Jurnal Teknik Informatika (JUTIF)
Vol. 5, No. 6, December 2024, pp. 1581-1589

DOI: https://doi.org/10.52436/1.jutif.2024.5.6.2498
p-ISSN: 2723-3863
e-ISSN: 2723-3871

WORD EMBEDDING OPTIMIZATION IN SENTIMENT ANALYSIS OF REVIEWS ON MYTELKOMSEL APP USING LONG SHORT-TERM MEMORY AND SYNTHETIC MINORITY OVER-SAMPLING TECHNIQUE

Muhammad Raffif Haziq[1], Yuliant Sibaroni[*2], Sri Suryani Prasetyowati[3]

[1,2,3]Informatics, School of Computing, Telkom University, Indonesia
Email: [1]raffifhzq@student.telkomuniversity.ac.id, [2]yuliant@telkomuniversity.ac.id, [3]srisuryani@telkomuniversity.ac.id

(Article received: July 11, 2024; Revision: August 06, 2024; published: December 29, 2024)

Abstract

Telkomsel is one of the internet service provider companies that has a mobile-based application called MyTelkomsel which functions to facilitate users in conducting online services independently. Users of the application certainly have their own responses about the application, so that users can provide responses to the application. Therefore, sentiment analysis can be one of the solutions to find out public sentiment towards the application. In this research, the author builds a system for sentiment analysis using word embedding Word2vec, GloVe, FastText to get word representation in vector form with classification using Long Short-Term Memory (LSTM) combined with Synthetic Minority Over-sampling Technique (SMOTE) which can handle data imbalance. The data used comes from user reviews of the MyTelkomsel application found on the Google Play Store. This study compares the performance of several word embedding in LSTM and LSTM-SMOTE classifiers. The results showed the results show that the performance of three-word embedding on the LSTM model is superior compared to the LSTM-SMOTE model. Overall, it was found that the combination of FastText and LSTM gave the best performance compared to the other five combinations with an accuracy value of 89.11%.

Keywords: *FastText, GloVe, LSTM-SMOTE, Sentiment analysis, Word Embedding, Word2vec.*

1. INTRODUCTION

Telkomsel is one of the leading internet service provider telecommunications companies in Indonesia that created a mobile-based application called MyTelkomsel. The Mytelkomsel application was launched by Telkomsel commercially as a digital channel with the aim of providing convenience for customers in conducting online self-service and it is hoped that users will freely interact with the application [1][2]. Along with the development of the MyTelkomsel application, which is widely used by the community, it has resulted in the emergence of reviews from its users. Therefore, an approach using sentiment analysis is needed. Where the main purpose of sentiment analysis is to determine the point of view and categorize the emotions or opinions of users about something [3], [4], [5]. This can help to measure user sentiment towards the application.

Research on the sentiment of MyTelkomsel application users has been conducted by R. Indra Kurnia and A. Suganda Girsang in [6]. The research used Word2vec word embedding and LSTM. The amount of data used was 16,359 data divided into 80:20 for training data and test data. The best f1-score and precision results were obtained in 5 classes using LSTM-SMOTE, namely 0.62 and 0.70. While the best f1-score and precision results in 3 classes using LSTM-SMOTE, namely 0.86 and 0.87.

The addition of the SMOTE algorithm can affect the accuracy value because it increases the accuracy value in each experiment. Word2vec is one of the word embedding methods created by Tomas Mikolov from the Google team [7]. Word2Vec technique is made for word embedding with two models: Skip-Gram and Continuous Bag of Words (CBOW). Research [8] shows that the use of the CBOW algorithm gives good results on news articles and the Skip-Gram algorithm gives better results on tweets.

There is another popular word embedding technique used in research, namely GloVe word embedding. Pennington, Socher and Manning introduced GloVe in 2014, where the GloVe technique is a new method for matching words that uses the probability ratio of co-occurrence between words [9]. Two classes of word representations are used in this model, namely global matrix factorization and Skip-Gram, which are used to extract better features by examining the relationship between words [10]. In research [11] conducted a comparison of word embedding methods for sentiment classification on product review data. The data used contains 19,977 data and is divided into 11,986 training data and 7,991 test data. This research uses CNN for sentiment classification by comparing three word embedding, namely Word2vec, GloVe, FastText to find which one is the best word embedding in this

classification. The accuracy results obtained from this research are 92.5% on Word2vec, 95.8% on GloVe, 97.2% on FastText. It was found that FastText word embedding is superior to other word embedding in classifying using CNN.

Bojanowski et al. developed the FastText technique in 2017, a refinement of the skip-gram model used in the Word2vec method, where this FastText approach analyses word representation by considering subword information from the word [12]. In research [13], a comparison of deep learning methods in text classification using CNN, RNN, and LSTM with word embedding Word2vec and FastText was conducted. It was found that the combination of FastText and LSTM had the best accuracy value with a value of 97.86%. This finding indicates that the combination is superior in handling text classification tasks compared to other combinations. In [14], [15] it was also proven that the performance of LSTM models is often superior to CNN and RNN in text classification tasks.

In this study, the authors compare several word embedding models such as Word2vec, GloVe, and FastText for optimizing word embeddings in classifiers using LSTM and LSTM combined with the SMOTE algorithm to address imbalance. This research is novel and has not been previously conducted. The dataset used in this study consists of user reviews of the MyTelkomsel application in Indonesian language from the Google Play Store. The dataset exhibits a tendency towards one category and other categories are less represented with an imbalance in the number of positive and negative sentiments.

2. RESEARCH METHOD

This research is made with a review sentiment analysis system on the MyTelkomsel application by comparing the performance of word embedding Word2vec, Glove, FastText on classifiers using LSTM and LSTM-SMOTE. The flow of the sentiment analysis system design can be seen in Figure 1.

2.1. Scraping Data

The data collection process in this study was carried out by taking data from user reviews of the MyTelkomsel application found on the Google Play Store. Data collection is done by data scraping method. This research uses 9000 review data as a dataset. The dataset from the scraping results is directly saved into a file in Comma Separated Values (CSV) format.

2.2. Labeling Data

During the data labeling stage, reviews were manually labeled by three individuals and categorized into two categories: positive and negative. Content perceived as supportive, beneficial, containing praise, and indicating high satisfaction were labeled as positive. On the other hand, content perceived as opposing, detrimental, containing complaints, and indicating low satisfaction were labeled as negative. The dataset consisted of 3,204 positive reviews and 5,796 negative reviews, totaling 9,000 reviews in all. An example of data labeling can be seen in Table 1.

Table 1. Manual Labeling Dataset

Text	Sentiment
Sering sekali gangguan ga jelas, disaat mau beli paket susah sekali, makin lama makin sebel pakek mytelkomsel	Negative
aplikasi nya sungguh sungguh bikin kecewa saya tidak bisa buka aplikasi mytelkomsel ini gimna cerita nya mau ngisi paket lwt aplk kalau aplksinya aja tidak bisa di buka aplksinya aneh tolong di perbaiki segera saya pelanggan telkomsel sudah hampir 25 tahun.	Negative
Mytelkomsel merupakan produk yang memberi rasa puas buat Penggunanya karena banyak memberikankan banyak pilihan dan kemudahan,lebih tingkatkan kinerja dan pelayanan...Sukses selalu.	Positive
MyTelkomsel memudahkan pelanggan dlm melakukan prmbelian paket data/nelpon,trm ksh MyTelkomsel	Positive

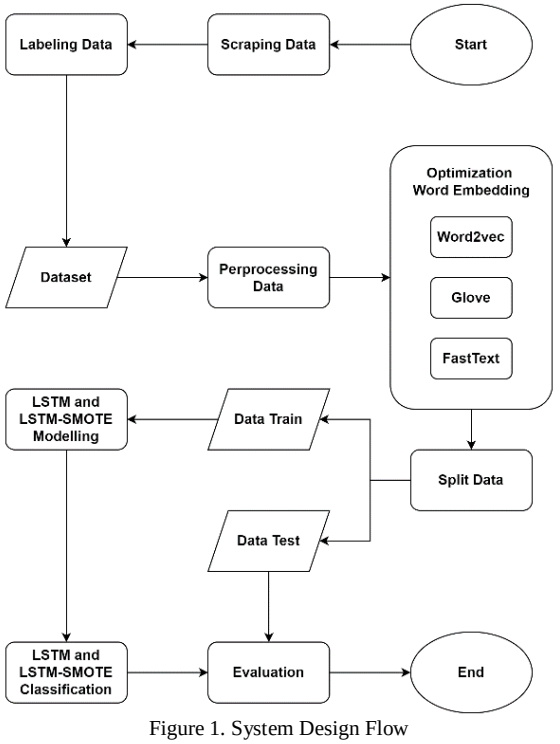

Figure 1. System Design Flow

2.3. Preprocessing Data

This stage was conducted to facilitate the model processing in subsequent steps and enhance the dataset's quality. The research implemented several preprocessing steps such as data cleaning, stemming, data normalization, stopword removal, tokenization, and case folding.

Data cleaning: At this stage, cleaning is done on data that contains things that are not needed or affect sentiment such as mentions (@), numbers,

operator operations, hashtags (#), URLs, special characters, and double spaces.

Stemming: At this stage, the removal of prefixes, endings, or a combination of both in a word is carried out to get the base word of the word. This stemming process is carried out using "StemmerFactory" from the Sastrawi library to get the base word of each existing word.

Data normalization: At this stage, cross-word correction or over-writing of words that are not standardized into words that are spelled and standardized according to KBBI standards.

Stopword removal: At this stage, words that appear frequently and do not have significant information in the text are removed. This process includes the removal of adverbs and conjunctions. This research uses a stopword dictionary containing 679 words made by Tala, F. Z. in research [16]. Examples of words removed in this process such as "me", "yg", "will", "following", "dong", "like", "why", "always", "general", "for example", etc.

Tokenization: In this stage, a text or sentence is segmented into words that are referred to as tokens. This segmentation process uses separation rules such as spaces, punctuation, or dashes.

Case folding: At this stage, all letters in the text are converted into the same format, by converting uppercase letters into lowercase letters.

2.4. Optimization Word Embedding

Data that has passed the preprocessing stage then enters the word embedding stage to get a representation of words in vector form. There are three word embeddings used for comparison in this research: Word2vec, GloVe, and FastText.

Word2vec: Word2vec is a Natural Language Processing (NLP) technology that uses a text corpus as input and represents words as vectors. Word2vec uses artificial neural networks to predict the relationship between words and their contexts, the semantic or functional similarity in a given context is indicated by proximity in vector space [8], [17], [18]. Word representations of words with similar meanings have similar vectors, while words with different meanings have more varied vectors [19]. Word2vec has two architectural models, namely Continuous Bag of Words (CBOW) and Skip-gram. However, in this research only one model is used in Word2vec, namely the Skip-Gram model.

Skip-gram uses the basic principle of predicting the vector of other words around a particular word. The projection layer in Skip-Gram predicts neighboring words that are close to the word in the input layer. The Skip-gram structure provides an advantage in vectorizing new words [8]. The architecture of the Word2vec Skip-Gram model can be seen in Figure 2.

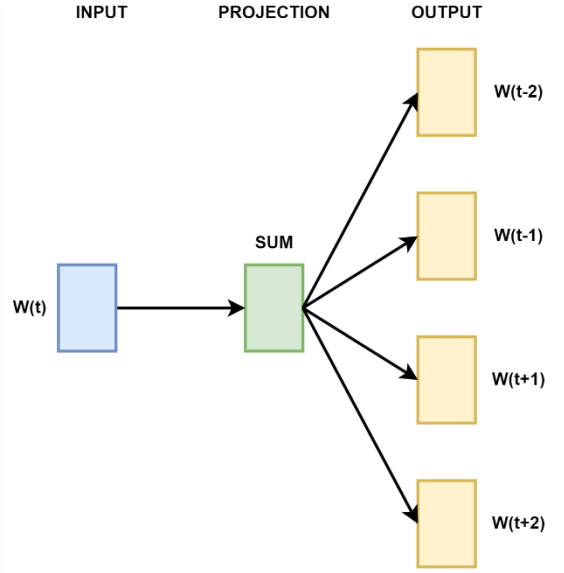

Figure 2. Architecture Skip-Gram

An example of the application of the Word2vec Skip-Gram model can be seen in Table 2.

Table 2. Word2vec+LSTM

Input	Output
bagus	mumpuni: 0.728231
	minim: 0.719939
	cocok: 0.718557
	berkesan: 0.714122
	sayangkan: 0.696667
	memuaskan: 0.693827
	menggembirakan: 0.682267
	mengagumkan: 0.680325
	impresif: 0.674129
	keren: 0.673389
jelek	menyebalkan: 0.854798
	membosankan: 0.848906
	kekanakkanakan: 0.841387
	konyol: 0.829101
	janggal: 0.825444
	malas: 0.818835
	gampang: 0.814619
	aneh: 0.810946
	muram: 0.797561
	keren: 0.792262

Table 2 gives an example of the results of applying the Word2vec word embedding Skip-Gram model to find out the relationship between the words obtained. The first column is the input column, where there are two input words as an example, namely "bagus" and "jelek". The second column contains a list of words produced by the model as output and includes the relationship value of each word to the input word. The greater the relationship value of the word in the output column indicates that the word is more closely related to the word in the input column.

GloVe: Global Vectors is a log-binary regression model on word representation that outperforms other models in terms of word analogy, word similarity, and named entity recognition. GloVe is created by drawing insights from the global statistics of the corpus, thus directly capturing the overall corpus statistics [9]. The architecture of GloVe can be seen in Figure 3, starting with a single

word representation as input. Next, in the model, the word insertion matrix serves as the weight matrix. The word insertion matrix serves as the weight matrix in the model, so the output of the model is the inner product vector of the word vectors.

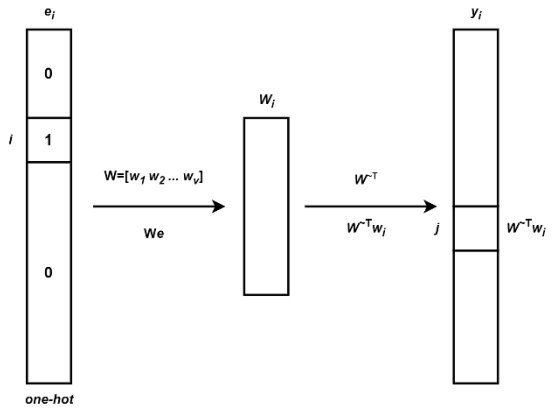

Figure 3. Architecture GloVe

Figure 3 shows the GloVe word embedding architecture to generate a word vector representation. Starting with one-hot encoding (e_i), that converts the word into a binary vector with only one element of value 1. Then the word is mapped into the embedding vector (w_i) which is a representation of the word in vector space that captures the semantic meaning of the word, through an embedding matrix. (W_e). The resulting vector is then multiplied by the transpose of the embedding matrix. $(W^{\sim T})$ and produce a dot product $(W^{\sim T}w_i)$ which is the relationship value between words and other words in the corpus. The final result is the output (y_i) a vector that shows the similarity value between the input word and other words in the corpus and gives an overview of how often the words co-occur in the same context.

FastText: FastText is a library developed by the Facebook team that can be used for word representation learning and text classification [12], [20]. FastText enriches the generated word vectors by adding substring vectors to each word, known as n-grams. Therefore, the FastText model can generate vectors for every word, even for compound words and misspelled words [21]. FastText uses softmax hierarchy to reduce computational complexity and increase speed in finding the predicted class [22].

FastText utilizes the n-gram feature to describe the word order in a sentence, thus improving the accurate representation of sentences. In the hidden layer, FastText uses averaging to combine the n-gram word representations from the input layer. At the output layer, FastText uses hierarchical softmax to predict the label of the input text, with the advantage of reducing the training time of the model [23]. The purpose of predicting the label of the input text here is to determine the category of the text whether it is positive or negative based on its content. The FastText architecture can be seen in Figure 4.

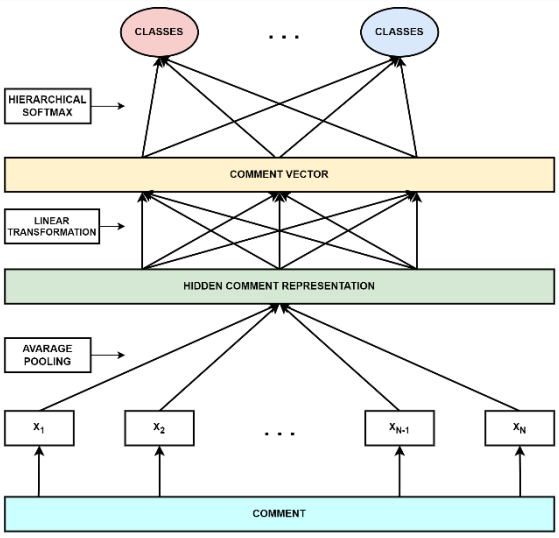

Figure 4. Architecture FastText

Figure 4 is FastText word embedding architecture to generate a vector representation of the text. It starts by breaking the input text (comment) into individual words. $(x_1, x_2, \ldots, x_N)$. Each split word is then transformed into an embedding vector, which is then combined using average pooling to obtain a single hidden comment representation that retains semantic information from the entire text. This representation then undergoes linear transformation to optimize the vector shape. The result is a comment vector ready for prediction of text classes using hierarchical softmax.

2.5. Split Data

At this stage, the data is divided into two parts with a ratio of 80% for training data and 20% for testing data from 9000 data. This division is done to ensure the model has enough data, so that the model can learn well and can recognize patterns effectively and accurately. In addition, the Cross-Validation method with k = 5 is also used with the aim of providing more accurate model performance and reducing the possibility of bias, so that the resulting model can have better performance and generalization.

2.6. Long Short-Term Memory (LSTM) Classification

LSTM is a development of Recurrent Neural Network (RNN) that can improve the ability of neural networks to store long-term information and overcome the vanishing gradient problem [24], [25]. LSTM overcomes this problem by replacing RNN nodes with LSTM cells in the hidden layer with the aim of retaining previous information. LSTM utilizes gates to organize and update information on the previous text by using three gates namely input gate, forget gate, and output gate [26]. Input gate (i_t) determines whether or not a new input can be added into the LSTM memory, forget gate (f_t) used to

remove irrelevant information from the LSTM memory, dan output gate (O_t) is used to determine which information to output [27]. The LSTM cell architecture can be seen in Figure 5.

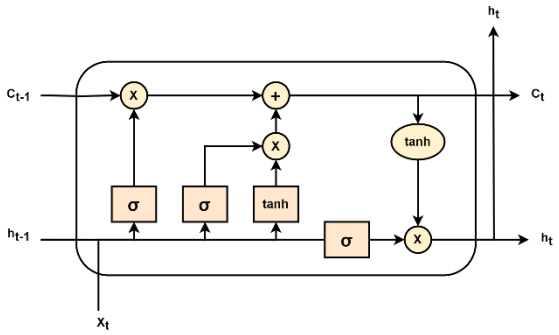

Figure 5. LSTM Cell

Forget gate uses sigmoid function, h_{t-1} and x_t are decisions, b_f is the bias value, f_t is the output at this gate with values 0 and 1. 0 is discarding the value and 1 is saving the value.

In this research, there are several layers used in the LSTM model. The first layer is the embedding layer, where this layer serves to call one of the embedding words that have been trained to be used in the LSTM model. Furthermore, this model consists of three LSTM layers with neurons of 60, 32, and 18 in order. The first and second LSTM layers are configured with "return_sequence=True" which indicates that both layers return the entire output sequence and allow the model to capture deeper sequence information. There is also a dropout layer between the LSTM layers with values of 0.3 and 0.2 respectively which aims to avoid overfitting by randomly deleting data according to a predetermined value. Finally, there is a "Dense" layer with one output unit that uses a sigmoid activation function to give a decision whether it is positive or negative. The last step is to compile all layers into a single unit by using the "RMSprop" optimizer and using the "binary_crossentropy" loss function which is ideal for classification with two classes.

2.7. Synthetic Minority Over-sampling Technique (SMOTE)

Over-sampling is a technique to balance the number of samples in the minority class with the number of samples in the majority class [28]. SMOTE is one of several over-sampling techniques. This technique increases the number of new minority class instances by using an interpolation method, where instances of adjacent minority classes are identified and used to create new minority class instances. The use of this technique can generate new synthetic samples [29]. The application of SMOTE certainly affects the number of data samples contained in the train data. Before the model was added with SMOTE, the number of data samples was 6019 and when the model was added with SMOTE, the number of data samples increased to 9068. This happens because the number of samples in the minority class increases, while the number of samples in the majority class stays the same. The distribution of samples without SMOTE and using SMOTE can be seen in Figure 6.

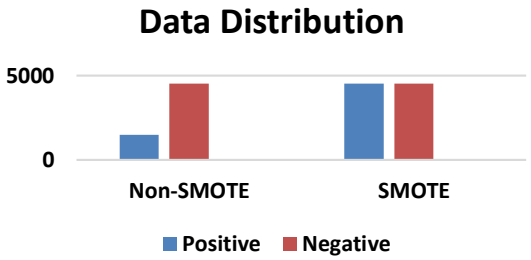

Figure 6. Comparison of Non-SMOTE and SMOTE Sample Distribution

2.8. Evaluation

The evaluation stage in this research uses the K-Fold Cross-Validation method with 5 iterations. This evaluation process assesses model performance through accuracy, precision, recall, and F-score metrics. The confusion matrix structure is represented by columns that represent predicted classes and rows that represent actual classes [30]. There are four terms contained in the confusion matrix table, namely True Positive (TP), False Negative (FN), False Positive (FP), and True Negative (TN) [31]. An explanation of the components and formulas used in the confusion matrix can be seen in Table 3.

Table 3. Confusion Matrix

Confusion Matrix		Predicted Class	
		Positive	**Negative**
Actual Class	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

- True Positive (TP): Data that is predicted positive by the system and is in fact positive.
- False Negative (FN): Data that is predicted to be negative by the system but is actually positive.
- False Positive (FP): Data that the system predicts to be positive but in reality is negative.
- True Negative (TN): Data that is predicted negative by the system and in reality the data is indeed negative.

The accuracy, precision, recall, and F-score values are obtained using the formulas as below.

Accuracy: A metric used to measure how often a method makes correct predictions, by calculating the number of correct predictions divided by the total number of predictions [31]. The formula is as follows:

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

Precision: calculation of the ratio of positive predictions and positive reality and divided by the

total number of positive predictions [25]. The formula is as follows:

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

Recall: calculation of the ratio of positive predictions and positive reality divided by the total number of positive actual data [25]. The formula is as follows:

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

F-Score: calculation by combining two metric values, namely precision and recall values [25]. The formula is as follows:

$$F - score = 2.\frac{precision.recall}{precision+recall} \quad (4)$$

3. RESULT AND DISCUSSION

This section will explain the results of the comparison of the performance of Word2vec, GloVe, and FastText word embedding classified using the LSTM model using review data from the MyTelkomsel app as much as 9000 data. The author uses two scenarios in the classification process to get the most optimal results in this study. The following are the scenarios in this study:

1) Using each word embedding method separately with the LSTM classifier model.
2) Using each word embedding method separately with an LSTM classifier model augmented with SMOTE.

The test was conducted five times for each scenario. There are also hyper parameters used in each word embedding, namely "vocab_len" and "maxLen". "vocab_len" is the number of unique words in the vocabulary that will be considered by the model. In this research, "vocab_len" is set to 55, which means there are 55 unique words that will be considered by the model. "maxLen" is the maximum length of input sequence that can be accepted by the model. In this study, "maxLen" is set to 30, which indicates that the processed words will be truncated or completed until they are exactly 30 words long.

Table 4. Word2vec+LSTM

Experiment	Accuraccy	Precision	Recall	F1-Score
Ke-1	87.25%	75.28%	72.04%	73.62%
Ke-2	88.51%	74.93%	80.37%	77.56%
Ke-3	87.31%	72.34%	78.76%	75.41%
Ke-4	87.91%	72.09%	83.33%	77.30%
Ke-5	**88.57%**	**75.90%**	**78.76%**	**77.30%**
average	87.91%	74.11%	78.65%	76.24%

The test results contained in Table 4 show the results of using Word2vec word embedding and classifying using LSTM with 5-fold. The highest value was obtained in the 5th experiment, with an accuracy value of 88.57%, precision 75.90%, recall 78.76%, and f1-score 77.30%.

Table 5. Word2vec+LSTM-SMOTE

Experiment	Accuraccy	Precision	Recall	F1-Score
Ke-1	86.52%	67.78%	86.55%	76.03%
Ke-2	85.52%	64.98%	89.78%	75.39%
Ke-3	87.11%	69.95%	83.87%	76.28%
Ke-4	**87.51%**	**70.72%**	**84.40%**	**76.96%**
Ke-5	86.13%	67.52%	84.40%	75.02%
average	86.56%	68.19%	85.80%	75.94%

The test results contained in Table 5 show the results of using Word2vec word embedding and classifying using LSTM added by SMOTE with 5-fold. The highest value was obtained in the 4th experiment, with an accuracy value of 87.51%, precision 70.72%, recall 84.40%, and f1-score 76.96%.

Table 6. GloVe+LSTM

Experiment	Accuraccy	Precision	Recall	F1-Score
Ke-1	87.05%	76.10%	69.35%	72.57%
Ke-2	87.45%	77.64%	69.08%	73.11%
Ke-3	**87.71%**	**71.89%**	**82.52%**	**76.84%**
Ke-4	86.98%	77.50%	66.66%	71.67%
Ke-5	87.45%	77.47%	69.35%	73.19%
average	87.33%	76.12%	71.39%	73.48%

The test results contained in Table 6 show the results of using GloVe word embedding and classifying using LSTM with 5-fold. The highest value was obtained in the 3rd experiment, with an accuracy value of 87.71%, precision 71.89%, recall 82.52%, and f1-score 76.84%.

Table 7. GloVe+LSTM-SMOTE

Experiment	Accuraccy	Precision	Recall	F1-Score
Ke-1	86.65%	68.15%	86.29%	76.15%
Ke-2	84.59%	63.30%	89.51%	74.16%
Ke-3	86.05%	66.59%	87.36%	75.58%
Ke-4	86.45%	67.72%	86.29%	75.88%
Ke-5	**86.78%**	**68.52%**	**86.02%**	**76.28%**
average	86.10%	66.86%	87.09%	75.61%

The test results contained in Table 7 show the results of using GloVe word embedding and classifying using LSTM added by SMOTE with 5-fold. The highest value was obtained in the 5th experiment, with an accuracy value of 86.78%, precision 68.52%, recall 86.02%, and f1-score 76.28%.

Table 8. FastText+LSTM

Experiment	Accuraccy	Precision	Recall	F1-Score
Ke-1	**89.11%**	**76.26%**	**81.18%**	**78.64%**
Ke-2	87.98%	78.50%	70.69%	74.39%
Ke-3	88.97%	75.75%	81.45%	78.49%
Ke-4	88.91%	75.94%	80.64%	78.22%
Ke-5	86.52%	67.35%	88.17%	76.36%
average	88.30%	74.76%	80.43%	77.22%

The test results contained in Table 8 show the results of using FastText word embedding and classifying using LSTM with 5-fold. The highest value was obtained in the 1st experiment, with an accuracy of 89.11%, precision 76.26%, recall 81.18%, and f1-score 78.64%.

\

Table 9. FastText+LSTM-SMOTE

Experiment	Accuraccy	Precision	Recall	F1-Score
Ke-1	85.85%	66.12%	87.63%	75.37%
Ke-2	86.12%	66.46%	88.44%	75.89%
Ke-3	85.59%	63.25%	89.78%	74.22%
Ke-4	**87.38%**	**70.31%**	**84.67%**	**76.82%**
Ke-5	83.93%	61.77%	91.66%	73.80%
average	85.77%	65.58%	88.44%	75.22%

The test results contained in Table 9 show the results of using FastText word embedding and classifying using LSTM added by SMOTE with 5-fold. The highest value was obtained in the 4th experiment, with an accuracy of 87.38%, precision 70.31%, recall 84.67%, and f1-score 76.82%.

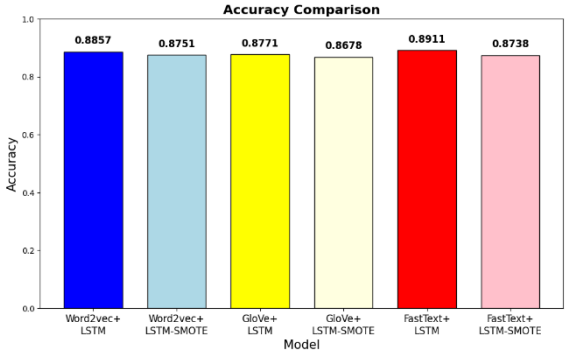

Figure 6. Model Combination Accuracy

Figure 6 shows the best accuracy comparison of each combination used in this study. In testing the first scenario, namely the combination of word embedding and LSTM without adding SMOTE, the best combination is FastText + LSTM with an accuracy value of 89.11%. Followed by the Word2vec + LSTM combination with an accuracy value of 88.57% and finally there is a combination with the smallest accuracy value, namely GloVe + LSTM with an accuracy value of 87.71%. Where the accuracy value of the three word embedding in the first scenario provides quite good accuracy results.

In testing the second scenario, which is the combination of word embedding and LSTM added to SMOTE, the best combination is Word2vec + LSTM-SMOTE with an accuracy value of 87.51%. Then followed by the FastText + LSTM-SMOTE combination with an accuracy value of 87.38% and finally the combination with the smallest accuracy value is GloVe + LSTM-SMOTE with an accuracy value of 86.78%. Overall, the use of SMOTE in this study did not increase the accuracy of the model combination and resulted in a decrease in the accuracy value of each model combination tested.

4. DISCUSSION

From the research results, the combination of FastText and LSTM provides the highest accuracy value with a value of 89.11%. This is followed by the combination of Word2vec and LSTM with an accuracy value of 88.57%. Then there is a combination of GloVe and LSTM with an accuracy value of 87.71%, then followed by 2 combinations with almost the same accuracy value, namely Word2vec and LSTM-SMOTE with an accuracy value of 87.51% and FastText and LSTM-SMOTE with an accuracy value of 87.38%. Finally, the combination with the lowest accuracy value in this study is the combination of GloVe and LSTM-SMOTE with an accuracy value of 86.78%. Of the two scenarios used in this study, it was found that the first scenario has a superior accuracy value compared to the second scenario. Where the results of the accuracy value of the first scenario are in the 1st, 2nd, and 3rd order in this study. It is also proven that the combination of FastText and LSTM is the most superior combination compared to the other five combinations.

Compared to previous studies that use the same word embedding but different classification models, this study provides quite good results and is similar to previous studies. In research [8] with classification using CNN, the accuracy value of the combination of each model is 92.5% for Word2vec, 95.8% for GloVe, and 97.2% for FastText. Research [32] with SVM classification obtained the accuracy value of the combination of each model, 65% for Word2vec, 85% for GloVe, and 71% for FastText. These results prove that the selection of word embedding and calcification models can affect the combination performance for each model. In this study, it was found that FastText + LSTM is the best combination compared to Word2vec + LSTM and GloVe + LSTM. This shows that FastText word embedding can capture more detailed sub-word information for text classification and help improve the performance of LSTM. Although SMOTE helps to make the model fairer to minority classes, the decrease in accuracy in this study shows that this method should be considered carefully to avoid overfitting the minority classes and ensure more accurate results.

5. CONCLUSION

This research focuses on the comparison of Word2vec, GloVe, and FastText word embedding on classifiers using LSTM. The experiment was conducted using two scenarios: LSTM-only classification and LSTM augmented with SMOTE to handle data imbalance. The aim is to find the most optimal combination of word embedding and classification strategies in sentiment analysis on MyTelkomsel app reviews.

Based on the experimental results on three word embedding with two existing scenarios, the combination of FastText and LSTM has the highest accuracy value with an accuracy value of more than 89%, followed by the combination of Word2vec and LSTM with an accuracy value of more than 88%, then the combination of GloVe and LSTM with an accuracy value of more than 87%. then the combination of Word2vec and LSTM-SMOTE with an accuracy value of more than 87% and the

combination of FastText and LSTM-SMOTE with an accuracy value of more than 87%. Then finally the combination of GloVe and LSTM-SMOTE with an accuracy value of more than 86%. From this study, the use of the SMOTE method has an unfavorable impact on the combination of methods used by reducing the accuracy value of the model. Although the difference in accuracy values obtained from this test is not too far, it shows that the application of each word embedding used in LSTM and LSTM-SMOTE has competitive performance. Suggestions for future research are to increase the number of datasets that are different from this research to validate whether the combination of FastText with LSTM really provides the highest accuracy value compared to other combinations and do a deeper exploration of the effect of SMOTE on LSTM.

REFERENCES

[1] W. H. Ali and M. Ariyanti, "Hyper-Segmentation Lapser MyTelkomsel Apps Using K-Means Clustering to Increase Data Package Purchases in Area 3-East Java, Central Java-DIY, Bali Nusa Tenggara," Bandung, Jun. 2023.

[2] A. Ibrahim, F. S. Elisa, J. Fernando, L. Salsabila, N. Anggraini, and S. N. Arafah, "Pengaruh E-Service Quality Terhadap Loyalitas Pengguna Aplikasi MyTelkomsel," Building of Informatics, Technology and Science (BITS), vol. 3, no. 3, pp. 302–311, Dec. 2021, doi: 10.47065/bits.v3i3.1076.

[3] S. Pandya and P. Mehta, "A Review On Sentiment Analysis Methodologies, Practices And Applications," International Journal Of Scientific & Technology Research, vol. 9, p. 2, Feb. 2020, [Online].

[4] G. Dharani Devi and Dr. S .Kamalakkannan, " Literature Review on Sentiment Analysis in Social Media: Open Challenges toward Applications", International Journal of Advanced Science and Technology, Vol. 29, No. 7, pp. 1462-1471, (2020)

[5] Jain, P.K. and Pamula, R., "A systematic literature review on machine learning applications for consumer sentiment analysis using online reviews", Computer Science Review, Vol. 41, (2021).

[6] R. Indra Kurnia and A. Suganda Girsang, "Classification of User Comment Using Word2vec and Deep Learning," Mar. 2021, doi: 10.25046/aj060264.

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," Jan. 2013, [Online]. Available: http://arxiv.org/abs/1301.3781

[8] B. Jang, I. Kim, and J. W. Kim, "Word2vec convolutional neural networks for classification of news articles and tweets," PLoS One, vol. 14, no. 8, Aug. 2019, doi: 10.1371/journal.pone.0220976.

[9] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation," Doha, Qatar, Oct. 2014. [Online]. Available: http://nlp.

[10] Mishev, K., Gjorgjevikj, A., Vodenska, I., Chitkushev, L. T., & Trajanov, D. (2020). Evaluation of sentiment analysis in finance: from lexicons to transformers. IEEE access, 8, 131662-131682.

[11] E. M. Dharma, F. Lumban Gaol, H. Leslie, H. S. Warnars, and B. Soewito, "The Accuracy Comparison Among Word2vec, GloVe, AND FastText Towards Convolution Neural Network (CNN) Text Classification," J Theor Appl Inf Technol, vol. 31, no. 2, 2022, [Online]. Available: www.jatit.org.

[12] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," vol. 5, pp. 135–146, 2017, doi: 10.1162/tacl_a_00051/1567442/tacl_a_00051.pdf.

[13] N. Alvi Hasanah, Nanik Suciati, and Diana Purwitasari, "Pemantauan Perhatian Publik terhadap Pandemi COVID-19 melalui Klasifikasi Teks dengan Deep Learning," Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), vol. 5, no. 1, pp. 193–202, Feb. 2021, doi: 10.29207/resti.v5i1.2927.

[14] Johnson, R., & Zhang, T. (2015). Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics.

[15] Ahmad, S., Ridwan, A. M., & Setyawan, G. D. (2023). Analisis Sentimen Product Tools & Home Menggunakan Metode Cnn Dan Lstm. Teknokom, 6(2), 133-140.

[16] Tala, F. Z., "A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia". M.Sc. Thesis. Master of Logic Project. Institute for Logic, Language and Computation. Universiteit van Amsterdam, The Netherlands, 2003.

[17] S. Al-Saqqa and A. Awajan, "The Use of Word2vec Model in Sentiment Analysis: A Survey," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Dec. 2019, pp. 39–43. doi: 10.1145/3388218.3388229.

[18] S. Li and B. Gong, "Word embedding and text classification based on deep learning

methods," *MATEC Web of Conferences*, vol. 336, p. 06022, 2021, doi: 10.1051/matecconf/202133606022.

[19] S. Sivakumar, L. S. Videla, T. R. Kumar, J. Nagaraj, S. Itnal, and D. Haritha, *Review on Word2Vec Word Embedding Neural Net*. India: 2020 International Conference on Smart Electronics and Communication (ICOSEC), 2020. doi: 10.1109/ICOSEC49089.2020.9215319.

[20] A. Amalia, O. S. Sitompul, E. B. Nababan, and T. Mantoro, An Efficient Text Classification Using fastText for Bahasa Indonesia Documents Classification. Medan: 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA), 2020. doi: 10.1109/DATABIA50434.2020.9190447.

[21] J. C. Young and A. Rusli, "Review and Visualization of Facebook's FastText Pretrained Word Vector Model," pp. 1–6, Aug. 2019, doi: 10.1109/ICESI.2019.8863015.

[22] A. Alessa, M. Faezipour, and Z. Alhassan, "Text classification of flu-related tweets using FastText with sentiment and keyword features," in Proceedings - 2018 IEEE International Conference on Healthcare Informatics, ICHI 2018, Institute of Electrical and Electronics Engineers Inc., Jul. 2018, pp. 366–367. doi: 10.1109/ICHI.2018.00058.

[23] T. Yao, Z. Zhai, and B. Gao, Text Classification Model Based on fastText. Dalian: Proceedings of 2020 IEEE International Conference on Artificial Intelligence and Information Systems: ICAIIS, 2020. doi: 10.1109/ICAIIS49377.2020.9194939.

[24] M. A. Riza and N. Charibaldi, "Emotion Detection in Twitter Social Media Using Long Short-Term Memory (LSTM) and Fast Text," International Journal of Artificial Intelligence & Robotics (IJAIR), vol. 3, no. 1, pp. 15–26, May 2021, doi: 10.25139/ijair.v3i1.3827.

[25] N. K. Sirohi, "Categorization of Text using Long Short-Term Memory with Glove," 2023, doi: 10.21203/rs.3.rs-3239199/v1.

[26] M. A. Nurrohmat and A. SN, "Sentiment Analysis of Novel Review Using Long Short-Term Memory Method," IJCCS (Indonesian Journal of Computing and Cybernetics Systems), vol. 13, no. 3, p. 209, Jul. 2019, doi: 10.22146/ijccs.41236.

[27] S. Siami-Namini, N. Tavakoli, and A. S. Namin, *The performance of LSTM and BiLSTM in forecasting time series*. Los Angeles, CA: 2019 IEEE International Conference on Big Data (Big Data), 2019. doi: 10.1109/BigData47090.2019.9005997.

[28] V. Rupapara, F. Rustam, H. F. Shahzad, A. Mehmood, I. Ashraf, and G. S. Choi, "Impact of SMOTE on Imbalanced Text Features for Toxic Comments Classification Using RVVC Model," IEEE Access, vol. 9, pp. 78621–78634, 2021, doi: 10.1109/ACCESS.2021.3083638.

[29] P. Jeatrakul, K. Wai Wong, and C. Che Fung, "Classification of Imbalanced Data by Combining the Complementary Neural Network and SMOTE Algorithm," 2010.

[30] V. M. Patro and M. Ranjan Patra, "Augmenting Weighted Average with Confusion Matrix to Enhance Classification Accuracy," Transactions on Machine Learning and Artificial Intelligence, vol. 2, no. 4, Aug. 2014, doi: 10.14738/tmlai.24.328.

[31] A. Giachanou and F. Crestani, "Like it or not: A survey of Twitter sentiment analysis methods," ACM Computing Surveys, vol. 49, no. 2. Association for Computing Machinery, Jun. 01, 2016. doi: 10.1145/2938640.

[32] A. Margaretha and N. Helena, " COMPARISON PERFORMANCE OF WORD2VEC, GLOVE, FASTTEXT USING SUPPORT VECTOR MACHINE METHOD FOR SENTIMENT ANALYSIS", Jurnal Teknik Informatika (JUTIF), Vol. 5, No. 3, pp. 669-674, June 2024, doi: 10.52436/1.jutif.2024.5.3.1366.