Jurnal Teknik Informatika (JUTIF)
Vol. 5, No. 6, December 2024, pp. 1759-1767

DOI: https://doi.org/10.52436/1.jutif.2024.5.6.2557
p-ISSN: 2723-3863
e-ISSN: 2723-3871

HATE SPEECH DETECTION USING GLOVE WORD EMBEDDING AND GATED RECURRENT UNIT

Aulia Riefqi Ardana[*1], Yuliant Sibaroni[2]

[1,2]Informatics, Informatics Faculty, Telkom University, Indonesia
Email: [1]auliardana@student.telkomuniversity.ac.id, [2]yuliant@telkomuniversity.ac.id

(Article received: July 23, 2024; Revision: August 11, 2024; published: December 29, 2024)

Abstract

Social media has become a tool that makes it easier for people to exchange information. The freedom to share information has opened the door for increased incidents of hate speech on social media. Hate speech detection is an interesting topic because with the increasing use of social media, hate speech can quickly spread and trigger significant negative impacts, discrimination, and social conflict. This research aims to see the effect of GRU method, GloVe word embedding and word modifier algorithm in detecting hate speech. GRU and GloVe are used in this research for the hate speech detection system, where deep learning with a Gated Recurrent Unit (GRU) and Word Embedding with the Global Vector model (GloVe) converts words in text into numerical vectors that represent the meaning and context of the words. GRU is chosen due to its ability to capture long-term dependencies in textual data with higher computational efficiency compared to Long Short-Term Memory (LSTM). Gated Recurrent Unit (GRU) model processes the sequence of words to understand the sentence structure. GRU model processes the sequence of words to understand the sentence structure. The evaluation results for the classification of hate speech using GRU and GloVe are 90.7% accuracy and 91% F1 score. With the combination of informal word modifier algorithms there is an increase with a value of 92.8% F1 and 92.4% accuracy. in conclusion, the use of informal word modifier algorithms can increase the evaluation value in detecting hate speech.

Keywords: *classification, GloVe, gated recurrent unit, hatespeech.*

1. INTRODUCTION

Hate speech is a phenomenon that often occurs on various social media platforms. Hate speech can trigger divisions, misunderstandings, and even acts of violence between individuals or groups, especially because of the prejudice it causes. Social media allows users to share opinions and information freely. [1]. Social media is also useful for obtaining data to see people's perceptions of a topic that is circulating. For example, a topic that is quite popular on twitter regarding the Russian and Ukrainian wars, then covid-19 which spread throughout the world. During the Covid-19 pandemic, social media has been helpful for policymakers to strategize by understanding the responses given by the public. [2], [3]. Social media can also illustrate the rating of a product or service. This is useful for companies to evaluate their products based on public opinion[4], [5], [6]. However, this freedom of speech and lax regulations have increased the spread of hate speech [7], which can have a significant negative impact on certain individuals and groups of people

The research adopts BiLSTM and Word2Vec models with continuous bag of words (CBOW) architecture. There are also studies that combine global vector algorithm with deep belief network. The use of GloVe in previous studies showed good accuracy; this study combined long short-term memory (LSTM) with GloVe word embedding for hate speech classifier, with accuracy reaching 81.5% in one-layer LSTM and 80.9% in two-layer LSTM. However, there are difficulties in identifying slang words and local languages. In addition, the model also misinterpreted some words in the context of animal names, rather than as hateful statements. Another study that combined fastText with GRU in classifying offensive and inoffensive texts showed a fairly good accuracy of 84%. The drawback of this model is that it does not fully capture the context of the sentence, and its performance is highly dependent on proper hyperparameter tuning, which can take significant time and computational resources. [8], [9], [10], [11], [12].

Although many studies have been conducted, there is still room for improvement in hate speech detection. In this study, the use of gated recurrent unit (GRU) and word embedding global vector (GloVe) methods is expected to provide good accuracy in hate speech detection. GRU is a type of recurrent neural network architecture designed to overcome several problems in traditional recurrent neural networks, especially problems related to vanishing gradient and exploding gradient. GRU introduces a “gate” structure to control the flow of information in and out of the hidden units which

allows the model to better retain which relevant information to retain or ignore[13], [14]. And GloVe is a word representation learning model that combines the strengths of count-based and predictive methods. It uses global word co-occurrence statistics to generate word vectors that capture meaning and semantic relationships. [15].

Based on previous research with the combination of fasttext-GRU and GloVe-LSTM, This research focuses on the use of deep learning methods GRU, GloVe as word embedding and implementing an informal word converter into a formal word that is expected to have better performance in detecting hate speech.

2. RESEARCH METHOD

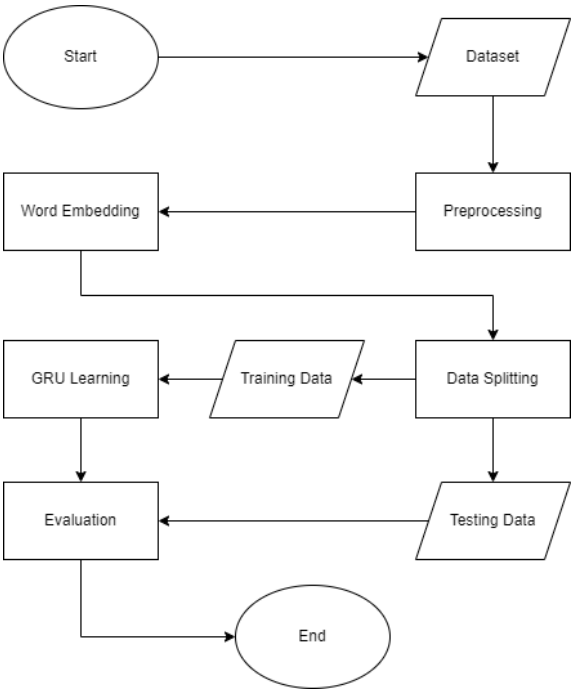

Figure 1. System Flow.

The system built for hate speech detection as shown in Figure 1, starting from dataset collection to evaluation is as follows.

2.1. Dataset

This research uses a dataset that has been labeled with 10291 lines of data. Of these, 5388 records contain hate speech and 4903 records do not contain hate speech. This research detects hate speech in English, so the dataset uses English. This dataset combines two data sources, there are [16] and [17], to increasing variation and reducing data imbalance.

2.2. Preprocessing

The dataset that has been collected is subjected to data cleaning, aiming to improve quality, reduce noise and data relevance [18]. These are several stages in the preprocessing technique.

1. Case Folding, this process converts all existing capital letters into lowercase letters [19].
2. Data Cleaning, this process of deleting unnecessary data [20].
3. Lemmatization, this process converts words into root words using WordNet.
4. Spell Correction, this process converts words that have unnatural reduplication of letters into their root word.[21]
5. Informal to Formal Words, This process converts informal words into standardized words.
6. Tokenizing, this process breaks down text into words or what are called tokens.

2.3. Word Embedding

This stage converts each word into a number vector using the global vector (GloVe) method. GloVe is a model developed to improve the learning of word representation in vector space. This model aims to capture patterns of relationships between words. This allows the model to work very well in word analogy and similarity between words[15].

This step converts each word in the dataset into a number vector. This number vector contains the semantic and syntactic information of the words. This research uses pretrained GloVe with a combined corpus from wikipedia and gigaword. This pretrained GloVe has 42 billion tokens and has 50 dimensions[22]. This dimension serves to determine how detailed information can be captured by a word vector.

2.4. Data Splitting

Divides the dataset for training and testing with a ratio of 80:20. Training data is utilized to train the model, whereas testing data is employed to evaluate the performance of the developed model.

2.5. Gated Recurrent Unit

Gated Recurrent Unit (GRU) is a type of deep learning used to process sequence data. GRU is a simpler variant of LSTM. GRU employs two types of gates: reset gates and update gates. These gates manage the flow of information and preserve essential context, achieving this without the need for the more complex structure found in LSTM.[23].

Using a Gated Recurrent Unit (GRU) architecture to detect hate speech works by receiving text that has been converted into a vector of numbers. This text then passes through a series of GRU layers to capture temporal and context information from the word sequence. The GRU uses two main gates, reset gate and update gate, to regulate the flow of information and retain or forget relevant information from the previous hidden state. After going through the GRU layers, the result is a

comprehensive contextual representation of the text[24].

2.6. Evaluation

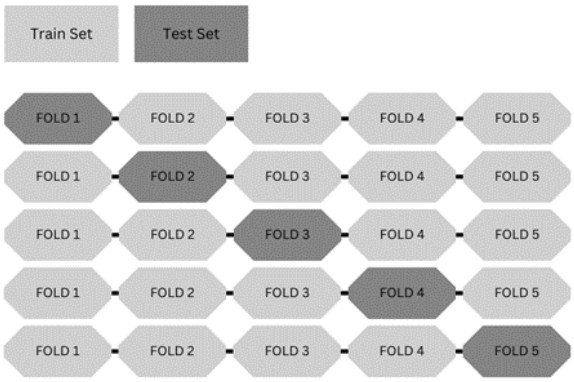

Figure 2. 5-Fold cross validation[25]

The model is evaluated using k-Fold Cross Validation and a confusion matrix to calculate accuracy, precision, recall, and F1-score values. In the k-Fold Cross Validation method, the dataset is split into k equally-sized subsets or folds as shown in Figure 2. During each of the k iterations, the model is trained on k-1 folds and tested on the remaining fold. This process ensures that each fold is used as a test set exactly once. Common values for k are 5 and 10. When k is very small, the results can be sensitive to the choice of folds, whereas a very large k increases computational costs. The overall model accuracy is then determined by averaging the accuracy scores from each iteration.[26]

Model performance is measured using accuracy, precision, recall, and F1-score calculated from the confusion matrix. Accuracy as shown in equation 1 aims to measure the correct predictions out of all predictions made by the model. Precision measures how much of what is classified as positive is actually positive as shown in equation 2. Recall assesses how effectively the model identifies all positive instances , the calculation formula is shown in equation 3. The F1 score as shown in equation 4 is a metric that merges precision and recall to provide a comprehensive evaluation of the classification model's performance. The following is the calculation of the evaluation value[27].

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F1-score = \frac{2\ x\ Precision\ x\ Recall}{Precision+Recall} \quad (4)$$

3. RESULT AND DISCUSSION

3.1. Dataset

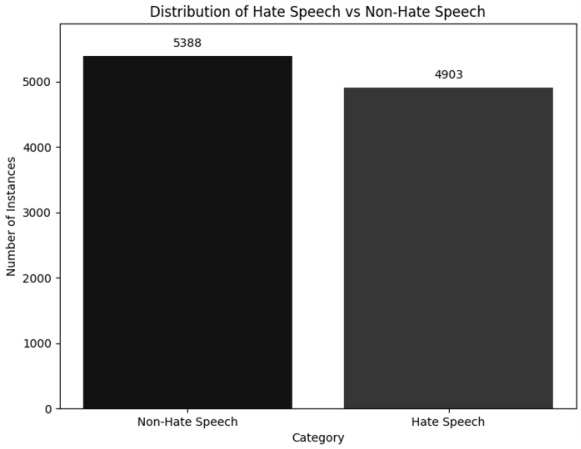

Figure 3. Data distribution

The dataset consists of 10291 rows that have 2 columns, namely the text column and the label column. The data consists of 5388 non- hate speech and 5388 hate speech as shown in Figure 3. The label column only contains between 2 values, namely 0 and 1. Label 1 means that the text contains hate speech, while 0 indicates that the text does not contain hate speech. An example dataset will be shown in the table below.

Table 1. Sample dataset

Label	Text
1	”man you are such a dickhead you half caste piece of shit”
1	” baby boomers are wankers and ruined everything”
1	”@user black professor makes assumptions about an entire race whilst speaking for entire race. next week the jews! #nazi”
0	” to all of the dad's out there...happy father's day! #fathersday #dads #thankyou ”
0	”The weather is so nice today, hope everyone has a great day!”
0	” So happy to spend the weekend with family and friends”

3.2. Preprocessing

This stage is done to reduce noise and improve model performance. There are several stages in preparing data before it is used for training. In the table 2 are the stages in data preprocessing.

Table 2. Preprocessing data

Proses	Text
Original Text	RT @user123: This is the worst thing i have ever read. People like youuu should go back where u came from! #disgusted #hatespeech
Case Folding	rt @user123: this is the worst thing i have ever read. people like youuu should go back where u came from! #disgusted #hatespeech
Data Cleaning	rt this is the worst thing i have ever read people like youuu should go back where u came from disgusted hatespeech
Lemmatization	rt this be the bad thing i have ever read people like youuu should go back where u come from disgust hatespeech
Word Normalization	rt this be the bad thing i have ever read people like you should go back where u come from disgust hatespeech
Informal word converter	rt this be the bad thing i have ever read people like you should go back where you come from disgust hatespeech
Tokenizing	['rt', 'this', 'be', 'the', 'bad', 'thing', 'i', 'have', 'ever', 'read', 'people', 'like', 'you', 'should', 'go', 'back', 'where', 'you', 'come', 'from', 'disgust', 'hatespeech']

```
# Tokenization
tokens = tokenizer.tokenize(s)
lowercased_tokens = [t.lower() for t in
tokens]
lemmatized_tokens = [lemmatizer.lemmatize
(t) for t in lowercased_tokens]
# Normalize words
normalized_tokens = [normalize_word(t) for
t in lemmatized_tokens]
```

Figure 4. Preprocessing phase

Figure 4 shows the data preprocessing stage for hate speech detection, several libraries are used to prepare the text to be cleaner and more structured. Library re is used for data cleaning, removing certain characters or patterns, and normalizing words by removing repeated characters. The lemmatization process is done using nltk to convert words into their basic form, while nltk is also used for text tokenization. All these stages aim to reduce noise and improve the performance of the model while training.

3.2.1. Informal Word Converter

This step converts informal words into formal words or words that are more common, usually called text normalization. The first step is to create a dictionary that contains two columns, namely informal and formal columns, in this dictionary there are 4556 rows of data with examples as in table 3 below.

Table 3. informal words dictionary

Informal	formal
u	you
y'all	you all
acab	all cops are bastard
bro	brother
gimme	give me

The informal converter is used when a word token cannot be found in the pretrained glove dictionary. In such cases, the word is first converted using the word converter dictionary before being processed further. The stages of converting informal words into formal words use a simple algorithm, which loops each word that has been preprocessed then checks the informal words in the dataset and dictionary, then converts them into formal words.

```
# Convert informal words to formal if not present
final_tokens = []
for token in normalized_tokens:
    if token not in words:
        if token in informal_to_formal:
            token = informal_to_formal[token]

    final_tokens.extend(token.split())
```

Figure 5. Informal words converter algorithm

Natural Language Processing (NLP) techniques for converting informal words into formal ones are important in handling text from platforms that often use nonstandard language, such as social media or online forums. Embedding models such as GloVe, which represent words as vectors with fixed dimensions, are often trained on formal text corpus, thus failing to cover informal words. To overcome this limitation, a common approach is to use a dictionary or mapping from informal words to formal words. The given code as shown in Figure 5 implements this approach by checking each normalized text token against the list of words available in the GloVe embedding.. If the token is not found, the code looks for its formal equivalent in the informal_to_formal dictionary and replaces it if it exists. This approach is effective for increasing the coverage of the embedding and allowing NLP models to better understand the text, especially in informal language contexts.

3.3. GloVe Word Embedding

Converting each word that has been previously converted into a token into a vector using the global vector method. The process of converting tokens into vectors is carried out based on the existing pretrained glove model. The contents of the pretrained glove can be seen in the table below.

Table 4. Pretrained GloVe

Word	Dimension 1	Dimension 2	Dimension 3	Dimension 4
the	0.418	0.249	-0.412	0.121
,	0.013	0.236	-0.168	0.409
.	0.151	0.301	-0.167	0.176
of	0.708	0.570	-0.471	0.180
to	0.680	-0.039	0.301	-0.177
the	0.013	0.236	-0.168	0.409

Each dimension of the embedding vector represents a particular feature or aspect of the word meaning, although the specific interpretation of each dimension is not always immediately obvious. These dimensions capture various semantic relationships between words and are the result of a complex process of learning from text data. For example, certain dimensions may relate to word types (nouns, verbs) or contextual concepts . Vector embedding enables comparisons between words through vector operations, so that words that appear frequently in the same context will have vectors that are more similar to each other. Vector embedding helps natural language processing models capture the meaning and semantic relationships between words more effectively.

Datasets that have been preprocessed in the previous stage are then weighted using pretrained glove. An example of text that is weighted using GloVe can be seen in the table below.

Table 5. GloVe word embedding

Text	Word	Vector
i hate fucking tosser faggot	i	0.118 0.152 -0.082 -0.741 0.759 -0.483 -0.310 0.514 -0.987 0.0006 -0.150 0.837 -1.079 -0.514 1.318 0.620 0.137 0.471 -0.072 -0.726 -0.741 0.752 0.881 0.295 1.354 -2.570 -1.352 0.458 1.006 -1.185 3.473 0.778 -0.729 0.251 -0.261 -0.346 0.558 0.750 0.498 -0.268 -0.002 -0.018 -0.280 0.553 0.037 0.185 -0.150 -0.575 -0.266 0.921
	hate	-0.852 -0.147 0.049 -0.787 0.194 0.489 0.343 -0.615 -0.372 0.771 -0.719 -0.129 -0.409 -0.653 0.949 -0.498 0.254 0.049 0.189 -0.149 -0.276 0.320 0.858 1.017 0.222 -1.823 -0.693 -0.018 1.454 -1.103 1.526 0.698 -0.590 -1.047 -1.2981 -0.472 -0.319 -1.348 -0.193 0.182 0.043 -0.172 0.167 0.661 0.716 0.169 -0.347 -0.151 -0.051 0.297
	fucking	-0.682 -0.703 0.069 -1.056 0.134 -0.341 0.134 0.121 -0.160 0.593 -0.439 0.262 -0.113 0.232 0.588 -0.191 0.893 0.778 0.033 0.270 -0.687 -0.060 0.282 0.664 0.801 -0.142 -1.757 0.529 1.130 -0.975 -0.071 0.299 0.175 1.156 -1.099 0.451 0.706 -0.741 0.327 -0.273 0.122 -0.650 -0.732 0.898 0.086 -0.379 -0.297 -0.269 0.443 0.191
	tosser	-0.949 -0.518 0.133 -0.234 -0.099 0.012 1.004 -0.415 0.548 0.635 -0.026 -0.172 -0.141 0.777 -0.621 -0.008 0.441 0.273 -0.043 -0.159 -0.282 -0.338 0.009 -0.340 -0.246 -0.089 -0.479 0.890 0.380 -0.409 -1.415 0.070 -0.123 1.297 0.057 0.464 0.296 0.392 -0.602 0.139 0.456 0.262 -0.142 0.074 0.809 -0.616 0.127 0.099 0.703 -0.444
	faggots	-0.060 -0.526 -0.368 -0.436 -0.476 0.390 0.844 -0.416 -0.147 0.337 -0.537 0.744 0.079 0.276 0.691 -0.542 0.784 0.537 0.503 -0.225 0.080 -0.131 0.446 0.540 0.084 0.841 -0.379 0.288 0.741 -0.737 -0.963 0.309 -0.149 0.475 -0.579 0.634 0.017 -0.795 0.052 0.184 -0.178 -0.393 0.457 0.509 -0.174 -0.192 0.026 -0.020 0.350 -0.120

3.4. Classification Using GRU

The model architecture employs a single GRU layer with 64 units, followed by a Dropout layer with a probability of 0.3 to mitigate overfitting during training. This configuration ensures that the model generalizes well to unseen data by randomly dropping units from the network, preventing it from becoming too reliant on any particular neuron. The final layer of the model is a Dense layer with a sigmoid activation function, which outputs a probability score that indicates the likelihood of the text containing hate speech.

The model is trained using the Adam optimizer combined with the binary cross-entropy loss function. Adam is chosen for its ability to adaptively adjust learning rates, enhancing the training process's efficiency and stability. The binary cross-entropy loss function measures the discrepancy between the predicted probabilities and the actual class labels, guiding the optimization of the model weights. Training continues until the model achieves satisfactory performance in detecting hate speech, as indicated by its accuracy on the validation set.

Table 6. GRU parameter

Parameter	Value
Batch Size	16
epoch	10
optimizer	Adam
Loss	Binary Crossentropy

As shown in Table 6, the use of batch size 16, epoch 10, Adam optimizer, and binary crossentropy loss in the hate speech detection program with GRU and GloVe is based on several considerations. Batch size 16 allows for memory efficiency and more

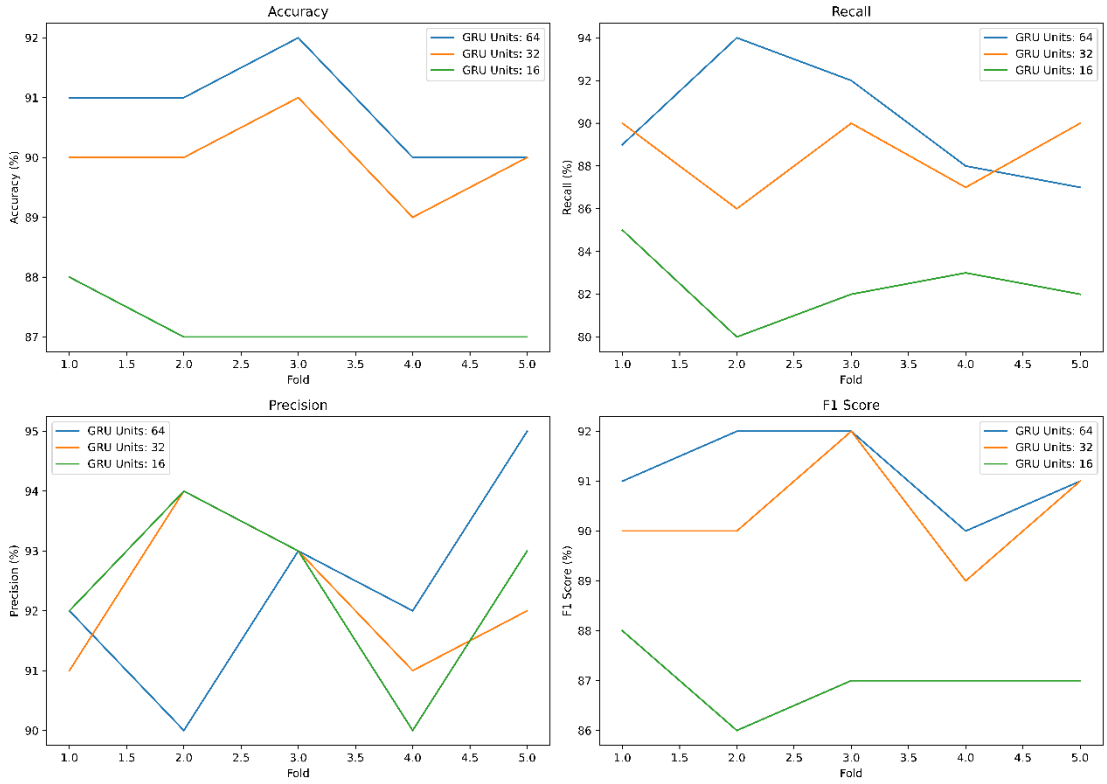

Figure 6. Comparison of GRU Model Performance Metrics with Various Units

frequent gradient updates, while 10 epochs were chosen because there is no significant improvement, saving computing time and resources. Adam optimizer was chosen for its ability to automatically adjust the learning rate for each parameter during training, combining the advantages of momentum and RMSProp. This allows for faster and more stable convergence, and improves model training efficiency. Binary crossentropy loss is used because it performs binary classification, so this loss is suitable for measuring how well the predicted probability distribution matches the target class .

```
model = Sequential([
    layers.Input(shape=(57, 50)),
    layers.GRU(64),
    layers.Dropout(0.3),
    layers.Dense(1, activation='sigmoid')
])
```

Figure 7. Sequential model for binary classification

The neural network model in figure 7 uses Keras Sequential library for binary classification tasks. The model accepts as input a sequence of data with dimensions (57, 50), where 57 is the number of time steps and 50 is the features per time step. The model consists of three main layers: first, a GRU (Gated Recurrent Unit) layer with 64 units, which captures temporal dependencies in the data. Then, a Dropout layer with a 30% dropout rate is used to prevent overfitting by randomly ignoring 30% of the units in the previous layer during training. Finally, a Dense layer with 1 unit and sigmoid activation produces an output that is the probability of the target class, which is suitable for binary

3.5. Evaluation

The evaluation applies 5-fold cross-validation to gauge the performance of the constructed model. This method calculates accuracy, recall, precision, and F1 score to measure effectiveness. Several scenarios were tested. The results are presented in Figure 6 for comparison, illustrating the performance metrics of both models.

In Figure 6, an attempt was made to change the number of units in the GRU model as part of the experiment. Although at this stage the model has not implemented informal to formal word conversion, the evaluation results provide important insights. In the three experiments conducted, GRU with 64 units showed the most optimal evaluation performance compared to models using a larger number of units, such as 128 units. Although increasing the number of units above 64 did not show any significant difference in the evaluation results, the use of larger units could potentially increase the computational load significantly. Therefore, 64 units was chosen as the best configuration to maintain a balance between performance and computational efficiency. This selection is based on the consideration to optimize the results while keeping in mind the limited

computational resources, which is crucial in the development of efficient and scalable models..

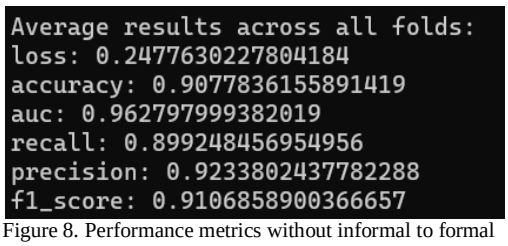

Figure 8. Performance metrics without informal to formal transformation

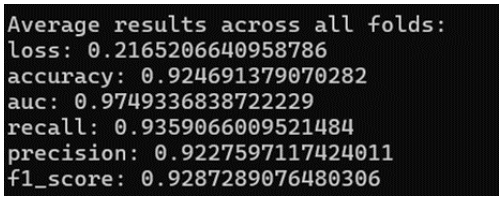

Figure 9. Performance metrics with informal to formal transformation

Figure 8 shows the average evaluation score of the model before applying the informal-to-formal word conversion technique. Meanwhile, Figure 8 displays the evaluation results after the technique was applied. The results in Figure 9 are actually consistent with the data presented in figure 6. However, it is important to note that the values in figure 6 have been converted to whole numbers to simplify the presentation of the data, thus providing a more concise view. This rounding does not compromise the accuracy of the results obtained, but provides a format that is easier to understand. A comparison between Figure 8 and Figure 9 provides a clearer picture of the positive impact of converting informal to formal words on the overall performance of the model. From this, it can be seen that the performance improvement after the modifications were made improved the accuracy of the model.This study evaluates the performance of the model with two scenarios: without informal to formal word modifiers and with informal to formal word modifiers. In the scenario without informal to formal word modifiers, the model showed an average accuracy of 90.7%, which means that 90.7% of all predictions made by the model were correct. The average recall of 89.9% indicates that the model was able to detect 89.9% of all true positive instances. Precision of 92.3% indicates that out of all the predicted positive instances, 92.3% of them were actually positive. F1-score, which is the harmonization of precision and recall, stands at 91%.

In scenarios where informal to formal word modifiers were applied, the model's performance improved. The average accuracy climbed to 92.4%, underscoring the model's enhanced ability to predict accurately using formal language. While recall at 93.5%, precision at 92.2%. The F1-score rose to 92.8%, demonstrating an overall improvement in the model's performance when formal language converter was employed.

These two scenarios shows that the use of informal to formal word modifiers has a positive impact on model performance. The 1.7% increase in accuracy from 90.7% to 92.4% indicates that the model can make more precise predictions with formal words. The increase in precision indicates that the model's positive predictions become more accurate, reducing the number of false positives.

This research proves that converting informal words into formal words can improve the performance value of the model. With higher accuracy and precision, the model becomes more reliable in the detection tasks tested. This confirms the importance of considering the use of formal words in model development to improve prediction accuracy and precision.

4. DISCUSSION

Table 7. Model Accuracy Results

Model	Accuracy
IndoBerTweet + BiLSTM	93.7%
CNN + FastText	80%
Deep Belief Network + GloVe	84.38%
GRU + GloVe + Informal words modifier	92.4%

As shown in table 8. IndoBERTweet combined with BiLSTM to improve text classification performance. The model was trained using a public dataset from Twitter that has been labeled as hate speech and not hate speech. The evaluation results show an accuracy of 93.7%[8].

There are various studies that have employed different methodologies to enhance accuracy in classification tasks. One such study used Convolutional Neural Network (CNN) combined with FastText word embedding. By employing the K-Fold Cross Validation process and determining the optimal dropout value, this system achieved an accuracy rate of 80%[9].

Another research took a different approach by developing a system based on the Deep Belief Network method, utilizing GloVe features to improve accuracy before the classification phase. The findings from this study showed an accuracy rate of 84.38%, demonstrating the potential of combining Deep Belief Networks with GloVe word embeddings for better performance[11].

In the study being discussed, the proposed detection system integrates Gated Recurrent Unit (GRU) with Word Embedding using the GloVe model. The evaluation of the classification process using GRU and GloVe resulted in an accuracy of 90.7% and an F1 score of 91%. Furthermore, the inclusion of an informal word modifier algorithm led to an increase in accuracy to 92.4% and an F1 score to 92.8%, highlighting the effectiveness of this approach in improving detection accuracy.

From the comparison of previous research, it can be seen that various deep learning methods such as IndoBERTweet with BiLSTM, CNN with FastText, Deep Belief Network with GloVe, and GRU with GloVe as well as informal word modifier algorithms are used to detect hate speech on social media. Each method has advantages and varying accuracy results, showing that there are various effective approaches to address the problem of hate speech on social media. This comparison also shows that using the right combination of models and techniques can significantly improve the accuracy of hate speech detection.

In order to improve better classification, it is recommended to make modifications to the dictionary that will be used to convert informal words into formal ones. Add informal language vocabulary to the dictionary, so that the model can work better.

5. CONCLUSION

This research uses Gated Recurrent Unit (GRU) and GloVe as word embedding to classify hate speech. Datasets that have been labeled are obtained from two different sources then put together and taken as much as 10292 data. This dataset has two categories, namely categories containing hate speech with label 1 and not containing hate speech with label 0, each category 5388 with label 1 and 4903 for label 0. The dataset is preprocessed including case folding, data cleaning, lemmatization, word normalization, informal to formal words, tokenization. Then the confusion matrix calculation is used to test the model that has been built. There are two scenarios in this study, namely the effect of using the informal to formal word converter algorithm which shows value of 92.4% accuracy, 93.5% recall, 92.2% precision and 92.8% F1. In scenarios that have not used informal to formal word modifiers show a value of 90.7% accuracy, 89.9% recall, 92.3% precision and 91% F1. By using informal to formal word modifiers, it can improve the performance of the model that has been created.

REFERENCES

[1] J. S. Malik, H. Qiao, G. Pang, and A. van den Hengel, "Deep Learning for Hate Speech Detection: A Comparative Study," Feb. 2022, [Online]. Available: http://arxiv.org/abs/2202.09517

[2] B. Breve, L. Caruccio, S. Cirillo, V. Deufemia, and G. Polese, "Analyzing the worldwide perception of the Russia-Ukraine conflict through Twitter," *J Big Data*, vol. 11, no. 1, Dec. 2024, doi: 10.1186/s40537-024-00921-w.

[3] C. Liu, Y. Tian, Y. Shi, Z. Huang, and Y. Shao, "An analysis of public topics and sentiments based on social media during the COVID-19 Omicron Variant outbreak in Shanghai 2022," *Computational Urban Science*, vol. 4, no. 1, Dec. 2024, doi: 10.1007/s43762-024-00128-y.

[4] I. Oluwasegun Adeniyi, N. A. Sande, A. Akinkunmi Author, and I. Oluwasegun, "Social Media Sentiment Analysis: A Comprehensive Analysis", doi: 10.13140/RG.2.2.31094.37441.

[5] Nasrabadi N, Wicaksono H, and Valilai O, "Shopping marketplace analysis based on customer insights using social media analytics," *MethodsX*, vol. 9, Jan. 2022, doi: 10.1016/j.mex.2022.101932.

[6] Q. A. B. K. Zaman, W. N. S. B. W. Yusoff, and Q. B. B. A. Shah, "Sentiment Analysis on The Place of Interest in Malaysia," *Journal of Advanced Research in Applied Sciences and Engineering Technology*, vol. 43, no. 1, pp. 54–65, Jan. 2025, doi: 10.37934/araset.43.1.5465.

[7] A. Müller and M. Lopez-Sanchez, "Countering Negative Effects of Hate Speech in a Multi-Agent Society," in *Frontiers in Artificial Intelligence and Applications*, IOS Press BV, Oct. 2021, pp. 103–112. doi: 10.3233/FAIA210122.

[8] J. Forry Kusuma and A. Chowanda, "Indonesian Hate Speech Detection Using IndoBERTweet and BiLSTM on Twitter," 2020. [Online]. Available: www.joiv.org/index.php/joiv

[9] F. Nadia Puteri and Y. Sibaroni, "Hate Speech Detection in Indonesia Twitter Comments Using Convolutional Neural Network (CNN) and FastText Word Embedding," vol. 7, no. 3, pp. 1154–1161, 2023, doi: 10.30865/mib.v7i3.6401.

[10] M. Hayaty, A. D. Laksito, and S. Adi, "Hate speech detection on Indonesian text using word embedding method-global vector," *IAES International Journal of Artificial Intelligence*, vol. 12, no. 4, pp. 1928–1937, Dec. 2023, doi: 10.11591/ijai.v12.i4.pp1928-1937.

[11] I. Zulfikar, M. Nasrun, S. Si, and C. Setianingsih, "Deteksi Ujaran Kebencian Menggunakan Algoritma Glove Dan Deep Belief Network (Dbn)." Universitas Telkom, 2019.

[12] N. Badri, F. Kboubi, and A. H. Chaibi, "Combining FastText and Glove Word Embedding for Offensive and Hate speech Text Detection," in *Procedia Computer Science*, Elsevier B.V., 2022, pp. 769–778. doi: 10.1016/j.procs.2022.09.132.

[13] R. Rana, "Gated Recurrent Unit (GRU) for Emotion Classification from Noisy Speech," Dec. 2016, [Online]. Available: http://arxiv.org/abs/1612.07778

[14] M. Zulqarnain, R. Ghazali, Y. M. M. Hassim, and M. Rehan, "Text classification based on gated recurrent unit combines with support vector machine," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 4, pp. 3734–3742, 2020, doi: 10.11591/ijece.v10i4.pp3734-3742.

[15] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation."https://nlp.stanford.edu/projects/glove (accessed: Apr. 01, 2024).

[16] M. Devansh, H. YiDong, Alves de Oliveira, and Thiago Eustaquio, "A Curated Hate Speech Dataset," 2022.

[17] A. Toosi, "Twitter Sentiment Analysis." Accessed: Apr. 25, 2024. [Online]. Available: https://www.kaggle.com/datasets/arkhoshghalb/twitter-sentiment-analysis-hatred-speech/

[18] D. Putri *et al.*, "Hate Speech Detection on Twitter Approaching The Indonesian Election Using Machine Learning," Universitas Indonesia, 2018.

[19] J. Patihullah and E. Winarko, "Hate Speech Detection for Indonesia Tweets Using Word Embedding And Gated Recurrent Unit," *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, vol. 13, no. 1, p. 43, Jan. 2019, doi: 10.22146/ijccs.40125.

[20] H. Imaduddin, L. A. Kusumaningtias, and F. Y. A'la, "Application of LSTM and GloVe Word Embedding for Hate Speech Detection in Indonesian Twitter Data," *Ingénierie des systèmes d information*, vol. 28, no. 4, pp. 1107–1112, Aug. 2023, doi: 10.18280/isi.280430.

[21] A. Ahmad Aliero, B. Sulaimon Adebayo, H. Olanrewaju Aliyu, A. Gogo Tafida, B. Umar Kangiwa, and N. Muhammad Dankolo, "Systematic Review on Text Normalization Techniques and its Approach to Non-Standard Words," 2023.

[22] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation."https://nlp.stanford.edu/data/glove.6B.zip(accessed: Apr. 01, 2024)

[23] A. Rahmadeyan and Mustakim, "Long Short-Term Memory and Gated Recurrent Unit for Stock Price Prediction," in *Procedia Computer Science*, Elsevier B.V., 2024, pp. 204–212. doi: 10.1016/j.procs.2024.02.167.

[24] R. Achmad, Y. Tokoro, J. Haurissa, and A. Wijanarko, "Recurrent Neural Network-Gated Recurrent Unit for Indonesia-Sentani Papua Machine Translation," *Journal of Information Systems and Informatics*, vol. 5, no. 4, pp. 1449–1460, Dec. 2023, doi: 10.51519/journalisi.v5i4.597.

[25] S. Manna, "K-Fold Cross Validation for Deep Learning Models using Keras." Accessed: Jul. 11, 2024. [Online]. Available: https://medium.com/the-owl/k-fold-cross-validation-in-keras-3ec4a3a00538

[26] A. M. Peco Chacón, I. Segovia Ramírez, and F. P. García Márquez, "K-nearest neighbour and K-fold cross-validation used in wind turbines for false alarm detection," *Sustainable Futures*, vol. 6, Dec. 2023, doi: 10.1016/j.sftr.2023.100132.

[27] M. Hasnain, M. F. Pasha, I. Ghani, M. Imran, M. Y. Alzahrani, and R. Budiarto, "Evaluating Trust Prediction and Confusion Matrix Measures for Web Services Ranking," *IEEE Access*, vol. 8, pp. 90847–90861, 2020, doi: 10.1109/ACCESS.2020.2994222.