The Indonesian Journal of Computer Science
www.ijcs.net
Volume 14, Issue 2, April 2025
https://doi.org/10.33022/ijcs.v14i2.4809

Optimization of Deep Learning with FastText for Sentiment Analysis of the SIREKAP 2024 Application

Handoko[1], Junadhi[2], Triyani Arita Fitri[3], Agustin[4]
handoko0193@gmail.com[1], junadhi@usti.ac.id[2], triyani@usti.ac.id[3], agustin@usti.ac.id[4]
[1,2,3,4]Universitas Sains & Teknologi Indonesia

Article Information

Received : 22 Mar 2025
Revised : 30 Mar 2025
Accepted : 15 Apr 2025

Keywords

Sentiment Analysis, SIREKAP 2024, Deep Learning, Word Embedding, FastText.

Abstract

This study analyzes public sentiment towards the SIREKAP 2024 application using deep learning. Data was collected from Google Playstore reviews and processed through cleaning, tokenization, and stemming. Word embedding was performed using FastText to capture more accurate word representations, including OOV words. The deep learning models compared were CNN, BiLSTM, and BiGRU. Performance evaluation used accuracy, precision, recall, and F1-score metrics. The results showed that the CNN model with FastText Gensim embedding achieved the highest accuracy of 95.98%, outperforming BiLSTM and BiGRU. This model was more effective in classifying positive and negative sentiments. This study provides insights for developers to improve the performance and public trust in SIREKAP 2024 and opens opportunities for further research with more complex embedding approaches and deep learning models.

A. Introduction

Various elections have been held in Indonesia since gaining independence in 1945 [1]. The organisation of elections in Indonesia is the responsibility of the General Elections Commission (KPU). The KPU is tasked with organising elections to elect the president and vice president, as well as members of the DPR, DPD, and DPRD, in accordance with Law Number 7 Year 2017. The KPU determines the number of seats for political parties participating in the election, the permanent voters list, campaign deadlines, election regulations, vote counting protocols, and polling station locations. In addition to carrying out other duties and other legal powers, the KPU also collects and determines election results. To ensure free and fair elections, the KPU must carry out its duties and authorities competently, openly, and accountably [2].

Sirekap is the name of the Recapitulation Information System. This information system is used to assist the Indonesian General Election Commission (KPU) in calculating and summarising election results, both general and regional elections. The system enables a faster, more efficient and transparent vote counting process by collating vote data at all levels, from local to national. Sirekap reduces the KPU's reliance on manual processes that have the potential for error and manipulation by assisting them in conducting online recapitulation. Therefore, Sirekap is expected to increase public confidence in the integrity of the electoral process and improve the accuracy and reliability of election results [3].

The process of identifying and classifying the various feelings or opinions that people have towards a service, product, event, or feature is called sentiment analysis, also known as opinion mining. The main goal of sentiment analysis is to determine whether the opinions expressed are neutral, negative or positive. In this context, the feelings or opinions can cover a wide range of topics, such as policy, product quality, consumer satisfaction, and so on [4].

Sentiment analysis can be done with deep learning. One popular method is to analyse text and extract relevant information to predict sentiment using neural network architectures such as Convolutional Neural Networks (CNN) or Recurrent Neural Networks (RNN), specifically Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) [5].

Due to its strong capability in feature learning and feature representation, Convolutional Neural Networks (CNN), which are widely used in natural language processing applications such as text classification and sentiment classification, are used in this study [6].

The purpose of this study is to understand how sentiment analysis is applied to public opinion related to elections in Indonesia, specifically related to the use of Sirekap by the General Election Commission (KPU). The main objectives of this research are to analyse public opinion [7], compare CNN, Bi-LSTM, and Bi-GRU models with conventional techniques, and provide recommendations to increase public trust in the political process. The approach used includes analysis of SIREKAP comments from Google Playstore, implementation of CNN, Bi-LSTM, and Bi-GRU models, and evaluation of model performance.

To obtain accurate features in sentiment analysis, word representation or word embedding is essential. FastText, an algorithm developed by Facebook AI Research [8], is one of the popular word representation techniques. FastText uses a unique approach by considering each word as a collection of n-gram characters. In contrast to conventional word representation methods, FastText allows the model to recognize word patterns, resulting in a more reliable representation even for words that are not in the vocabulary [9]. FastText is particularly suitable for languages with complex morphology such as Indonesian, which has a variety of word forms, thanks to its n-gram-based approach. Therefore, FastText is used in this study to build a word embedding suitable for Indonesian texts in the context of election-related sentiment analysis.

This research is important as it provides insight into the application of deep learning in political sentiment analysis. It is hoped that the results of this research can assist the KPU in understanding public views on technologies such as Sirekap, thus enabling better decision-making to increase transparency and trust in the electoral process. In addition, the results of this analysis can form the basis for policy-making that is more responsive to public sentiment, thereby strengthening the legitimacy of the electoral process in Indonesia. By providing a better understanding of public opinion, this research has the potential to increase public trust in technology and the political process, which is crucial for democratic stability.

Previous research conducted by Sio Journalist Pipin showed the superiority of Fast R-CNN (94.5% accuracy) over CNN (86%) in the analysis of public opinion on ChatGPT [10]. Dehghani used CNN-LSTM (ParsBERT) for political sentiment analysis in Persian, achieving 89% accuracy on a three-class dataset and 71% on a seven-class dataset [11]. The study conducted by Sadiq, S., Aljrees, T., & Ullah, S. proposed a deepfake text detection method on social media by combining CNN and FastText embeddings. The model was tested on the TweepFake dataset and achieved 93% accuracy, surpassing other machine learning and deep learning methods, including RoBERTa and BERT [12]. The study conducted by Handoko et al. found that the CNN model achieved an accuracy of 85.90% in Sirekap sentiment analysis, outperforming CNN-LSTM (79.91%). These results demonstrate the effectiveness of CNN in election sentiment classification in Indonesia[13]. The study conducted by Mifrah, S., & Benlahmar, E. compares various deep learning models for sentence-level sentiment classification using the Sentiment140 dataset, which contains 1.6 million tweets. The experimental results show that the BERT model achieved the highest accuracy of 87.36%, followed by BiLSTM (79.73%), BiGRU (79.02%), and LSTM (78.64%). The GRU model had the lowest accuracy at 50.03%, This study confirms that transformer-based models like BERT outperform RNN- and CNN-based models in sentiment classification [14]. Based on this research, the combination of FastText with CNN, BiLSTM, and BiGRU is an appropriate approach for sentiment analysis of the SIREKAP application in Google Playstore to improve model accuracy and efficiency.

B. Research Method

The research process aims to collect objective data and information, which will then be used as a reference in the research. With these data, it is expected that the resulting research can have high quality and make a significant contribution to the development of science. This research process can be illustrated in Figure 1.

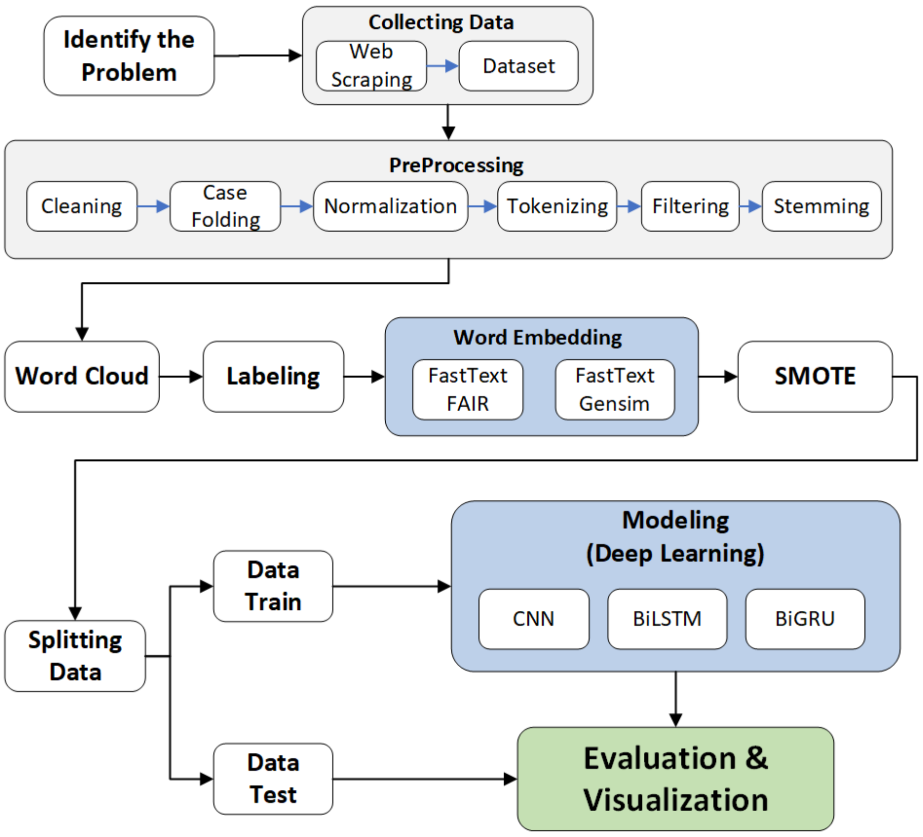


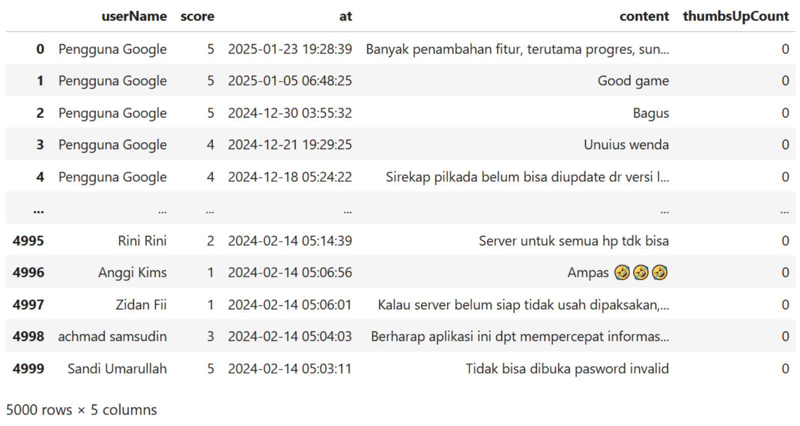
Figure 1. Stages of Research

Identify the Problem

The KPU's use of SIREKAP 2024 in elections faces various challenges, especially in public acceptance, application reliability, and the accuracy of vote recapitulation results. User reviews on Google Playstore show complaints related to performance, ease of use, and trust in election results. In addition, sentiment analysis of user reviews faces the constraints of informal language and Out-of-Vocabulary (OOV) words, so an effective word embedding method such as FastText is needed. A comparison of deep learning models (CNN, BiLSTM, and BiGRU) is also needed to determine the best algorithm to classify sentiment for this application.

Collecting Data

The data used in this research consists of user reviews from the official page of the SIREKAP 2024 application on the Google Play Store (*id.go.kpu.SIREKAP2024*), where many Indonesian users have expressed their opinions about the application. The process used for data collection is web scraping. Figure 2 is the result of the web scraping that has been done. In this study, researchers only used the content column as a dataset.



	userName	score	at	content	thumbsUpCount
0	Pengguna Google	5	2025-01-23 19:28:39	Banyak penambahan fitur, terutama progres, sun...	0
1	Pengguna Google	5	2025-01-05 06:48:25	Good game	0
2	Pengguna Google	5	2024-12-30 03:55:32	Bagus	0
3	Pengguna Google	4	2024-12-21 19:29:25	Unuius wenda	0
4	Pengguna Google	4	2024-12-18 05:24:22	Sirekap pilkada belum bisa diupdate dr versi l...	0
...	...	...	...	...	...
4995	Rini Rini	2	2024-02-14 05:14:39	Server untuk semua hp tdk bisa	0
4996	Anggi Kims	1	2024-02-14 05:06:56	Ampas 🤣🤣🤣	0
4997	Zidan Fii	1	2024-02-14 05:06:01	Kalau server belum siap tidak usah dipaksakan,...	0
4998	achmad samsudin	3	2024-02-14 05:04:03	Berharap aplikasi ini dpt mempercepat informas...	0
4999	Sandi Umarullah	5	2024-02-14 05:03:11	Tidak bisa dibuka pasword invalid	0

5000 rows × 5 columns

Figure 2. Result of web scraping

PreProcessing

Data Preprocessing is an important step in sentiment analysis. This process uses previously collected review data to obtain clean data, which allows word vector generation and sentiment classification to be more accurate [15]. Depending on the data and the model being created, the data preprocessing stages differ for each situation. The following are the preprocessing stages performed:

1. *Cleaning*, unwanted characters such as symbols, italics, punctuation marks, URLs, and unnecessary special characters are removed from the text by performing cleaning [16], [17]. Improving data quality and enabling proper sentiment analysis are the two main goals of data cleaning.
2. *Case Folding*, The process of converting all characters in the text to lowercase. Irregular formatting can cause problems in data analysis, so this step is almost always applied in text processing. By applying case folding, inconsistencies due to differences in the use of uppercase and lowercase letters can be minimized, thus improving accuracy in data analysis and modeling [18].
3. *Normalization*, INANLP dictionary and Colloquial Indonesian Lexicon dictionary, which can be obtained through GitHub, are important parts of the normalization phase of this research. Both dictionaries were used as data dictionary types in the Python code during this operation. The basic idea behind this normalization procedure is that each token in the data will be converted to its base form if it corresponds to a word in the dictionary [19].
4. *Tokenizing*, A text is divided into separate words, or tokens. Since key units (tokens) can be identified, this approach simplifies text processing by breaking sentences into smaller parts [20].
5. *Filtering*, Removal of stop words and other redundant words, is done after tokenization. To highlight more important terms, this technique focuses on removing common words that do not add to the sentiment analysis [15]. By doing this, we hope to get more accurate sentiment results.
6. *Stemming*, Transforming the words in the text into their base word or basic form, is the last phase. In stemming, different word endings are removed, and words are reduced to their most basic form, which is referred to as "stem" or "root" [21]. In natural language processing, stemming is often used to simplify text and make text analysis easier, including document classification and data retrieval. The Sastrawi algorithm will be used in this study.

Word Cloud

A word cloud is a visual representation of text that displays words of varying sizes, where the size of each word reflects its frequency or significance in the text dataset. The more frequently a word appears, the larger it appears in the word cloud [22]. This technique is very useful in text analysis to visually identify dominant words. And from this technique the researcher can see whether or not there are still unnecessary words.

Labeling

The labeling performed is manual, where a collection of words (lexicon) with predefined sentiment values are used in a lexicon-based labeling approach, a sentiment analysis technique, to determine the sentiment of the text under study [23]. In this case, the labeling assigns neutral, negative, and positive values to the text.

Word Embedding

To represent text into numerical form, this research utilizes the FastText-based text insertion method. This method was chosen for its ability to capture semantic information from words, including words that are not in the vocabulary (OOV), through the use of n-gram characters [24]. The two FastText implementations used are Facebook AI Research and Gensim, where these two Word Embedding libraries are different.

SMOTE

The problem of class imbalance in datasets is addressed by a method called SMOTE (Synthetic Minority Over-sampling Technique). When a class contains fewer samples than other classes, it is considered imbalanced, which may result in machine learning models often ignoring minority classes. We can improve the performance of the model on unbalanced datasets and guarantee that the model produces more equal predictions for minority classes by incorporating SMOTE into deep learning [25].

Splitting Data

Data Splitting is the process of separating a dataset into different subsets for testing and training machine learning or deep learning models. Since it ensures that the developed model can be reliably assessed and has good generalization ability on unknown data, this stage is very important in the machine learning workflow. The train_test_split function in the scikit-learn library is often used to split the data into training and testing sets randomly. Using this method, you can specify parameters such as train_size and test_size to find out how big these split sets are. For example, if test_size = 0.25, then 75% of the data will go to the training set and 25% to the testing set. Furthermore, the stratification parameter helps in maintaining the same class distribution in the training and testing sets, and the random_state parameter can be used to guarantee the reproducibility of the division [26].

Modeling (Deep Learning)

The accuracy performance of three models - Convolutional Neural Networks (CNN), Bidirectional Long Short-Term Memory (BiLSTM), and Bidirectional Gated Recurrent Unit (BiGRU) - are compared in this study. CNN neural network architecture is well known for its ability to interpret grid-like data structures, such as images or sequential data in matrix form. To hierarchically extract features from input data, these networks use convolutional, pooling, and fully connected layers [27]. In contrast, BiLSTM and BiGRU are variations of Recurrent Neural Network (RNN) intended to capture temporal relationships in sequential data. BiLSTM is able to more thoroughly understand context across sequences by processing data both forward and backward. BiGRU is a compact, parameterized form of BiLSTM that is computationally efficient yet able to accurately predict temporal patterns [28]. To compare the accuracy of the three models, they will be applied separately to the same dataset. Finding the best and most efficient model to solve the problem in the research domain is the goal of this comparison.

Evaluation and Visualization

An important phase in deep learning modeling is evaluation and visualization. The shift in class distribution before and after data balancing is depicted in a bar graph displaying the impact before and after the SMOTE technique [29]. The performance comparison of CNN, BiLSTM, and BiGRU helps in choosing the model that best suits the data and the purpose of the analysis. FastText FAIR and FastText Gensim will be used to test these three models. By showing the proportion of accurate and inaccurate predictions, the Confusion Matrix will provide an overview of how well the classification model performs [30].

Several assessment criteria widely used in data analysis and machine learning will be used to assess the models used in this study. The steps that make up the evaluation process are as follows [31]:

a. **Accuracy** shows the proportion of accurate forecasts among all forecasts. It provides a summary of the model's effectiveness.

$$Accuracy = \frac{True\ Positive + True\ Negative}{Total\ Sample} \quad (1)$$

b. **Precision** measures how many positive predictions are correct compared to all the positive predictions the model makes.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (2)$$

c. **Recall** measures how much positive data was successfully detected compared to the total positive data.

$$Recall = \frac{True\ Positive}{True\ Poitive + False\ Negative} \quad (3)$$

d. **F1-Score** is the harmonic mean between Precision and Recall, which provides a balance between the two

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

C. Result and Discussion

In this study, the researcher used Python programming language and FastText pre-trained word embedding for feature extraction, Deep Learning approach was applied in this study. The tools used in the research include an Intel Xeon W-2223 processor with a speed of 3.60GHz (8 CPUs), an 8GB NVIDIA Quadro T1000 GPU, and 32 GB RAM. The evaluation techniques used include training datasets, model evaluation, and score metrics to ensure the performance of the proposed model can run optimally.

Dataset

After the preprocessing process, the number of valid data was recorded as 4877 data. Next, a word cloud is done to see if there are still words that do not affect the deep learning algorithm. Figure 3 is the result of the word cloud after preprocessing. It can be seen that there is still the word 'tidak', because the word no is very influential in this research, so the word 'tidak' is not deleted.

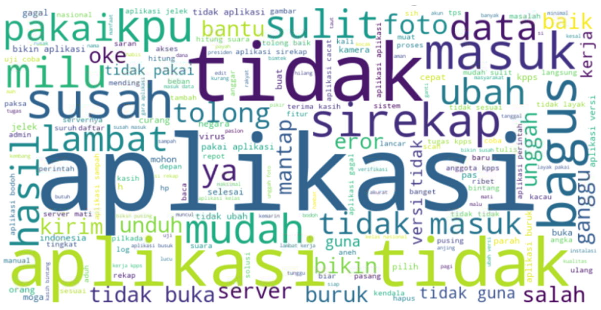


Figure 3. Word cloud after preprocessing

Next, labelling is done with positive, negative, and neutral categories. The results of label distribution can be seen in Figure 4. In this study, researchers only used data that was labelled positive and negative. Figure 5 shows the condition after the data with neutral labels are removed. Thus, after removing the neutral label, the number of datasets used becomes 4557 data.

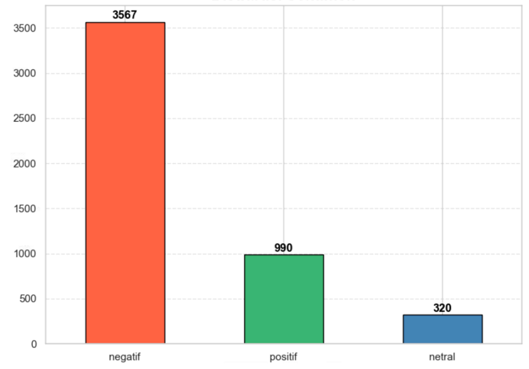


Figure 4. Distribution of data labelling

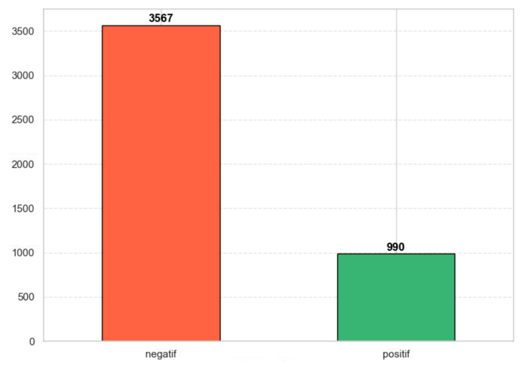


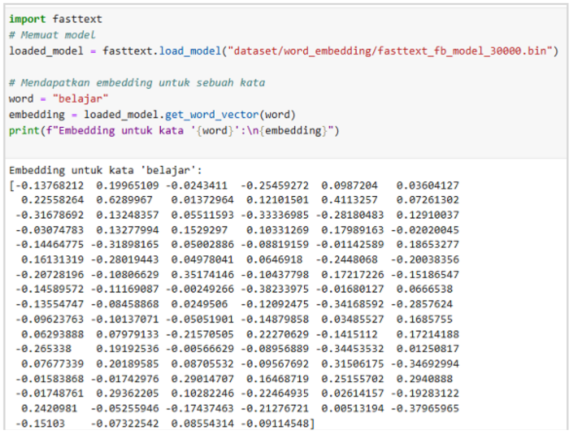
Figure 5. After neutral labelled data is removed

Figures 6 and 7 show the embedding results of the word ‘belajar’ obtained from the models that have been trained using two different FastText libraries. Before the embedding process is performed, the data first goes through a pre-processing stage to ensure the quality of the data used in the model training. In addition, the researcher has set the dimension of the embedding vector at 100 so that the resulting word representation has an optimal feature depth.Table 1 shows the parameters for both FastText libraries.

Table 1. Parameters of both FastText

Aspect	FastText (Facebook)	FastText (Gensim)
Library	fasttext (Facebook)	gensim.models.FastText
Preprocessing	Preformatted text for FastText	Tokenization using nltk.word_tokenize()
Model Type	fasttext.train_unsupervised(input="FastText_FB.txt", model="skipgram")	FastText(sentences=tokenized_docs, ...)
Vector Size	100	100
Window	Not explicitly defined (default Facebook FastText)	5
Min Count	2	2
Training Type	Unsupervised Skip-gram (Facebook FastText)	Unsupervised Skip-gram (gensim)
Epochs	25	25
Unsupervised Training	Yes	Yes

The analysis of the embedding results aims to determine the differences in word representations generated by each library and evaluate the extent to which the model can capture the semantic meaning of the word ‘belajar’.

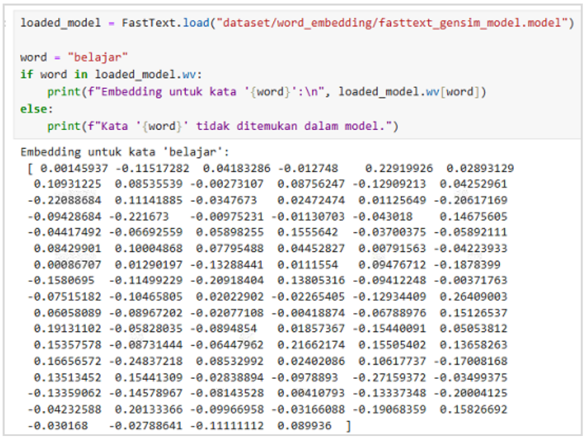


```
import fasttext
# Memuat model
loaded_model = fasttext.load_model("dataset/word_embedding/fasttext_fb_model_30000.bin")

# Mendapatkan embedding untuk sebuah kata
word = "belajar"
embedding = loaded_model.get_word_vector(word)
print(f"Embedding untuk kata '{word}':\n{embedding}")
```

```
Embedding untuk kata 'belajar':
[-0.13768212  0.19965109 -0.0243411  -0.25459272  0.0987204   0.03604127
  0.22558264  0.6289967   0.01372964  0.12101501  0.4113257   0.07261302
 -0.31678692  0.13248357  0.05511593 -0.33336985 -0.28180483  0.12910037
 -0.03074783  0.13277994  0.1529297   0.10331269  0.17989163 -0.02020045
 -0.14464775 -0.31898165  0.05002886 -0.08819159 -0.01142589  0.18653277
  0.16131319 -0.28019443  0.04978041  0.0646918  -0.2448068  -0.20038356
 -0.20728196 -0.10806629  0.35174146 -0.10437798  0.17217226 -0.15186547
 -0.14589572 -0.11169087 -0.00249266 -0.38233975 -0.01680127  0.0666538
 -0.13554747 -0.08458868  0.0249506  -0.12092475 -0.34168592 -0.2857624
 -0.09623763 -0.10137071 -0.05051901 -0.14879858  0.03485527  0.1685755
  0.06293888  0.07979133 -0.21570505  0.22270629 -0.1415112   0.17214188
 -0.265338    0.19192536 -0.00566629 -0.08956889 -0.34453532  0.01250817
  0.07677339  0.20189585  0.08705532 -0.09567692  0.31506175 -0.34692994
 -0.01583868 -0.01742976  0.29014707  0.16468719  0.25155702  0.2940888
 -0.01748761  0.29362205  0.10282246 -0.22464935  0.02614157 -0.19283122
  0.2420981  -0.05255946 -0.17437463 -0.21276721  0.00513194 -0.37965965
 -0.15103    -0.07322542  0.08554314 -0.09114548]
```

Figure 6. Embedding with FAIR



```
loaded_model = FastText.load("dataset/word_embedding/fasttext_gensim_model.model")

word = "belajar"
if word in loaded_model.wv:
    print(f"Embedding untuk kata '{word}':\n", loaded_model.wv[word])
else:
    print(f"Kata '{word}' tidak ditemukan dalam model.")
```

```
Embedding untuk kata 'belajar':
 [ 0.00145937 -0.11517282  0.04183286 -0.012748    0.22919926  0.02893129
  0.10931225  0.08535539 -0.00273107  0.08756247 -0.12909213  0.04252961
 -0.22088684  0.11141885 -0.0347673   0.02472474  0.01125649 -0.20617169
 -0.09428684 -0.221673   -0.00975231 -0.01130703 -0.043018    0.14675605
 -0.04417492 -0.06692559  0.05898255  0.1555642  -0.03700375 -0.05892111
  0.08429901  0.10004868  0.07795488  0.04452827  0.00791563 -0.04223933
  0.00086707  0.01290197 -0.13288441  0.0111554   0.09476712 -0.1878399
 -0.1580695  -0.11499229 -0.20918404  0.13805316 -0.09412248 -0.00371763
 -0.07515182 -0.10465805  0.02022902 -0.02265405 -0.12934409  0.26409003
  0.06058089 -0.08967202 -0.02077108 -0.00418874 -0.06788976  0.15126537
  0.19131102 -0.05828035 -0.0894854   0.01857367 -0.15440091  0.05053812
  0.15357578 -0.08731444 -0.06447962  0.21662174  0.15505402  0.13658263
  0.16656572 -0.24837218  0.08532992  0.02402086  0.10617737 -0.17008168
  0.13513452  0.15441309 -0.02838894 -0.0978893  -0.27159372 -0.03499375
 -0.13359062 -0.14578967 -0.08143528  0.00410793 -0.13337348 -0.20004125
 -0.04232588  0.20133366 -0.09966958 -0.03166088 -0.19068359  0.15826692
 -0.030168   -0.02788641 -0.11111112  0.089936  ]
```

Figure 7. Embedding with Gensim

The two figures above show that the differences in the resulting vector values are due to differences in the implementation of model training in each library. Each library has its own approach in calculating word representation, including the learning process, n-gram processing, and weighting strategy, which ultimately results in differences in the vector representation of the same word.

Datasets that have been manually categorised into positive and negative labels do not seem to be directly comparable. To make the labelled datasets more comparable, the SMOTE (Synthetic Minority Over-sampling Technique) technique is used to perform resampling. Figure 8 shows the aftermath of SMOTE.

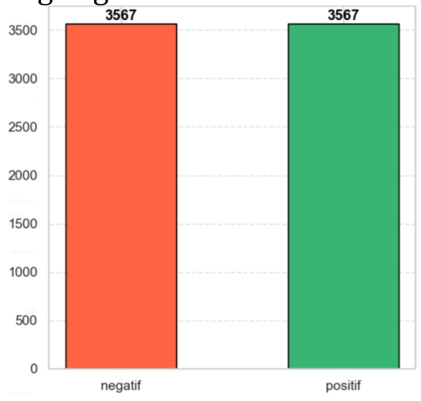


Figure 8. Data after SMOTE

After applying SMOTE, the data is divided by different proportions, namely 60%, 70%, 80%, and 90%. Table 1 shows the number of training samples after each division.

Table 2. Splitting Data Train and Test

Splitting Data (%)		Ammount of Data	
Train	**Test**	**Train**	**Test**
90%	10%	6420	714
80%	20%	5707	1427
70%	30%	4993	2141
60%	40%	4280	2854

Evalutaion Models

The initial stage in the model testing procedure is to determine the hyperparameter settings for epoch, unit, batch size, dropout, and optimiser method. The final results of hyperparameter testing for CNN, BiLSTM, and BiGRU architectures are shown in Table 2.

Table 3. Hyperparameters of the Models

Model	*Epoch*	*Unit*	*Batch Size*	*Drop Out*	*Opt*	*Regulation*	*Activation*	*Los Function*
CNN	30	64,16	64	0.5	Nadam	L2 (0.01)	ReLu, Softmax	Categorical Crossentropy
BiLSM	30	64,16	64	0.5	Nadam	L2 (0.01)	ReLu, Softmax	Categorical Crossentropy
BiGRu	30	64,16	64	0.5	Nadam	L2 (0.01)	ReLu, Softmax	Categorical Crossentropy

To evaluate the performance of the model during training, tests were conducted with various test set sizes, as well as a comparison between FastText FAIR and FastText Gensim combined with the deep learning model. Table 3 presents the results for test set sizes of 10%, 20%, 30%, and 40% on each architecture.

Table 4. Accuracy of all Models

Test Size	*Accurary FastText FAIR* (%)			*Accuracy FastText Gensim* (%)		
	CNN	**BiLSTM**	**BiGRU**	**CNN**	**BiLSTM**	**BiGRU**
10%	94.40	91.32	93.84	95.94	95.38	95.52
20%	92.57	91.52	93.69	95.80	95.16	95.80
30%	92.95	90.52	91.97	95.98	94.54	95.05
40%	92.22	89.59	92.08	94.88	94.43	94.78

In Table 3, it can be seen that both FastText word embedding methods show good accuracy. However, FastText Gensim shows superior performance when applied to deep learning models compared to FastText FAIR. In terms of accuracy, CNN model shows better results compared to BiLSTM and BiGRU models. Despite the comparison using FastText with two different libraries, the CNN model still shows superior performance.

Table 5. Highest Accuracy Taken from 2 FastText

Scenario	**Test Size**	***Accuracy***	***Recall***	***Precision***	***F1-Score***	**Time Training/sec**
FAIR + CNN	10%	94.40	94.36	94.47	94.39	16.49
Gensim + CNN	30%	95.98	95.98	95.98	95.98	14.67

Researchers took the 2 highest accuracies from each FastText, seen in Table 4 that the highest was Gensim + CNN with 30% test data. When viewed from the Training Time (sec) that FAIR + CNN is faster than Gensim + CNN, because the two FastTexts have different Test Size.

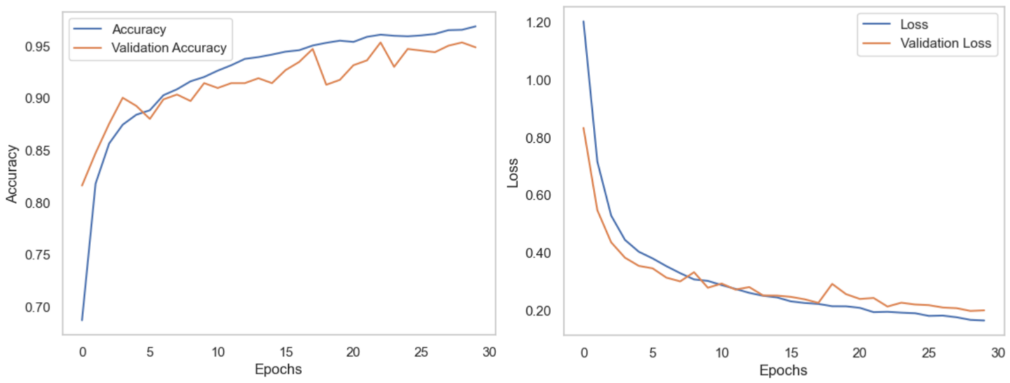


Figure 9. FAIR + CNN Training for Accuracy and Loss Validation

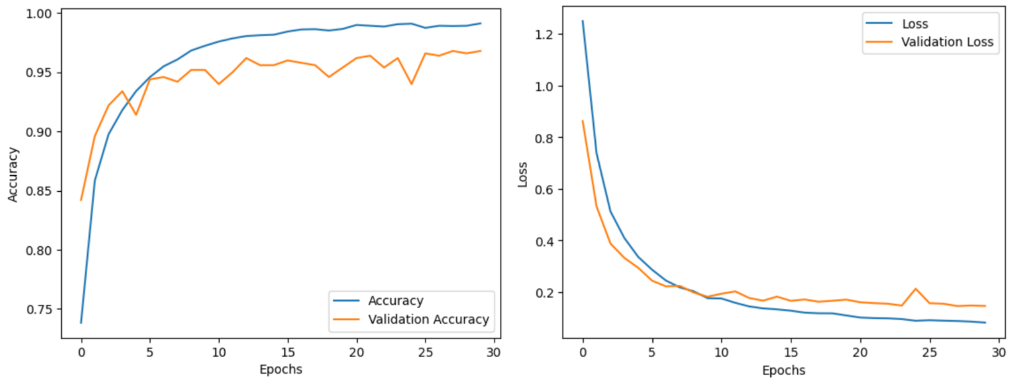


Figure 10. Gensim + CNN Training for Accuracy and Loss Validation

Figure 9 and Figure 10 show the model training process using the previously prepared training data. Both model trainings show good results, but training with Gensim + CNN is slightly superior to training with FAIR + CNN based on accuracy stability and faster loss reduction.

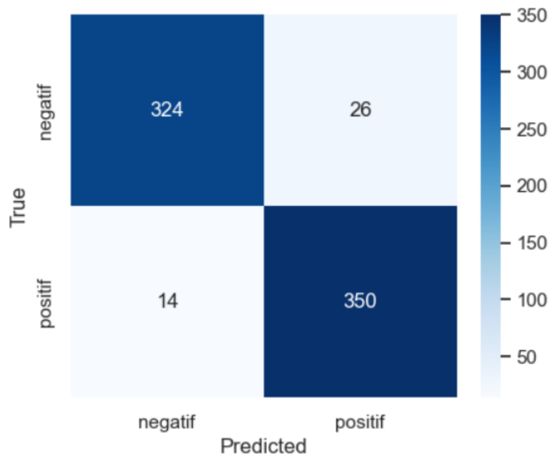


Figure 11. Confusion Matrix FAIR + CNN

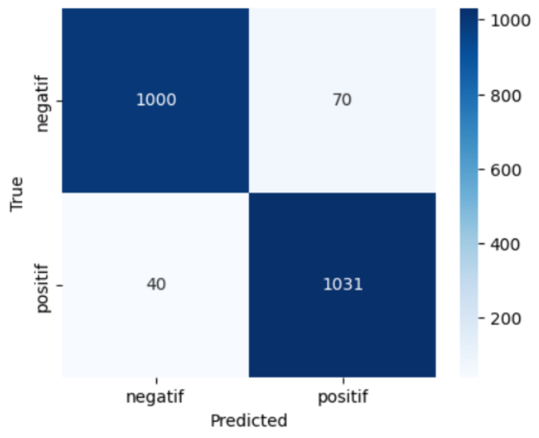


Figure 12. Confusion Matrix Gensim + CNN

Figures 11 and 12 show the Confusion matrix of the two models. Overall, the Gensim + CNN model shows a more stable and reliable performance on larger datasets. However, both models already show excellent performance with accuracy above 90%

D. Conclusion

This research has analysed public sentiment towards the SIREKAP application used in elections in Indonesia. By applying deep learning methods such as CNN, BiLSTM, and BiGRU and using FastText as a word embedding technique, this research successfully evaluated the performance of the model in sentiment classification. The results showed that FastText Gensim with CNN achieved the highest accuracy of 95.98%, while FastText FAIR with CNN achieved 94.40% accuracy. CNN model shows more stable performance than BiLSTM and BiGRU in sentiment classification.

The implications of this research show that the combination of FastText Gensim and CNN can be an effective approach in political sentiment analysis in Indonesia, especially in understanding public opinion towards the use of election technology such as SIREKAP. With these results, it is expected that the General Election Commission (KPU) can use insights from sentiment analysis to improve transparency, public trust, as well as the effectiveness of digital system implementation in the upcoming election process.

Future research can expand the scope by considering additional factors such as temporal aspects in voter sentiment, integration of transformer models such as BERT, as well as multimodal (text, image, and video) based sentiment analysis to gain a more comprehensive understanding of public opinion.

E. References

[1] "KPU - PAGE." Accessed: Mar. 07, 2024. [Online]. Available: https://www.kpu.go.id/page/read/12/pemilu-dalam-sejarah

[2] "KPU - PAGE." Accessed: Mar. 10, 2024. [Online]. Available: https://www.kpu.go.id/page/read/5/tugas-dan-kewenangan

[3] "Manfaatkan Sirekap, Transparan dan Kemudahan untuk Masyarakat - KPU." Accessed: Mar. 10, 2024. [Online]. Available: https://www.kpu.go.id/berita/baca/10143/manfaatkan-sirekap-transparan-dan-kemudahan-untuk-masyarakat

[4] M. K. Anam, M. I. Mahendra, W. Agustin, R. Rahmaddeni, and N. Nurjayadi, "Framework for Analyzing Netizen Opinions on BPJS Using Sentiment Analysis and Social Network Analysis (SNA)," *INTENSIF J. Ilm. Penelit. dan Penerapan Teknol. Sist. Inf.*, vol. 6, no. 1, pp. 11–28, 2022, doi: 10.29407/intensif.v6i1.15870.

[5] A. Z. R. Adam and E. B. Setiawan, "Social Media Sentiment Analysis using Convolutional Neural Network (CNN) dan Gated Recurrent Unit (GRU)," *J. Ilm. Tek. Elektro Komput. dan Inform.*, vol. 9, no. 1, pp. 119–131, 2023, doi: 10.26555/jiteki.v9i1.25813.

[6] Y. Cheng *et al.*, "Sentiment Analysis Using Multi-Head Attention Capsules with Multi-Channel CNN and Bidirectional GRU," *IEEE Access*, vol. 9, pp. 60383–60395, 2021, doi: 10.1109/ACCESS.2021.3073988.

[7] M. A. Kausar, A. Soosaimanickam, and M. Nasar, "Public Sentiment Analysis on Twitter Data during COVID-19 Outbreak," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 2, pp. 415–422, 2021, doi: 10.14569/IJACSA.2021.0120252.

[8] N. Badri, F. Kboubi, and A. H. Chaibi, "Combining FastText and Glove Word Embedding for Offensive and Hate speech Text Detection," *Procedia Comput. Sci.*, vol. 207, no. Kes, pp. 769–778, 2022, doi: 10.1016/j.procs.2022.09.132.

[9] S. Ghosal and A. Jain, "Depression and Suicide Risk Detection on Social Media using fastText Embedding and XGBoost Classifier," *Procedia Comput. Sci.*, vol. 218, pp. 1631–1639, 2022, doi: 10.1016/j.procs.2023.01.141.

[10] M. Daffa Dhiyaulhaq, S. Jurnalis Pipin, F. Mikael Sinaga, S. Winardi, and M. Noor Hakim, "Sentiment Analysis Classification of ChatGPT on Twitter Big Data in Indonesia Using Fast R-CNN," *Media Inform. Budidarma*, vol. 7, no. 4, pp. 2137–2148, 2023, doi: 10.30865/mib.v7i4.6816.

[11] M. Dehghani and Z. Yazdanparast, "Political Sentiment Analysis of Persian Tweets Using CNN-LSTM Model," 2023, doi: 10.48550/arXiv.2307.07740.

[12] F. M. Shiri, T. Perumal, N. Mustapha, and R. Mohamed, "A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU," no. Ml, pp. 1–61, 2023, [Online]. Available: http://arxiv.org/abs/2305.17473

[13] H. Handoko, A. Asrofiq, J. Junadhi, and A. S. Negara, "Sentiment Analysis of Sirekap Tweets Using CNN Algorithm," *INTENSIF J. Ilm. Penelit. dan Penerapan Teknol. Sist. Inf.*, vol. 8, no. 2, pp. 312–329, 2024, doi: 10.29407/intensif.v8i2.23046.

[14] S. Mifrah and E. H. Benlahmar, "Sentence-Level Sentiment Classification A Comparative Study between Deep Learning Models," *J. ICT Stand.*, vol. 10, no. 2, pp. 339–352, 2022, doi: 10.13052/jicts2245-800X.10213.

[15] S. Khairunnisa, A. Adiwijaya, and S. Al Faraby, "Pengaruh Text Preprocessing terhadap Analisis Sentimen Komentar Masyarakat pada Media Sosial Twitter (Studi Kasus Pandemi COVID-19)," *J. Media Inform. Budidarma*, vol. 5, no. 2, p. 406, 2021, doi: 10.30865/mib.v5i2.2835.

[16] B. Kanwal *et al.*, "Opinion Mining from Online Travel Reviews: An Exploratory Investigation on Pakistan Major Online Travel Services Using Natural Language Processing," *IEEE Access*, vol. 11, no. March, pp. 29934–29945, 2023, doi: 10.1109/ACCESS.2023.3260114.

[17] M. J. Tan and C. H. Guan, "Are people happier in locations of high property value? Spatial temporal analytics of activity frequency, public sentiment and housing price using twitter data," *Appl. Geogr.*, vol. 132, p. 102474, 2021, doi: 10.1016/j.apgeog.2021.102474.

[18] A. G. Gozal, H. Pranoto, and M. F. Hasani, "Sentiment analysis of the Indonesian community toward face-to-face learning during the Covid-19 pandemic," *Procedia Comput. Sci.*, vol. 227, pp. 398–405, 2023, doi: 10.1016/j.procs.2023.10.539.

[19] F. Zuhad and N. Wilantika, "Perbandingan Penggunaan Kamus Normalisasi dalam Analisis Sentimen Berbahasa Indonesia," *J. Linguist. Komputasional*, vol. 5, no. 1, pp. 13–23, 2022, doi: 10.26418/jlk.v5i1.60.

[20] A. Erkan and T. Gungor, "Analysis of Deep Learning Model Combinations and Tokenization Approaches in Sentiment Classification," *IEEE Access*, vol. 11, no. December, pp. 134951–134968, 2023, doi: 10.1109/ACCESS.2023.3337354.

[21] M. Young *et al.*, "Natural language processing to assess the epidemiology of delirium-suggestive behavioural disturbances in critically ill patients," *Crit. Care Resusc.*, vol. 23, no. 2, pp. 144–153, 2021, doi: 10.51893/2021.2.oa1.

[22] S. In Jung, S. Dong Ho, and J. Kim, "Word Cloud Techniques for Data Analysis," no. 2, pp. 278–283, 2024, doi: 10.46254/ev01.20230123.

[23] H. S. Hota, D. K. Sharma, and N. Verma, *Lexicon-based sentiment analysis using Twitter data*. Elsevier Inc., 2021. doi: 10.1016/B978-0-12-824536-1.00015-0.

[24] I. N. Khasanah, "Sentiment Classification Using fastText Embedding and Deep Learning Model," *Procedia CIRP*, vol. 189, pp. 343–350, 2021, doi: 10.1016/j.procs.2021.05.103.

[25] D. Dablain, B. Krawczyk, and N. V. Chawla, "DeepSMOTE: Fusing Deep Learning and SMOTE for Imbalanced Data," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 34, no. 9, pp. 6390–6404, 2023, doi: 10.1109/TNNLS.2021.3136503.

[26] "Splitting Datasets With scikit-learn and train_test_split() (Overview) – Real Python." Accessed: May 25, 2024. [Online]. Available:

https://realpython.com/lessons/splitting-datasets-overview/

[27] Yudi Widhiyasana, Transmissia Semiawan, Ilham Gibran Achmad Mudzakir, and Muhammad Randi Noor, "Penerapan Convolutional Long Short-Term Memory untuk Klasifikasi Teks Berita Bahasa Indonesia," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 10, no. 4, pp. 354–361, 2021, doi: 10.22146/jnteti.v10i4.2438.

[28] M. R. R. Rana, A. Nawaz, T. Ali, A. M. El-Sherbeeny, and W. Ali, "A BiLSTM-CF and BiGRU-based Deep Sentiment Analysis Model to Explore Customer Reviews for Effective Recommendations," *Eng. Technol. Appl. Sci. Res.*, vol. 13, no. 5, pp. 11739–11746, 2023, doi: 10.48084/etasr.6278.

[29] Y. Bao and S. Yang, "Two Novel SMOTE Methods for Solving Imbalanced Classification Problems," *IEEE Access*, vol. 11, no. December 2022, pp. 5816–5823, 2023, doi: 10.1109/ACCESS.2023.3236794.

[30] M. Heydarian, T. E. Doyle, and R. Samavi, "MLCM: Multi-Label Confusion Matrix," *IEEE Access*, vol. 10, pp. 19083–19095, 2022, doi: 10.1109/ACCESS.2022.3151048.

[31] Ç. Oğuz and M. Yağanoğlu, "Detection of COVID-19 using deep learning techniques and classification methods," *Inf. Process. Manag.*, vol. 59, no. 5, pp. 1–18, 2022, doi: 10.1016/j.ipm.2022.103025.