-LSTM showed more balanced results overall. This can also be seen in the F1-Score, where LSTM achieved 80.35% and Bi-LSTM achieved 81.81%. The results obtained by Bi-LSTM are superior because it allows for a two-way training process, namely forward and backward, enabling Bi-LSTM to learn data in a more complex manner.

3.2 Experiment using Bi-LSTM with Word Embedding Method

The second experiment was conducted using Bi-LSTM with a 15-stage text pre-processing technique. In addition, this experiment was carried out with several word embedding methods, namely Word2Vec, FastText, and GloVe. The results of the experiment are shown in Table 5.

Table 5. Experimental Results using Bi-LSTM with Word Embedding & Pre-trained Word Embedding

	Word Embedding		Pre-trained Word Embedding		
	Word2Vec	FastText	Word2Vec	FastText	Glove
Accuracy	85.00%	87.00%	81.99%	86.00%	81.00%
Loss	11073%	14.21%	17.65%	10.43%	19.63%
Precission	80.70%	97.43%	75.00%	87.50%	73.13%
Recall	92.00%	76.00%	96.00%	84.00%	98.00%
F1 Score	85.98%	85.39%	84.21%	85.71%	83.76%

Pre-trained FastText shows better results than the others, with values that are close to each other. Word2Vec shows 85% accuracy, FastText shows 87% accuracy, and Pre-trained FastText shows 86% accuracy. However, the precision of 80.7% and recall of 92% shown by Word2Vec seem unbalanced, as the recall value is much better than the precision. The difference in recall and precision values indicates a lack of balance between the two.

FastText has the highest accuracy compared to both Word2Vec and FastText. One of the advantages of using FastText is its ability to convert words into a vector of numbers in words that are not in the dictionary, commonly known as out of vocabulary words. However, the precision of 97.43% and recall of 76% for FastText seem unbalanced. Thus, FastText is considered capable of classifying user data indicative of mental disorders effectively, but it is not as proficient at classifying normal user data. In the trial results with Pre-trained FastText, the precision of 87.5% and recall of 84% showed more balanced results.

The results of this research are compared with previous sentiment analysis studies that utilized LSTM and Bi-LSTM algorithms with varying insertion strategies and settings, as shown in Table 6.

Table 6. Comparison with Previous Research

Research	Algorithm	Pre-processing	Embeddings	Context	Best Accuracy	F1-Measure
Ardila et al. (2024)	Multi Layer Bi-LSTM	not explained	Glove	Identify sarcastic comments	95,41%	-
Sari et al. (2022)	RNN & LSTM	6 stages	not explained	Abusive comment detection	-	94%
Kholifah et al. (2020)	LSTM	4 stages	Feature extraction	Mental heath detection	70,89%	-
This research	Bi-LSTM	15 stages	Pre-trained FastText	Mental health analysis	86%	85,71%

Ardila et al. (2024) focus on sarcasm detection using Multi-Layer Bi-LSTM with GloVe embeddings, achieving the highest accuracy of 95.41%, while Sari et al. (2022) apply RNN & LSTM for abusive comment detection, incorporating a 6-stage preprocessing pipeline and attaining a 94% F1-measure. In contrast, Kholifah et al. (2020) address mental health detection using LSTM with basic feature extraction, yielding 70.89% accuracy. Building on this, the present research enhances mental health detection by employing Bi-LSTM, a 15-stage preprocessing pipeline, and Pre-trained FastText embeddings, resulting in an 86% accuracy—significantly surpassing Kholifah et al. (2020).

Bi-LSTM consistently outperforms LSTM because of its ability to learn bidirectional context. Multi-Layer Bi-LSTM used in sarcasm detection achieves the highest accuracy, indicating that deeper networks are advantageous when dealing with complex text structures. RNN-LSTM hybrid also performs well in abusive comment detection, demonstrating that hybrid models can enhance performance. This research confirms that Bi-LSTM improves mental health detection, surpassing Kholifah et al. (2020) by a wide margin (86% vs. 70.89%).

More preprocessing stages lead to better performance. The 15-stage preprocessing in this research significantly contributes to improving accuracy over Kholifah et al. (2020). Ardila et al. (2024) and Sari et al. (2022) do not explicitly mention preprocessing depth, making direct comparisons difficult, but the improvements in sentiment and sarcasm detection suggest that preprocessing plays a crucial role. Pre-trained FastText outperforms Word2Vec and traditional feature extraction methods in mental health detection, while GloVe embeddings achieve the highest performance in sarcasm detection, possibly due to their robust understanding of word co-occurrence. However, the lack of explicit embedding details in abusive comment detection makes it unclear how much embeddings contributed to their high accuracy in this area.

Compared to sarcasm and abusive comment detection, which achieve accuracies above 90%, mental health detection remains more challenging due to subtle emotional variations in user expressions. Sarcasm and hate speech often contain distinct linguistic markers, making them easier to detect. In contrast, mental health indicators are often nuanced, requiring more sophisticated context-aware models.

This research bridges the gap between mental health analysis and high-performing deep learning models by incorporating advanced word embeddings and comprehensive preprocessing techniques. The significant accuracy improvement over prior studies highlight the effectiveness of the proposed approach, proving its value in enhancing mental health disorder detection through social media mining.

4. Conclusion

From the results of the experiments and analysis, this research demonstrates significant improvements over previous mental health detection models by leveraging Bi-LSTM, comprehensive text preprocessing, and pre-trained embeddings, achieving an accuracy of 86%, which significantly surpasses the 70.89% accuracy observed in mental health disorder detection. The findings indicate that Bi-LSTM outperforms one-way LSTM, particularly in terms of precision-recall balance, due to its ability to process data in both forward and backward directions. However, the results also reveal that some word embedding methods, such as FastText, tend to produce unbalanced precision-recall values, highlighting the need for improved classification techniques.

Future research should explore the use of Multi-Layer Bi-LSTM for mental health detection, as seen in sarcasm detection which achieved 95.41% accuracy. Implementing a multi-layered approach could enhance the model's ability to capture complex linguistic patterns associated with mental health symptoms. Additionally, addressing the imbalance in precision-recall scores, such as the 97.43% precision versus 76% recall observed in FastText, is crucial to ensure that the model does not disproportionately misclassify normal user data. This can be mitigated through data balancing strategies, including oversampling minority classes or fine-tuning classification thresholds.

Furthermore, the size and diversity of the dataset should be carefully considered in future studies, as a larger dataset can improve model generalization and reduce overfitting, while a more diverse dataset ensures robustness across different linguistic expressions of mental health symptoms. Expanding the dataset would help the model better differentiate between users with and without mental health disorders, reducing the risk of biased predictions. Lastly, integrating hybrid architectures, such as CNN+Bi-LSTM, or incorporating attention mechanisms could further enhance model performance by capturing deeper contextual relationships within text data. This is expected to increase the accuracy and sensitivity in detecting mental disorders, as well as expand the benefits of the system to more individuals who need psychological support.

Acknowledgement

The authors would like to express gratitude to all parties who have provided support in this research. Special thanks are extended to the research supervisor for valuable guidance and input, as well as to the institution that provided the necessary facilities and resources. The authors would also like to thank their family and colleagues who have provided motivation and encouragement during the research process. Hopefully the results of this research can contribute to the development of science and problem solving in the field of mental health.

References

[1] WHO, “Suicide,” 2024.

[2] N. A. Wirawan, “Angka Kasus Bunuh Diri di Indonesia Meningkat 60% dalam 5 Tahun Terakhir,” 2024.

[3] K. K. R. Biro Komunikasi dan Pelayanan Publik, “Cegah Bunuh Diri, Kemenkes Ajak Remaja Bicara Soal Kesehatan Mental – Sehat Negeriku,” 2024.

[4] E. Velthorst *et al.*, “The impact of loneliness and social relationship dissatisfaction on clinical and functional outcomes in Dutch mental health service users,” *Psychiatry Research*, vol. 342. 2024. https://doi.org/10.1016/j.psychres.2024.116242

[5] B. Kholifah, I. Syarif, and T. Badriyah, “Mental Disorder Detection via Social Media Mining using Deep Learning,” *Kinet. Game Technol. Inf. Syst. Comput. Network, Comput. Electron. Control*, vol. 4, pp. 309–316, 2020. https://doi.org/10.22219/kinetik.v5i4.1120

[6] M. L. Zou, M. X. Li, and V. Cho, “Depression and disclosure behavior via social media: A study of university students in China,” *Heliyon*, vol. 6, no. 2, p. e03368, 2020. https://doi.org/10.1016/j.heliyon.2020.e03368

[7] I. Syarif, N. Ningtias, and T. Badriyah, “Study on Mental Disorder Detection via Social Media Mining,” *2019 4th Int. Conf. Comput. Commun. Secur.*, pp. 1–6, 2019. https://doi.org/10.1109/CCCS.2019.8888096

[8] C. Ouni, E. Benmohamed, and H. Ltifi, “Deep learning-based Soft word embedding approach for sentiment Deep learning-based Soft word embedding approach for sentiment analysis analysis,” *Procedia Comput. Sci.*, vol. 246, pp. 1355–1364, 2024. https://doi.org/10.1016/j.procs.2024.09.720

[9] R. Phukan, P. J. Goutom, and N. Baruah, “Assamese Fake News Detection: A Comprehensive Exploration of LSTM and Bi-LSTM Techniques,” *Procedia Comput. Sci.*, vol. 235, pp. 2167–2177, 2024. https://doi.org/10.1016/j.procs.2024.04.205

[10] J. Zhao, D. Zeng, Y. Xiao, L. Che, and M. Wang, “User personality prediction based on topic preference and sentiment analysis using LSTM model,” *Pattern Recognition Letters*, vol. 138. pp. 397–402, 2020. https://doi.org/10.1016/j.patrec.2020.07.035

[11] T. I. Sari, Z. N. Ardilla, N. Hayatin, and R. Maskat, “Abusive comment identification on Indonesian social media data using hybrid deep learning,” *IAES Int. J. Artif. Intell.*, vol. 11, no. 3, pp. 895–904, 2022. https://doi.org/10.11591/ijai.v11.i3.pp895-904

[12] V. Kishore and M. Kumar, “Enhanced Multimodal Fake News Detection with Optimal Feature Fusion and Modified Bi-LSTM Architecture,” *Cybernetics and Systems*. 2023. https://doi.org/10.1080/01969722.2023.2175155

[13] Z. N. Ardilla, T. I. Sari, N. Hayatin, and C. Fatichah, “Sarcasm detection on news headline using multilayer bidirectional-LSTM with glove embeddings,” *AIP Conf. Proc.*, vol. 2927, no. 1, p. 60035, Mar. 2024. https://doi.org/10.1063/5.0192254

[14] U. Naseem, I. Razzak, and P. W. Eklund, “A survey of pre-processing techniques to improve short-text quality: a case study on hate speech detection on twitter,” *Multimedia Tools and Applications*, vol. 80, no. 28–29. pp. 35239–35266, 2021. https://doi.org/10.1007/s11042-020-10082-6

[15] M. Siino, I. Tinnirello, and M. La Cascia, “Is text preprocessing still worth the time? A comparative survey on the influence of popular preprocessing methods on Transformers and traditional classifiers,” *Inf. Syst.*, vol. 121, no. March 2023, p. 102342, 2024. https://doi.org/10.1016/j.is.2023.102342

[16] N. H. Yahya and H. Abdul Rahim, “Linguistic markers of depression: Insights from english-language tweets before and during the COVID-19 pandemic,” *Lang. Heal.*, vol. 1, no. 2, pp. 36–50, 2023. https://doi.org/10.1016/j.laheal.2023.10.001

[17] A. Daud, D. Irwanto, M. Said, M. Mubyl, and Mustamin, “Identification Of Chatbot Usage In Online Store Services Using Natural Language Processing Methods,” *Adv. Sustain. Sci. Eng. Technol.*, vol. 6, no. 2, pp. 1–9, 2024. https://doi.org/10.26877/asset.v6i2.18309

[18] N. Garg and K. Sharma, “Text pre-processing of multilingual for sentiment analysis based on social network data,” *Int. J. Electr. Comput. Eng.*, vol. 12, no. 1, pp. 776–784, 2022. https://doi.org/10.11591/ijece.v12i1.pp776-784

[19] A. Jabbar, S. Iqbal, M. I. Tamimy, A. Rehman, S. A. Bahaj, and T. Saba, “An Analytical Analysis of Text Stemming Methodologies in Information Retrieval and Natural Language Processing Systems,” *IEEE Access*, vol. 11, no. October, pp. 133681–133702, 2023. https://doi.org/10.1109/ACCESS.2023.3332710

[20] S. Biradar, G. T. Raju, and K. M. Divakar, “Negation Handling and Domain Generalization in Sentiment Analysis using Machine Learning Models,” *2024 International Conference on Knowledge Engineering and Communication Systems, ICKECS 2024*. 2024. https://doi.org/10.1109/ICKECS61492.2024.10616885

[21] P. K. Singh and S. Paul, “Deep Learning Approach for Negation Handling in Sentiment Analysis,” *IEEE Access*, vol. 9, pp. 102579–102592, 2021. https://doi.org/10.1109/ACCESS.2021.3095412

[22] D. J. Ladani and N. P. Desai, “Stopword Identification and Removal Techniques on TC and IR applications: A Survey,” *2020 6th International Conference on Advanced Computing and Communication Systems, ICACCS 2020*. pp. 466–472, 2020. https://doi.org/10.1109/ICACCS48705.2020.9074166

[23] Z. H. Kilimci and S. Akyokus, “The Evaluation of Word Embedding Models and Deep Learning Algorithms for Turkish Text Classification,” *UBMK 2019 - Proceedings, 4th Int. Conf. Comput. Sci. Eng.*, pp. 548–553, 2019. https://doi.org/10.1109/UBMK.2019.8907027

[24] M. Mars, “From Word Embeddings to Pre-Trained Language Models: A State-of-the-Art Walkthrough,” *Appl. Sci.*, vol. 12, no. 17, 2022. https://doi.org/10.3390/app12178805

[25] S. Sivakumar, L. S. Videla, T. Rajesh Kumar, J. Nagaraj, S. Itnal, and D. Haritha, “Review on Word2Vec Word Embedding Neural Net,” *Proc. - Int. Conf. Smart Electron. Commun. ICOSEC 2020*, no. Icosec, pp. 282–290, 2020. https://doi.org/10.1109/ICOSEC49089.2020.9215319

[26] T. Yao, Z. Zhai, and B. Gao, “Text Classification Model Based on fastText,” *Proc. 2020 IEEE Int. Conf. Artif. Intell. Inf. Syst. ICAIIS 2020*, pp. 154–157, 2020. https://doi.org/10.1109/ICAIIS49377.2020.9194939

[27] P. J. Worth, “Word Embeddings and Semantic Spaces in Natural Language Processing,” *International Journal of Intelligence Science*, vol. 13, no. 01. pp. 1–21, 2023. https://doi.org/10.4236/ijis.2023.131001

[28] C. O. Ewald and Y. Li, “The role of news sentiment in salmon price prediction using deep learning,” *J. Commod. Mark.*, vol. 36, 2024. https://doi.org/10.1016/j.jcomm.2024.100438

[29] R. Kizito, P. Scruggs, X. Li, M. Devinney, J. Jansen, and R. Kress, “Long Short-Term Memory Networks for Facility Infrastructure Failure and Remaining Useful Life Prediction,” *IEEE Access*, vol. 9, pp. 67585–67594, 2021. https://doi.org/10.1109/ACCESS.2021.3077192

[30] S. Shan *et al.*, “A Deep Learning Model with Attention-BiLSTM Networks Combining XGBoost Residual Correction for Short-Term Water Demand Forecast,” 2022.

[31] B. H. Nayef, S. N. H. S. Abdullah, R. Sulaiman, and A. M. Saeed, “Text Extraction with Optimal Bi-LSTM,” *Comput. Mater. Contin.*, vol. 76, no. 3, pp. 3549–3567, 2023. https://doi.org/10.32604/cmc.2023.039528

[32] M. A. K. Raiaan *et al.*, “A systematic review of hyperparameter optimization techniques in Convolutional Neural Networks,” *Decis. Anal. J.*, vol. 11, no. March, p. 100470, 2024. https://doi.org/10.1016/j.dajour.2024.100470