Jurnal Analisis Komputasi Digital

Vol.8 No.2, Februari 2024
ISSN: 24455131

PERFORMANCE COMPARISON OF LSTM, BILSTM AND CNN WITH MULTI-HEAD SELF-ATTENTION ON HATE SPEECH ANALYSIS

Irvan Yuniar M.[1], **Dwiza Riana**[2]
Universitas Nusa Mandiri
E-mail: 14210175@nusamandiri.co.id[1], dwiza@nusamandiri.co.id[2]

Abstrak

Media sosial seperti Twitter menjadi yang banyak digunakan orang untuk mengungkapkan perasaan dan pendapat mereka dengan bebas. Komunikasi memainkan peran penting dalam interaksi sosial. Pengumpulan data opini-opini pengguna tentang suatu masalah dapat dilakukan untuk proses analisis sentimen dengan memasukan dalam kategori positif, negatif, atau netral. Penelitian ini bertujuan untuk mencari performa dari masing-masing model yang diujicoba dalam mencapai tujuan mencari accuracy untuk teks klasifikasi. Pendekatan linguistik diuji menggunakan berbagai jenis fitur dan model untuk analisis sentimen yang akurat, pengujian model LSTM, BiLSTM, dan CNN+MHSA dengan word embedding GloVe untuk mengetahui analisis sentimen ujaran kebencian. Penelitian dilakukan dengan melakukan pengumpulan dataset dari Twitter, labelling data, balance data, text processing, word embedding, modelling dan hasil pengujian. Pengujian model menggunakan model LSTM, BiLSTM dan CNN dengan Multi-Head Self Attention serta word embedding GloVe dengan jumlah dataset sebanyak 6923 data yang terdiri dari 6465 data sentimen bukan ujaran kebencian dan 467 data sentimen ujaran kebencian. Hasil pengujian menunjukkan bahwa ketiga metode yang dilakukan sama baiknya dalam analisis sentimen ujaran kebencian menggunakan dataset yang dikumpulkan sendiri, dengan hasil akurasi yang cukup tinggi yaitu 93.32%.
Kata Kunci *— Multi-Head Self Attention, LSTM, BiLSTM, CNN, analisis sentimen*

Abstract

Social media, such as Twitter, is widely used by people to express their feelings and opinions freely. Communication plays an important role in social interactions. Data collection on user opinions about an issue can be carried out for the sentiment analysis process by entering it into positive, negative, or neutral categories. This research aims to find out the performance of each model tested in achieving the goal of finding accuracy for text classification. The linguistic approach was tested using various types of features and models for accurate sentiment analysis, testing the LSTM, BiLSTM, and CNN+MHSA models with GloVe word embedding to determine the sentiment analysis of hate speech. The research was carried out by collecting datasets from Twitter, labelling data, balancing data, text processing, word embedding, modelling, and testing results. Model testing uses LSTM, BiLSTM, and CNN models with multi-head self-attention and GloVe word embedding, with a total dataset of 6923 data points consisting of 6465 non-hate speech sentiment data and 467 hate speech sentiment data. The test results show that the

three methods used are equally good at analysing the sentiment of hate speech using a dataset collected by ourselves, with quite high accuracy results, namely 93.32%.
Keywords*— Multi-Head Self Attention, LSTM, BiLSTM, CNN, sentiment analysis*

1. INTRODUCTION

Communication has undergone major changes from the past due to the rapid development of technology today. Communication plays an important role in social interaction; this is evidenced by the increasing use of social media in recent years and has entered into people's daily lives [1]. Social media such as Twitter, Meta (formerly Facebook), Instagram or other open forums are what many people use to express their feelings and opinions freely [2]. Collecting data on user opinions on an issue can be done for sentiment analysis by categorizing them as positive, negative, or neutral. Analyzing opinions on social networks can help understand public opinion. For example, entrepreneurs know what they can improve about their products by analyzing customer opinions. Also, political parties can use opinion analysis to create action plans.

Due to its vast potential, sentiment analysis has become one of the most widely used techniques in recent years, different interests and motivations with users all over the world having the opportunity to freely express their opinions on various matters [3], [4]. Sentiment analysis, also known as opinion research, involves Natural Language Processing (NLP) and Deep Learning approaches to systematically extract and identify subjective information using machine learning techniques [4]. Sentiment analysis is an activity that is able to process information submitted on social media to understand the behavior and attitudes of individuals towards certain issues using machine learning techniques using computer algorithms that automatically improve as experience is stored [5], simply put, sentiment analysis analyzes the polarity and sentiment of written text to determine whether it is positive, negative, or neutral. Sentiment analysis of text data seems like an easy task for humans, but when it comes to analyzing thousands of text data simultaneously, sentiment analysis becomes a difficult and time-consuming task [6].

This research uses a private dataset collected from Twitter. The raw tweets collected are 7220 tweets which will be processed to determine sentiment analysis regarding hate speech. Previous research that discusses sentiment analysis is quite a lot, including conducted by Jitendra Soni and friends [7] using the LSTM model, which is a network architecture designed to "remember" previously read values over a period of time, BiLSTM Long Short-Term Memory (LSTM) neural network formed by two layers of LSTM neural networks, namely the advanced LSTM layer to model the previous context and the backward LSTM layer to model each subsequent context [8], BERT and LSTM Encoder with Self-Attention. The results showed that the LSTM encoder model with Self-Attention outperformed the three other methods tested and could be used as an effective technique for selecting features that have a large impact on the accuracy obtained by 86%. Further research conducted by Chih-Hsueh Lin and friends [9] conducted research on text representation using Twitter datasets with the proposed method using the BERT method and distilled version of BERT (DistilBERT). Researchers extract features selected from representative vectors of input sentences, while the combination of LSTM and CNN is carried out to increase the accuracy of text representation.

Based on the research conducted previously, it can be seen that Self Attention is used to detect sentiment analysis with fairly good accuracy. The success of Self Attention depends on the selection of the right features. In other words, it can be said that the selection of parameters and features greatly affects classification accuracy [10]. Feature selection is the stage of selecting and extracting important and valuable data information.

The attributes and information included in the right sentiment analysis model can reduce time and cost, and even increase accuracy. It is necessary to select attributes on the right dataset to improve accuracy results [10].

By considering the aspects and context hidden behind the text, researchers will test three models to determine the performance of the three models in analyzing the sentiment of hate speech. Three models will be tested, namely LTSM, BiLSTM, and CNN with Multi-Head Self Attention (CNN+MHSA) [11], [12], [13] Attention mechanisms can help improve the performance of Deep Learning models such as Machine Translation [3], Text Summarization [14] In this research, various factors and different contexts and aspects have been considered while analyzing Tweet opinions. Augmentation techniques are performed by embedding words in the data using GloVe [8], [15], this is done in order to generate more training examples for model learning and better generalization capabilities so that an increase in accuracy will be obtained.

Based on the above background, it can be seen that LSTM, BiLSTM and CNN have excellent classification capabilities for sentiment analysis, but the large number of attributes in a data can reduce the level of accuracy and increase the complexity of the algorithm, to overcome this it is necessary to select which attributes will be removed and which attributes will be used. The purpose of this research is to find the performance of each model tested in achieving the goal of finding accuracy for text classification, namely by testing the LSTM, BiLSTM, and CNN + MHSA models with GloVe word embedding to determine the sentiment analysis of hate speech datasets.

2. METHODS

This research will use a private dataset collected from Twitter from March 1, 2023 to May 31, 2023 with the amount of raw data collected as many as 7220 tweets with 8 features (username, full_text, quote_count, reply_count, retweet_count, favorite_count, lang, user_id_str). This research method steps can be seen in Figure 1. below:

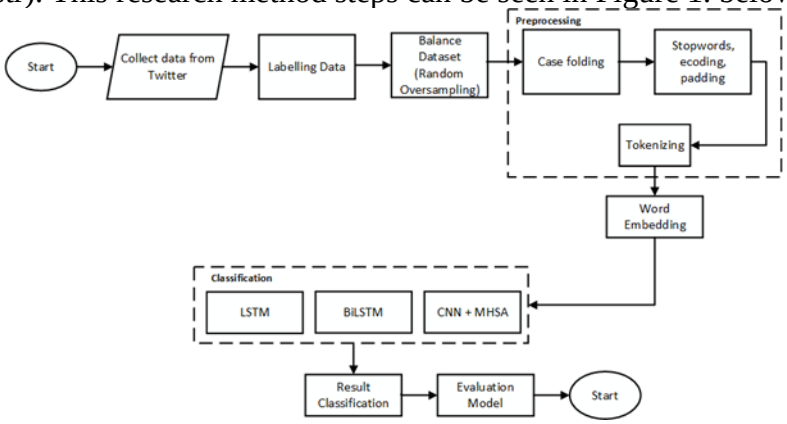

Figure 1 Research Method Steps

Dataset Data

Where and how the data came from is explained in this section. The data collected for this study defines what happened in the study by finding reference data from previous studies. Search for available data, obtain any additional data required, and integrate all data into the dataset, including the variables required for the process. This phase is the initial phase of the research to understand the scope of the problem and determine the objectives of the research.

Labelling Data

At this stage it is explained about data labeling, where the data that has been collected will be divided into 2 classes, namely the class of non-hate speech or negative

sentiment and the class of hate speech or positive sentiment, by labeling each data that has been collected. Data labels are needed so that there are parameters that can be used to measure the sentiment analysis of the data to be processed.

Balance Data

At this stage, the data used is unbalanced, so the balancing process is needed so that the processed data can produce high accuracy. In the previous stage, it is known that the data to be processed is imbalanced data, this unbalanced data cannot be directly tested to find accuracy because the resulting results are not good. To overcome imbalance data, the resampling method is carried out using Oversampling to balance negative and positive data. This oversampling process changes positive data to the same amount as negative data so that the data becomes balanced. The results of data balancing using oversampling can be seen in Figure 2. below.

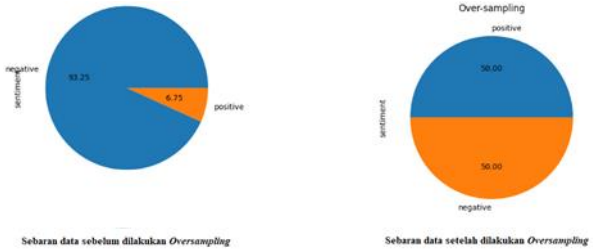

Figure 2 Dataset distribution

Text Processing

In this section, text processing is carried out to clean the data from characters or words that are not needed in model testing. At this stage, data is separated into training and testing data needed for modeling.

Table 1 Data Processing Results

Before data processing	After data processing	Sentiment
Hidup china komunis! https:\/\/t.co\/b1RheVKrXl	hidup china komunis https co rhevkrxl	1
ketutunan cina dari luar kota dan pulau mulai kejakarta mau coblos ahok dengan KTP aspal dari kamboja	ketutunan cina dari luar kota dan pulau mulai kejakarta mau coblos ahok dengan ktp aspal dari kamboja	1

Word Embedding

Word embedding is done to map each word in the document into a vecto. This word embedding allows the semantic and syntactic meaning of the word to be captured. Word embedding by inserting words into the document forms vectors, where each vector represents the projection of the word in the vector space. In this study we decided to use the pre-trained GloVe Twitter model, the embedding size is limited to 100 words.

Modelling

This section describes the steps for testing the model including selecting the appropriate model or method proposed to produce results that match the method used, grouping related variables together. After the data is analyzed, the model is applied based on the data type. This experiment is conducted using a private dataset taken from Twitter, the dataset to be processed is imbalance data, where the number of negative and positive sentiments is not balanced. The dataset that has been done the stages of labeling, data processing, word embedding produces data as much as 6932 Tweets consisting of 2 classes of non-hate speech (0) and hate speech class (1). In this study, three proposed models will be tested, namely: LSTM, BiLSTM, and CNN with Multi-Head Self Attention (CNN+MHSA). This method testing is done to find accuracy, precision, and recall. Testing each method uses the same dataset, batch size and epoch.

Test Result

In this section, the results of model testing are evaluated and validated to determine the accuracy of the model. The model that has been formed in the next stage is carried out the testing and validation process to determine the accuracy, precision, and recall values.

3. RESULT AND DISCUSSION

Dataset Collection Results

This research uses a dataset that is self-collected from Twitter social media using python. To facilitate data collection, during the search, the keywords "china", "bastard", "lgbt", "ahok", "anies ganjar", "incompetent candidate", "jahaman", "arrogant, "rocky gerung", "do not know yourself" were used. The keywords used are not all directly related to words that have the meaning of hate speech, this is done rather than most of the data searched is expected to be only negative tweets. The data search was limited to a maximum of 100 tweets per keyword, the data collected from 10 keywords varied in results. Data searches began with the time frame of March 1, 2023, to May 31, 2023, this time frame was chosen due to limitations from Twitter which began to limit data searches from Twitter for non-paid users, to be able to collect a lot of data you must have a paid Twitter API account.

The results of data collection before labeling and data preprocessing are checked to remove content that is not related to the desired dataset, for example the contents of tweets about advertisements submitted by Twitter users, news headlines from social media news pages such as detik.com, kompas.com. The data that has been removed from unnecessary contents is 7220 data with 8 features (username, full_text, quote_count, reply_count, retweet_count, favorite_count, lang, user_id_str) with data content that is still not clean, where each data still contains punctuation that must be cleaned and there is no data labeling whether the existing tweets are negative (0) or positive (1). The 8 features in the raw dataset will be selected suitable features for data testing using a predetermined model for hate speech sentiment analysis.

Labelling Data

The raw dataset that has been collected with a total of 7220 data and still has not been processed will be labeled whether the tweets are not hate speech or hate speech, this is done to facilitate data recognition and testing to find out the characteristics of the data to be tested. Data labeling is done manually where each data is checked whether the word is hate speech or not. Before the data labeling process, the 7220 existing data is checked for data duplication, if there is then the duplicated data will be removed from the dataset. The process of checking and removing duplicate data is done using python, the amount of data after removing duplicate data becomes 6932 data and labeled with the results of 6456 data with labels not hate speech (93.25%), and 467 data with hate speech labels (6.75%).

Balance Data

Before balancing the data, the dataset that has been displayed is removed from the columns that are not needed because these columns do not contain significant data to be processed. Of the 8 existing columns, only 2 columns are used, namely the sentiment column which contains sentiment labels and text which contains sentences / tweets.

Data balancing is done using the Oversampling technique, where data labeled sentiment 1 will be equalized with data labeled sentiment 0 by randomly duplicating data from positive sentiment, this is done so that the amount of positive sentiment data is balanced with negative sentiment data, as in Figure 3. below.

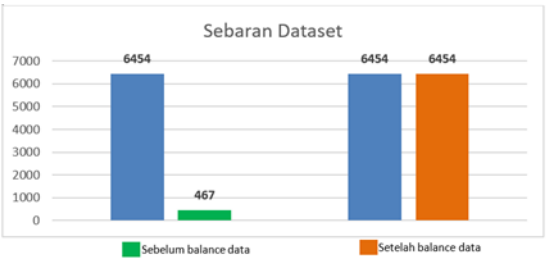

Figure 3 Comparison Dataset before and after data balance

Text Processing

At this stage the data that has been balanced will be text processed, starting from changing all words to lowercase letters (case folding), removing punctuation, stopwords, split dataset, tokenization, label encoder, and padding, all of these processes are carried out to make a clean dataset.

The first stage of text processing is to perform case folding (lowercase) and remove punctuation in each sentence. The process of removing punctuation is done using the python module, regex (regular expression), this module can function to return a string where all matching sentences of the specified pattern are replaced with a replacement string. Each sentence is checked to convert it to lowercase, clean up the punctuation, so that the resulting words match the specified pattern. Case folding and removing punctuation sentences are done in the text feature, sentences that have gone through the process are entered into a new feature called clean text to store sentences that have been changed to lowercase and no more punctuation.

The following is the result of the case folding process and removing punctuation which can be seen in Figure 4 below.

	text	clean text
0	Ini pasti salah Jokowi, Ahok dan kafir-kafir a...	ini pasti salah jokowi ahok dan kafir kafir at...
1	@fahmiabuazzam1 biar aja biar tai nya kena bap...	fahmiabuazzam biar aja biar tai nya kena bapak...
2	RT @BungWinar: Angin memporakporandakan basis ...	rt bungwinar angin memporakporandakan basis pe...
3	RT @Lupuz0503: Prestasi Ahok, \nSelain mjadi m...	rt lupuz prestasi ahok nselain mjadi mafia kor...
4	Ulama Kompak Nyatakan #HaramPemimpinKafir Pili...	ulama kompak nyatakan harampemimpinkafir pilih...
...	...	...
6918	USER Pak Recep.....anda salah, itu gu...	user pak recep anda salah itu gubernur pakkkk ...
6919	brengsek itu orang terbuat dr apa bikin gue be...	brengsek itu orang terbuat dr apa bikin gue be...
6920	Kapolda Babi! Biadap dan Bodoh! Gak punya otak...	kapolda babi biadap dan bodoh gak punya otak kali
6921	USER jangan asal ngomong ndasmu. congor lu yg ...	user jangan asal ngomong ndasmu congor lu yg s...
6922	USER Mana situ ngasih(": itu cuma foto ya kuti...	user mana situ ngasih itu cuma foto ya kutil onta

Figure 4 Result of case folding and punctuation removal

Sentences that have been cleared of punctuation are then subjected to a stopwords process, this is done to reduce the number of words in the sentence so that the testing process is faster. The longer the sentence will affect the speed and performance of the NLP process. This process is done using the stopwords library for Indonesian from NLTK. There are 758 Indonesian stopwords in the NLTK library. To add stopwords that are not in the library, 60 stopwords are added in the sentence in the form of abbreviations and slang that will be used to eliminate these words when the stopwords process is run.

The next text processing process is stemming and tokenizer. The stemming process is done to change the words in the sentence into basic words, while the tokenizer is done to divide the sentence into words. Stemming is done using the Sastrawi library developed from PHP Sastrawi, with the help of the Sastrawi library it can change the affixed words in Indonesian to their original form (basic words), while the tokenizer uses a library from NLTK, namely word_tokenize to convert sentences into words per word.

Next, the dataset is divided into training data and test data with a division of 80%

training data and 20% test data using the train_test_split library from sklearn. The dataset that has been divided is carried out a padding process so that the existing data has the same length, for data that is less, the number 0 will be added so that the length is the same.

Word Embedding (GloVe)

The word embedding method is used to test this sentiment analysis, word embedding is a technique used to represent words into vectors. This method is widely used in research that discusses sentiment analysis. In this research, the GloVe (Global Vector) word embedding model is used because of its better results compared to other word embedding such as Word2vec (CBOW & Skip-gram).

Furthermore, the dataset and word embedding are used for analysis for sentiment classification. Word embedding in this research will be used using Deep Learning techniques with LSTM, BiLSTM, and CNN+MHSA. The GloVe used in this word embedding process uses the file glove.twitter.27B.100d.txt with a word embedding dimension of 100.

Testing Methods

The next stage in this section is testing the dataset using the three proposed methods. The dataset that has been done data balancing and text processing is tested using google colab and python 3.9. The results of this research are available in the form of a calculation process based on the LSTM, BiLSTM and CNN+MHSA models.

The LSTM, BiLSTM and CNN with Multi-Head Self Attention methods were tested using the same hyperparameters for batch size of 64 and epoch 10 and learning rates of 0.001 and 0.005. The performance of the model is highly dependent on the hyperparameters used. The hyperparamater used is the hyperparameter that was tested and produced the best results.

In this test, the dataset is divided into training dataset with a total of 5538 data, and testing dataset with a total of 1385 data.

1. Long Short-Term Memory + GloVe Method (LSTM + GloVe)

The first test with the LSTM method using GloVe word embedding by using hyperparameters to improve model performance can be seen in Table 3 below:

Table 3 LSTM Hyperparameter Table

Batch size	64
Epoch	10
Learning rate	0.001, 0.005
Loss	*binary crossentropy*
Optimizer	*adam*
Activation	*sigmoid*
Dropout	0.8
Dense	1

```
model = Sequential()
model.add(embedding_layer)
model.add(Dropout(0.8))
model.add(LSTM(64, dropout=0.8, recurrent_dropout=0.8))
model.add(Dense(1, activation='sigmoid'))

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate=0.001,
    decay_steps=1e-6,
    decay_rate=0.9)
optimizer = tf.keras.optimizers.Adam(learning_rate=lr_schedule)

sgd = optimizers.SGD(learning_rate=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.summary()
```

```
model.compile(loss='binary_crossentropy',
              optimizer="adam",
              metrics=['accuracy'])
```

```
model = Sequential()
model.add(embedding_layer)
model.add(Dropout(0.8))
model.add(LSTM(64, dropout=0.8, recurrent_dropout=0.8))
model.add(Dense(1, activation='sigmoid'))

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate=0.005,
    decay_steps=1e-6,
    decay_rate=0.9)
optimizer = tf.keras.optimizers.Adam(learning_rate=lr_schedule)

sgd = optimizers.SGD(learning_rate=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.summary()
```

```
model.compile(loss='binary_crossentropy',
              optimizer="adam",
              metrics=['accuracy'])
```

Figure 5 LSTM modeling pyhton code learning rate 0.001 & 0.005

Table 4 Test results of LSTM + GloVe method

Model	Batch Size	l-rate	Accuracy	Precison	F1-score
LSTM+GloVe	64	0.001	93.07%	98.0%	96.0%
LSTM+GloVe	64	0.005	93.07%	98.0%	96.0%

The results of testing the LTSM + GloVe method with a learning rate of 0.001 and 0.005 show the same accuracy, precision, and f1-score results, namely 93.07% accuracy, 98.0% precision, and 96.0% for f1-score.

Figures 6 below are graphs of the test results of training data and testing data when testing with a learning rate of 0.001 and 0.005. The graphs of training and validation loss and training and validation accurac show good results, where the training data does not show overfitting, which is an ideal model.

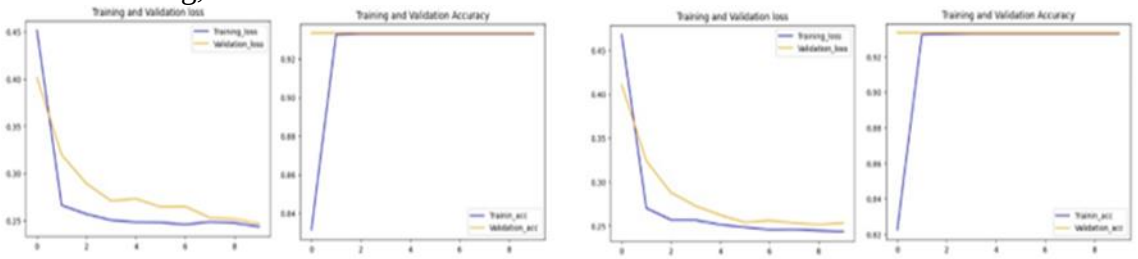

Figure 6 LSTM + GloVe test results with LR 0.001 & 0.005

2. Bidirectional Long Short-Term Memory + GloVE (BiLSTM-GloVe) Method

Testing the second method is the same as in the first test using LSTM, in this second test the BiLSTM model will be used, with word embedding using GloVe with testing done using epoch 10, batch size 64, learing rate 0.001 and 0.005 just like the LSTM method. This is done to determine the resulting accuracy between LSTM and BiLTSM.

Table 5 BiLSTM Hyperparameter Table

Batch size	64
Epoch	10
Learning rate	0.001, 0.005
Loss	*binary crossentropy*
Optimizer	*adam*
Activation	*sigmoid*
Dropout	0.8
Dense	1

```
model = Sequential()
model.add(embedding_layer)
model.add(Dropout(0.8))
model.add(Bidirectional(LSTM(64, dropout=0.8, recurrent_dropout=0.8)))
model.add(Dense(1, activation='sigmoid'))

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate=0.001,
    decay_steps=1e-6,
    decay_rate=0.9)
optimizer = tf.keras.optimizers.Adam(learning_rate=lr_schedule)

model.summary()
```

```
model.compile(loss='binary_crossentropy',
              optimizer="adam",
              metrics=['accuracy'])
```

```
model = Sequential()
model.add(embedding_layer)
model.add(Dropout(0.8))
model.add(Bidirectional(LSTM(64, dropout=0.8, recurrent_dropout=0.8)))
model.add(Dense(1, activation='sigmoid'))

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate=0.005,
    decay_steps=1e-6,
    decay_rate=0.9)
optimizer = tf.keras.optimizers.Adam(learning_rate=lr_schedule)

model.summary()
```

```
model.compile(loss='binary_crossentropy',
              optimizer="adam",
              metrics=['accuracy'])
```

Figure 7 Python code modeling BiLSTM learning rate 0.001 & 0.005

The following are the results of testing the BiLSTM + Glove model with a batch size of 64 and learning rates of 0.001 and 0.005.

Table 6 Test results of the BiLSTM + GloVe method

Model	Batch Size	l-rate	Accuracy	Precison	F1-score
BiLSTM+GloVe	64	0.001	93.07%	88.0%	94.0%
BiLSTM+GloVe	64	0.005	93.07%	88.0%	94.0%

The results of testing the BiLTSM + GloVe method with a learning rate of 0.001 and 0.005 show the same accuracy, precision, and f1-score results as the LSTM method, namely 93.07% accuracy, 88.0% precision, and 94.0% for f1-score.

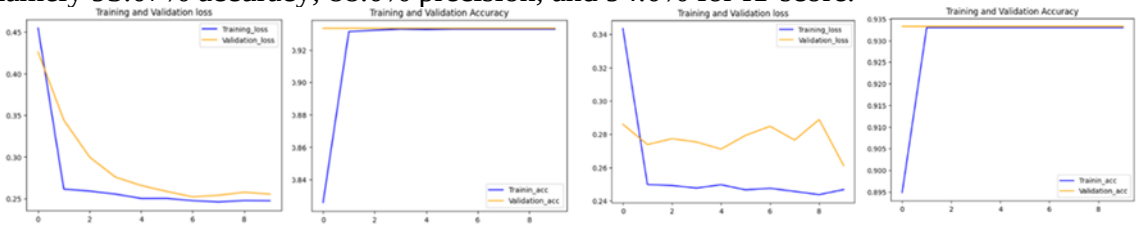

Figure 8 BiLSTM + GloVe test results with LR 0.001 & 0.005

Figure 8 shows the graph of the test results of training data and testing data when testing with a learning rate of 0.001 produces an optimal model where there is no overfit data and for used of learning rate 0.005 show the graph generated for validation data shows overfitting even though the training and validation accuracy produce the same results as testing using a learning rate of 0.001 does not show overfitting.

From the 2 images of the BiLSTM testing graph, the test results with a learning rate of 0.001 get a more optimal test model compared to testing using a learning rate of 0.005.

3. Convolutional Neural Network + Multi-Head Self Attention (CNN+MHSA+GloVe)

The third method tested was CNN+MHSA with fixed word embedding using GloVe and tested using epoch 10, batch size 64, activation sigmoid with adam optimizer and learing rate 0.001 and 0.005. This hyperparamer is the same as the LSTM and BILSTM methods, for this test the dropout used is 0.2, when using a dropout of 0.8 the test results show overfitting data.

The following hyperparameter table is the result of testing the CNN + MHSA + Glove model with a batch size of 64, and learning rates of 0.001 and 0.005.

Table 7 CNN Hyperparameter Table

Batch size	64
Epoch	10
Learning rate	0.001, 0.005
Loss	*binary crossentropy*
Optimizer	*adam*
Activation	*sigmoid*
Dropout	0.2
Dense	5
Conv1D	128

Figure 9 Code pyhton modeling CNN+MHSA learning rate 0.001 & 0.005

Table 8 Test results of the BiLSTM + GloVe method

Model	Batch Size	l-rate	Accuracy	Precison	F1-score
CNN+GloVe +MHSA	64	0.001	93.32%	98.0%	96.0%
CNN+GloVe +MHSA	64	0.005	93. 32%	98.0%	96.0%

The test results of the CNN+MHSA+GloVe method with the same test as LSTM and BiLSTM show slightly better accuracy than LSTM and BiLSTM, both for tests with a learning rate of 0.001 and a learning rate of 0.005. The test results obtained 93.32% accuracy, 98.0% precision, and 96.0% for f1-score.

Figure 11 below is a graph of the test results of training data and testing data when testing with a learning rate of 0.001. Test results using learning rate 0.001 and 0.005 for training and validation testing in Figure 11 shows a fairly optimal model during the training and validation data process, where there is no overfit data, the resulting curve shows the training and testing data graphs remain the same, there is no pattern that shows underfit or overfit.

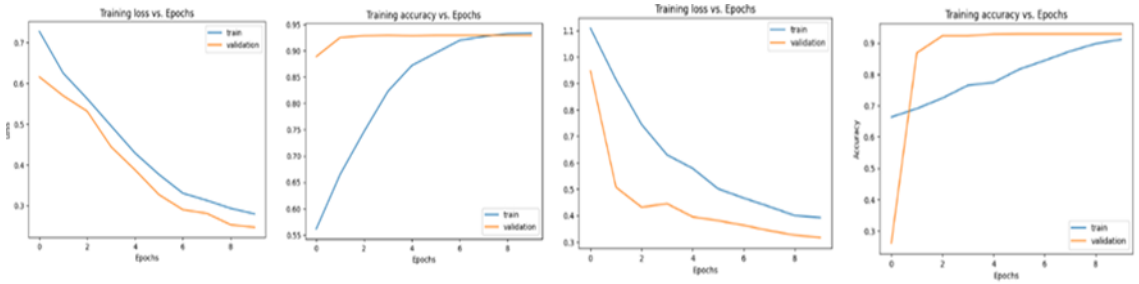

Figure 10 CNN + MHSA + Glove test result with LR 0.001 & 0.005

When looking at the graph of the dataset test results using a learning rate of 0.001 and 0.005, testing with a learning rate of 0.001 gets the optimal model with data that is not overfit.

Confusion matrix testing of these three models to see prediction and actual testing shows an unbalanced combination composition, where in this confusion matrix the data is spread only in the True Positive (TF) and False Negative (NF) columns, while the False Positive (TP) and True Negative (TN) data are empty. With this confusion matrix, it can be concluded that the data used to predict hate speech sentiment analysis can be predicted correctly.

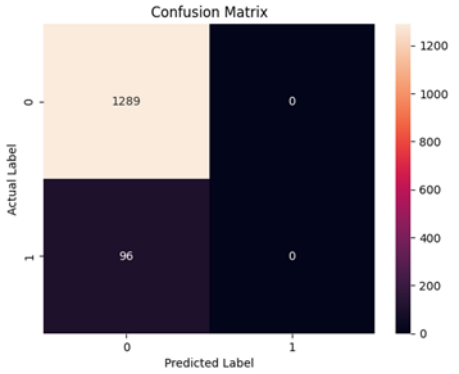

Figure 11 Confusion Matrix

Test Result

From testing the above methods to predict the data used whether it is hated speech or not shows that the three models can be used to perform sentiment analysis quite well. By adding GloVe as word embedding, the tested data produces stable accuracy and performance.

Of the three models tested, the CNN+MHSA+GloVe model shows slightly better accuracy results than LSTM and BiLSTM, with an accuracy value of 93.32%.

Table 9 Sentiment analysis prediction results

sentiment	clean text	sentiment_pred
0	[partai, kunyuk]	not hate speech
0	[cie, butuh, dunia, internasional, xef, xbc, f...	not hate speech
1	siang, malam, maki, ulama, nongkrongi, medsos, ulama, nbela, nista, agama, sombong, ampun	not hate speech
1	hahaha, pilkada, dasar, polisi, made, in, cina, lindung, nista, agama	hate speech

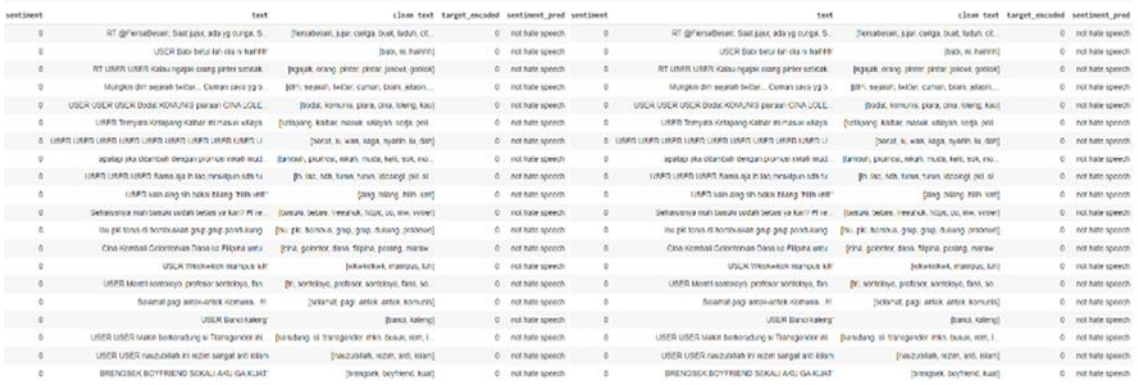

Figure 12 Prediction results of hate speech sentiment analysis

4. CONCLUSIONS

The conclusion that can be drawn from this research is that the research method used in this study is to analyze the sentiment of hate speech using three Deep Learning methods namely LSTM, BiLSTM, and CNN using GloVe and Multi-Head Self Attention. The dataset used is a private Twitter tweet data that is not balanced (imbalance data) the amount of data between non-hate speech data and hate speech data. Test results that show good performance are obtained from the CNN + MHSA + GloVe model with an accuracy rate of 93.32%. This shows that sentiment analysis with CNN classification is very good for sentiment analysis classification.

This research was conducted with several limitations, among others, the dataset used is very small, the methods tested are only 2 methods and these are commonly used methods for classification. The tests carried out do not use many layers so that the data testing process is more accurate. Therefore, there are limitations to this research, it is recommended that in future research, in addition to using the LSTM and BiLSTM methods, this research can use other models such as BERT with BERT Embedding using word embedding such as Word2vec or FastText.

REFERENCES

[1] A. Karami, M. Lundy, F. Webb, and Y. K. Dwivedi, “Twitter and Research: A Systematic Literature Review through Text Mining,” IEEE Access, vol. 8, pp. 67698–67717, 2020, doi: 10.1109/ACCESS.2020.2983656.

[2] M. U. Salur and I. Aydin, “A Novel Hybrid Deep Learning Model for Sentiment Classification,” IEEE Access, vol. 8, pp. 58080–58093, 2020, doi: 10.1109/ACCESS.2020.2982538.

[3] S. Kamiş and D. Goularas, “Evaluation of Deep Learning Techniques in Sentiment Analysis from Twitter Data,” in Proceedings - 2019 International Conference on Deep Learning and Machine Learning in Emerging Applications, Deep-ML 2019, Institute of Electrical and

Electronics Engineers Inc., Aug. 2019, pp. 12–17. doi: 10.1109/Deep-ML.2019.00011.
[4] K. L. Tan, C. P. Lee, K. M. Lim, and K. S. M. Anbananthen, “Sentiment Analysis With Ensemble Hybrid Deep Learning Model,” IEEE Access, vol. 10, pp. 103694–103704, 2022, doi: 10.1109/ACCESS.2022.3210182.
[5] A. A. Nagra, K. Alissa, T. M. Ghazal, S. Saigeeta, M. M. Asif, and M. Fawad, “Deep Sentiments Analysis for Roman Urdu Dataset Using Faster Recurrent Convolutional Neural Network Model,” Applied Artificial Intelligence, vol. 36, no. 1, 2022, doi: 10.1080/08839514.2022.2123094.
[6] A. Mohammadi and A. Shaverizade, “Ensemble deep learning for aspect-based sentiment analysis,” International Journal of Nonlinear Analysis and Applications, vol. 12, no. Special Issue, pp. 29–38, 2021, doi: 10.22075/IJNAA.2021.4769.
[7] J. Soni and K. Mathur, “Sentiment analysis based on aspect and context fusion using attention encoder with LSTM,” International Journal of Information Technology (Singapore), vol. 14, no. 7, pp. 3611–3618, Dec. 2022, doi: 10.1007/s41870-022-00966-1.
[8] A. R. Isnain, A. Sihabuddin, and Y. Suyanto, “Bidirectional Long Short Term Memory Method and Word2vec Extraction Approach for Hate Speech Detection,” IJCCS (Indonesian Journal of Computing and Cybernetics Systems), vol. 14, no. 2, p. 169, Apr. 2020, doi: 10.22146/ijccs.51743.
[9] C. H. Lin and U. Nuha, “Sentiment analysis of Indonesian datasets based on a hybrid deep-learning strategy,” J Big Data, vol. 10, no. 1, Dec. 2023, doi: 10.1186/s40537-023-00782-9.
[10] A. Sriram, D. Sinha, and S. Gupta, “Performance Comparison of Deep Learning Algorithms for Sentiment Analysis,” International Conference on IoT based Control Networks and Intelligent Systems (ICICNIS 2020) , 2020, [Online]. Available: https://ssrn.com/abstract=3896958
[11] H. A. Shehu et al., “Deep Sentiment Analysis: A Case Study on Stemmed Turkish Twitter Data,” IEEE Access, vol. 9, pp. 56836–56854, 2021, doi: 10.1109/ACCESS.2021.3071393.
[12] L. Khan, A. Amjad, K. M. Afaq, and H. T. Chang, “Deep Sentiment Analysis Using CNN-LSTM Architecture of English and Roman Urdu Text Shared in Social Media,” Applied Sciences (Switzerland), vol. 12, no. 5, Mar. 2022, doi: 10.3390/app12052694.
[13] A. K. Sharma, S. Chaurasia, and D. K. Srivastava, “Sentimental Short Sentences Classification by Using CNN Deep Learning Model with Fine Tuned Word2Vec,” in Procedia Computer Science, Elsevier B.V., 2020, pp. 1139–1147. doi: 10.1016/j.procs.2020.03.416.
[14] B. Alkouz, Z. Al Aghbari, and J. H. Abawajy, “Tweetluenza: Predicting flu trends from twitter data,” Big Data Mining and Analytics, vol. 2, no. 4, pp. 273–287, Dec. 2019, doi: 10.26599/BDMA.2019.9020012.
[15] V. Tyagi, A. Kumar, and S. Das, “Sentiment Analysis on Twitter Data Using Deep Learning approach,” in Proceedings - IEEE 2020 2nd International Conference on Advances in Computing, Communication Control and Networking, ICACCCN 2020, Institute of Electrical and Electronics Engineers Inc., Dec. 2020, pp. 187–190. doi: 10.1109/ICACCCN51052.2020.9362853.