JURNAL RISET INFORMATIKA **P-ISSN: 2656-1743 |E-ISSN: 2656-1735**
Vol. 5, No. 2. March 2023 DOI: https://doi.org/10.34288/jri.v5i2.507
Accredited rank 3 (SINTA 3), excerpts from the decision of the Minister of RISTEK-BRIN No. 200/M/KPT/2020

COMPARATIVE ANALYSIS OF USING WORD EMBEDDING IN DEEP LEARNING FOR TEXT CLASSIFICATION

Mukhamad Rizal Ilham[1], Arif Dwi Laksito[2*)]

Fakultas Ilmu Komputer
Universitas Amikom Yogyakarta
Yogyakarta, Indonesia
mukhamad.ilham@students.amikom.ac.id[1], arif.laksito@amikom.ac.id[2*)]

(*) Corresponding Author

Abstract

A group of theory-driven computing techniques known as natural language processing (NLP) are used to interpret and represent human discourse automatically. From part-of-speech (POS) parsing and tagging to machine translation and dialogue systems, NLP enables computers to carry out various natural language-related activities at all levels. In this research, we compared word embedding techniques FastText and GloVe, which are used for text representation. This study aims to evaluate and compare the effectiveness of word embedding in text classification using LSTM (Long Short-Term Memory). The research stages start with dataset collection, pre-processing, word embedding, split data, and the last is deep learning techniques. According to the experiments' results, it seems that FastText is superior compared to the glove technique. The accuracy obtained reaches 90%. The number of epochs did not significantly improve the accuracy of the LSTM model with GloVe and FastText. It can be concluded that the FastText word embedding technique is superior to the GloVe technique.

Keywords: Word Embedding; Sentiment Analysis; Deep Learning; LSTM

Abstrak

Natural Language Processing (NLP) adalah seperangkat teknik komputasi yang didorong oleh teori untuk secara otomatis menganalisis dan mewakili bahasa manusia. NLP memungkinkan komputer untuk melakukan berbagai tugas terkait bahasa alami di semua tingkatan, mulai dari penguraian dan penandaan part-of-speech (POS) hingga Machine translation dan sistem dialog. Dengan banyaknya data dan peningkatan jumlah dokumen yang signifikan per hari, klasifikasi teks menjadi semakin penting karena digunakan dalam berbagai aplikasi seperti penyaringan informasi, penyaringan spam, hingga mengkategorikan text. Tujuan dari penelitian ini untuk menganalisis perbandingan performa kinerja word embedding Glove dan Fasttext pada klasifikasi text. Dalam penelitian ini juga menggunakan model deep learning algoritma LSTM (Long Short-Term Memory). Berdasarkan hasil eksperiman metodologi Fasttext lebih unggul dibanding dengan teknik Glove akurasi yang didapatkan mencapai 90% dengan menggunakan pelatihan di semua epoch dan perbandingan akurasi masing masing epoch tidak kelihatan signifikan. Dapat disimpulkan bahwa Teknik word embedding Fasttext lebih unggul dibanding dengan teknik GloVe.

Kata kunci: Word Embedding; Sentiment Analisis; Deep Learning; LSTM

INTRODUCTION

A group of theory-driven computing techniques known as natural language processing (NLP) are used to interpret and represent human discourse automatically. NLP research has evolved from Punch cards and Batch processing, where decoding a single sentence took up to seven minutes, to an era like Google, where millions of web pages can be processed in less than a second (Young, Hazarika, Poria, & Cambria, 2018). From part-of-speech (POS) parsing and tagging to machine translation and dialogue systems, NLP enables computers to carry out various natural language-related activities at all levels.

Text categorisation, which is used in various applications, including information filtering, spam filtering, and text categorisation, is becoming more and more crucial due to the vast quantity of data and considerable growth in the number of documents produced daily. The main research topics include efficient document text representation and the selection of better deep learning algorithms. The technique of automatically comprehending, gathering, and analysing textual data to extract sentiment information from views expressed in text is known as sentiment analysis or opinion mining (Wang, Nulty, & Lillis, 2020).

One technique to convert words into continuous vectors of a certain length is word embedding. Word embedding converts words into vectors that summarise their syntactic and semantic information. Therefore, word embedding is considered suitable as a feature representation in neural network models for Natural Language Processing (NLP) tasks (Deho, Agangiba, Aryeh, & Ansah, 2018). Word weighting is a pre-processing data strategy that assigns an appropriate weight to each term to represent the term's relevance to the text. This model plays a vital role in improving text classification with high efficiency. Word embedding is a crucial technique in deep learning since it can analyse the text as an input for the deep learning model.

Deep learning is a technique for feature extraction, pattern recognition, and classification that involves employing several layers of processing to build different models and execute classification tasks from the gathered data (Imaduddin, Widyawan, & Fauziati, 2019). The deep learning algorithm used in this research is LSTM (Long Short-Term Memory), one of the variations of RNN (Recurrent Neural Network). LSTM can be used to overcome the weakness of RNN, which is its inability to store data during learning if too much data has to be stored.

The bag of words technique, the first technique created for encoding words into vector form, marked the beginning of the development of word embedding. In 1972 Karen Spärck Jones introduced the TF-IDF (Term Frequency - Inverse Document Frequency) technique, a combination of TF (Term frequency) and IDF (inverse document frequency) is a statistical measure that describes the words in several documents (Jones, 1972). To build practical neural network-based word insertion training in 2013, Tomas Mikolov and his colleagues at Google created the new word2vec approach (Mikolov, Chen, Corrado, & Dean, 2013). After a year, Jeffrey Penington and his colleagues created the GloVe (Global Vectors) technique, an extension of the effective word2vec learning technology (Brennan, Loan, Watson, Bhatt, & Bodkin, 2017). The last technique, FastText, was developed by Facebook in 2017, which is very fast and effective in learning word representation and text classification (Bojanowski, Grave, Joulin, & Mikolov, 2017).

There have been numerous studies in the area of sentiment analysis that used word embedding techniques like Bag of Word (Imaduddin et al., 2019; Marukatat, 2020), Word2Vec (AlSurayyi, Alghamdi, & Abraham, 2019; Imaduddin et al., 2019; Kilimci & Akyokus, 2019; Marukatat, 2020; Rahman, Sari, & Yudistira, 2021), doc2vec (Imaduddin et al., 2019), GloVe (AlSurayyi et al., 2019; Imaduddin et al., 2019; Kilimci & Akyokus, 2019), and FastText (Kilimci & Akyokus, 2019; Marukatat, 2020). Those word embedding approaches were then evaluated using RNN (Kilimci & Akyokus, 2019; Rahman et al., 2021), CNN(Kilimci & Akyokus, 2019), LSTM (Kilimci & Akyokus, 2019; Rahman et al., 2021), Naïve Bayes (Rahman et al., 2021). A study by (Deho et al., 2018) offered word embedding to identify the polarity of sentiment (positive, negative, or neutral) from existing text. This improved the accuracy of sentiment categorisation. Additionally, a new technique, known as Improved Word Vector (IWV), was presented by (Bojanowski et al., 2017) to increase the precision of pre-trained word embedding in sentiment analysis. The study's findings indicate that the word embedding technique can increase the precision of text classification (Deho et al., 2018). The IWV approach significantly improves the researcher's proposed sentiment analysis technique (Rezaeinia, Ghodsi, & Rahmani, 2017).

The performance of word embedding word2vec Continuous Bag of Words (CBOW), word2vec, doc2vec, GloVe, and FasText was compared by other researchers in addition to new approaches being suggested and word embedding techniques being identified. Accuracy of 95.52% on the domain of hotel reviews from the Traveloka site with a total of 5,000 reviews. The GloVe method has the highest accuracy rate compared to other methods (Imaduddin et al., 2019). This research is similar to previous research (Kamiş & Goularas, 2019) that GloVe can improve almost all configuration performance.

The effectiveness of Deep Learning has been compared in another research. Previous research by (AlSurayyi et al., 2019) compared RNN combined with LSTM, RNN combined with Bi-LSTM (Bidirectional LSTM), and CNN (Convolutional neural networks) for word representation using word2vec and GloVe techniques. The results showed that RNN combined with Bi-LSTM using the glove technique got better accuracy than other methods. This study used the domain of restaurant reviews from Yelp. Researchers (Rahman et al., 2021) compared LSTM, Naïve Bayes, RNN, and word representation using the Word2vec technique. The outcomes demonstrated that LSTM was better than other approaches.

This study uses GloVe and FastText word embedding in the LSTM model for text representation. This research aims to analyse and compare the effectiveness of word embedding on text represented by LSTM deep learning architecture.

RESEARCH METHODS

This research is expected to provide high-accuracy text classification and good performance between word embedding gloves and FastText techniques evaluated using LSTM. To achieve the expected research objectives, we used the methodology shown in Figure 1.

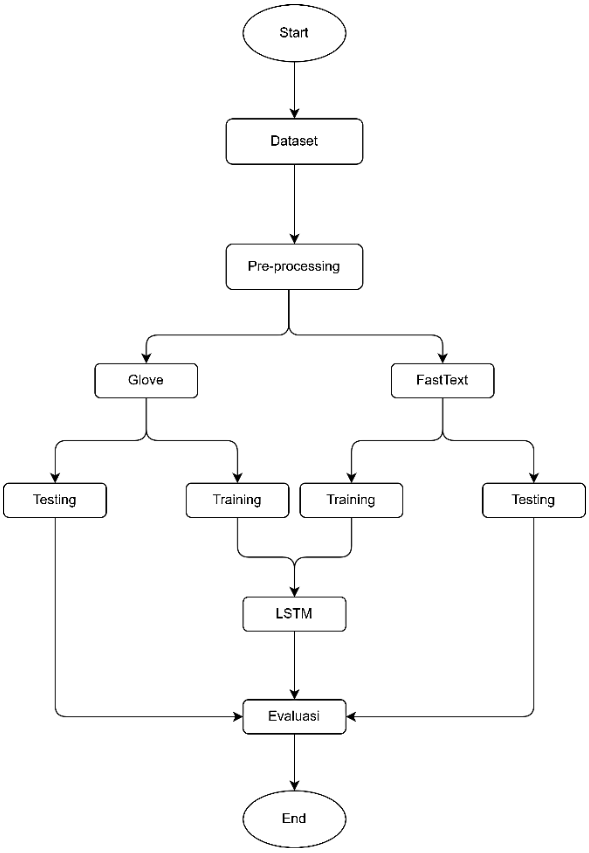


Figure 1. Flowchart

Data collection and labelling

The Spotify application review dataset from the Kaggle.com website is used in this study. The datasets of our study can be downloaded from the URL: https://www.kaggle.com/datasets/mfaaris/spotify-app-reviews-2022.

Spotify app reviews on Google Play Store were collected from January 1, 2022, to July 9, 2022. The dataset consists of 5 columns, namely time_submitted, review, rating, total_thumbsup, replay, and 61,594 rows. However, the columns used for this study are the review column and the rating column.

Pre-processing

Before the data is used for sentiment analysis, several preparatory processes must be done to get the best classification results. In the first stage, symbols, punctuation marks, and emojis are removed from the data set. The second stage, tokenisation and case folding is breaking down the sentences in the dataset into words, also known as tokens, and converting all capital letters into lowercase letters. The third step is filtering or removing stop words, taking important words and discarding words that are unimportant or have no meaning.

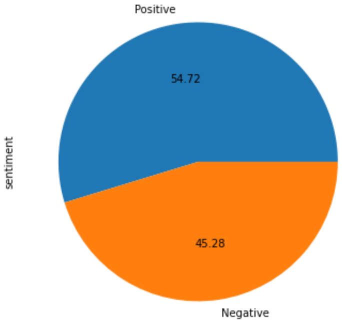


Figure 2. Percentage of labelling data

The library used for filtering is NLTK (Natural Language Toolkit), developed by Steven Bird and Edward Loper at the University of Pennsylvania in 2001 (Botrè, Lucarini, Memoli, & D’Ascenzo, 1981). The fourth step is Stemming, converting unstandardised words into common words or removing affixes. The last step is labelling data. The data is grouped into positive and negative sentiments based on application ratings, as in the research (AlSurayyi et al., 2019; Imaduddin et al., 2019), which only uses positive sentiments and negative sentiments in data labelling. This study's total percentage of labelling data is shown in Figure 2.

Word Embedding

A. Glove

Word embedding converts words into a continuous vector form with a predefined text length. Many methods have been developed to convert words into vectors, including a bag of words, TF-IDF (Jones, 1972), Word2vec (Mikolov et al., 2013), GloVe (Brennan et al., 2017), and FastText (Bojanowski et al., 2017). GloVe is a method that combines local context-based learning in word2vec with global statistical matrix factorisation techniques like LSA (Brennan et al., 2017). Matrix factorisation and the Skip-Gram method are combined in the GloVe methodology. The co-occurrence matrix created by GloVe (word context X) is used for prediction and calculation outside the existing corpus (Imaduddin et al., 2019).

$$J_0 \sum_{i,j=1}^{v} f\left(X_{ij}\right)\left(w_i^T \widetilde{w_J} + b_i + \widetilde{b_J} - logX_{ij}\right)^2 \text{ ....(1)}$$

Where each element X_{ij} Indicates the number of times the word appears in word J, for w_j is a vector for the context word w_i vector of the main word and b_i, b_j Scalar bias for the main word and context (Kilimci & Akyokus, 2019).

B. FastText

FastText, an open-source project from Facebook Research is a fast and efficient technique for learning word representations and performing text classification often used for NLP. The primary function of FastText insertion is to analyse the internal structure of words. This works particularly well in morphologically complex languages as it allows learning of self-representations for various word morphologies (Bojanowski et al., 2017).

$$s(w,c) = \sum_{g\epsilon g\omega} Z_g^T v_c \text{ .... (2)}$$

The Word2Vec-proposed negative sampling skip-gram model is implemented by FastText using a modified skip-gram function. The word score is calculated by adding the vector representation of the n-grams in the set $G_w \subset \{1, \dots, G\}$, which is the set of n-grams found in the word w.

Split Dataset

The data set is divided into training and testing data to train the machine learning model. In this experiment, the data is divided into a ratio of 80:20, with 80% of the data used to train the model and 20% used to test it.

LSTM

Long Short-Term Memory (LSTM) networks are a complex deep learning approach. LSTM works very well on various problems and is widely used by many researchers (AlSurayyi et al., 2019). Due to its complex dynamics, LSTM may quickly "memorise" data over a lengthy period. In a vector of memory cells called $c_t^l \in R^n$, the "long-term" memory is stored. Although different LSTM designs vary in terms of connection layout and activation function. All LSTM architectures have explicit memory cells that can store data for long periods of time. The LSTM has the option of replacing, retrieving, or storing the memory cell for later (Zaremba, Sutskever, & Vinyals, 2014). There are three gates for storing information for an extended period. The forget gate removes information from the cell that is not needed. The input gate adds beneficial information. The output gate pulls valuable information from the cell state for output values. Through gates that let information flow through or are blocked by the LSTM unit, the LSTM unit decides what to store and when to permit reads, writes, and deletions(Kilimci & Akyokus, 2019)

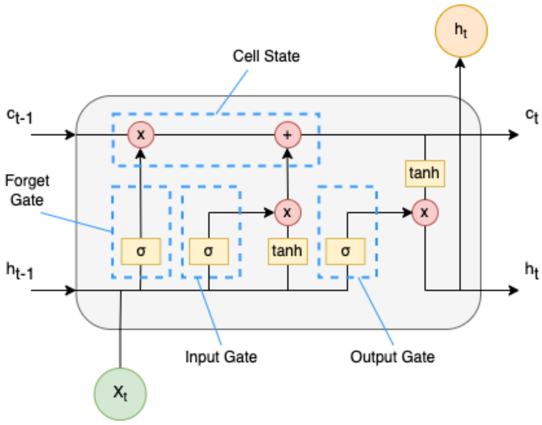


Figure 3. LSTM

A collection of LSTM architectures or memory cells are shown in Figure 3.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1}b_i) \text{ .... (3)}$$

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right) \text{ .... (4)}$$

$$c_t = f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c \text{ .... (5)}$$

$$\mathrm{o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o)} \text{ .... (6)}$$

$$h_t = o_t \, tanh(c_t) \text{ .... (7)}$$

where
i_t = input gate.
f_t = forget gate.
o_t = output gate.
c_t = sell activation vector.
X_t = the input at time t.
h_{t-1}= the previous state.
C_{t-1}= the previous state memory.
C_t= current memory state.
h_t= the current state.

P-ISSN: 2656-1743 |E-ISSN: 2656-1735
DOI: https://doi.org/10.34288/jri.v5i2.507
Accredited rank 3 (SINTA 3), excerpts from the decision of the Minister of RISTEK-BRIN No. 200/M/KPT/2020

Evaluation

The evaluation stage is a step to check the accuracy of the experimental results and measure the performance of the model that has been produced. The performance of the algorithm is measured in this study using a confusion matrix, and the metrics utilised for assessment are True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

$$Accuracy = \frac{TP+TN}{TP+TN+FP} \dots\dots\dots (8)$$

$$Precission = \frac{TP}{TP+FP} \dots\dots\dots (9)$$

$$Recall = \frac{TP}{TP+FN} \dots\dots\dots (10)$$

$$F - Measure = 2 * Recall * \frac{Precision}{Recall+Precision} \dots\dots (11)$$

RESULTS AND DISCUSSION

In this section, we analyse and compare how well word embedding represented by LSTM architecture performs. This research was conducted using phyton version 3.10.7 and jupyter notebook version 6.4.11 and a device with an intel core i3-8100 processor, 8 GB RAM and windows ten pro operating system. The dataset used in this study consists of 61,594 Spotify application reviews from the Google Play Store that have gone through pre-processing and labelling. As shown in Figure 2, the dataset is divided into positive and negative sentiments, with negative sentiments based on reviews with ratings 1-2 and positive sentiments based on reviews with ratings 4-5.

After pre-processing phase, word embedding techniques were employed to convert words into vector form with a predetermined length. In this step, we compared GloVe and FastText in the English language with 300 dimensions. It took about 2 minutes 30 seconds to fetch 4.7 GB of GloVe data, while it took 5 minutes 50 seconds to load 4.2 GB of FastText data.

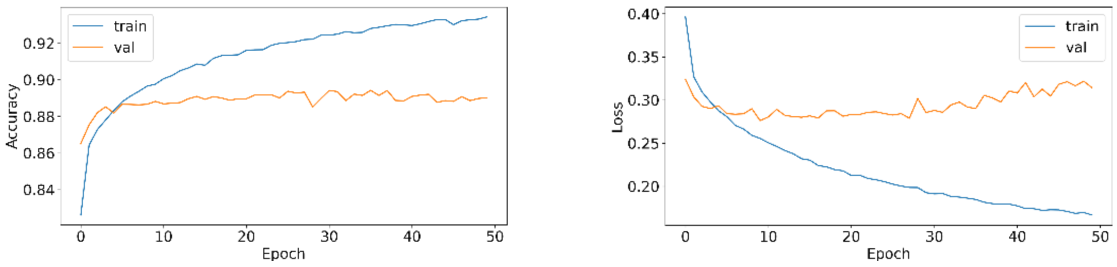
Table 1. Classification accuracy of the Word embedding GloVe deep learning model

	GloVe	
	accuracy	time
epoch 50	89%	33 minute 11 second
epoch 100	89%	1 hour 7 minute 45 second
epoch 200	89%	2 hour 16 minute 6 second

Table 2. Classification accuracy of the word embedding FastText deep learning model

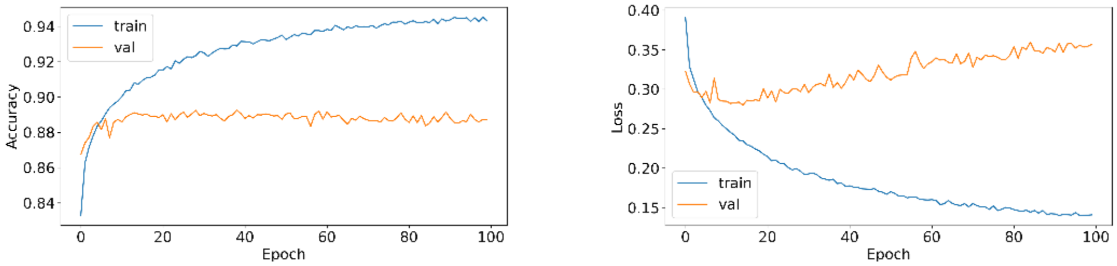
	FastText	
	accuracy	time
epoch 50	90%	34 minute 15 second
epoch 100	90%	1 hour 7 minute 22 second
epoch 200	90%	2 hour 14 minute 56 second

Further, the dataset is split into training and testing data in a ratio of 80:20, with 80% of the data used to train the model and 20% to test it. To make the data more balanced, random oversampling (ROS) was used to double the minority class and add it to the training dataset before starting the data split. The positive and negative opinions were 29,937 and 24,771 before the ROS process. Consequently, the number of labels in the minority (Negative) was set at 29,937, which was also the number in the majority (Positive). In this study, we used a single LSTM architecture or one layer, 64 units, Adam optimiser, and the learning rate is 0.001. The training was performed in three iterations of 50, 100, and 200 epochs. Figures 4 to 6 illustrate the LSTM in three epochs using the GloVe word embedding approach. Training and validation have a wide gap in accuracy and loss.



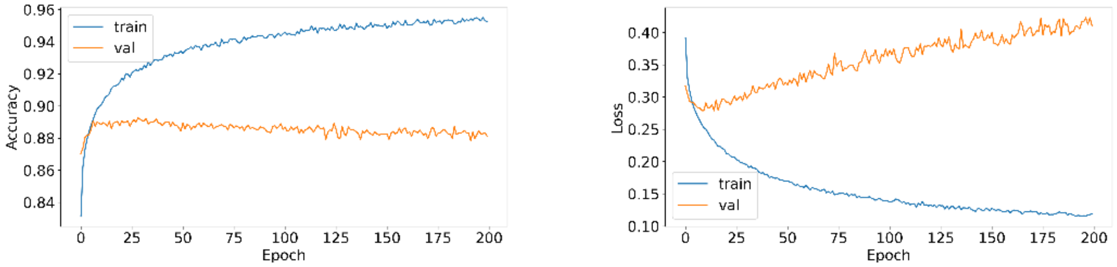
(a) (b)

Figure 4. Training (a) Accuracy and (b) Loss using LSTM and GloVe at 50 epochs



(a) (b)

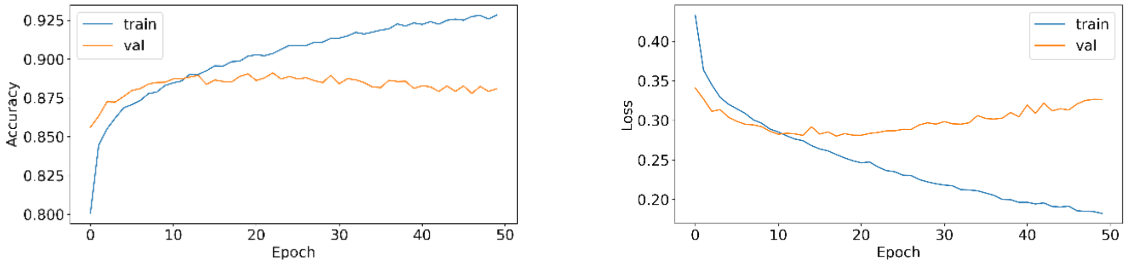
Figure 5. Training (a) Accuracy and (b) Loss using LSTM and GloVe at 100 epochs



(a) (b)

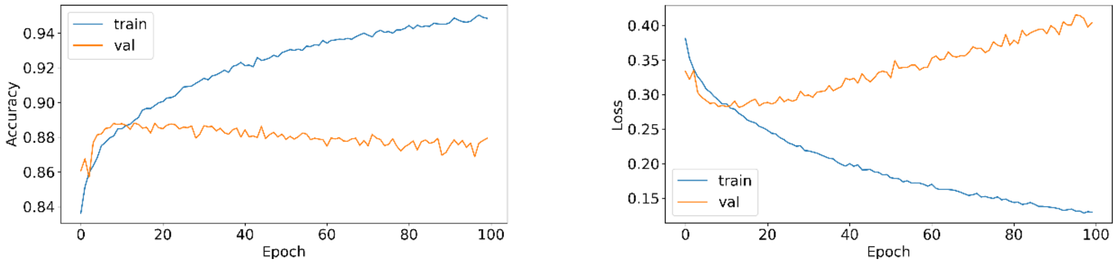
Figure 6. Training (a) Accuracy and (b) Loss using LSTM and GloVe at 200 epochs

Figures 7 to 9 depict the accuracy and loss during LSTM training using FastText word embedding with epochs of 50, 100, and 200. The FastText word embedding technique is superior to all epochs compared to the Glove technique, as seen in table 2. However, the training and validation have a wide gap in accuracy and loss, similar to GloVe word embedding.



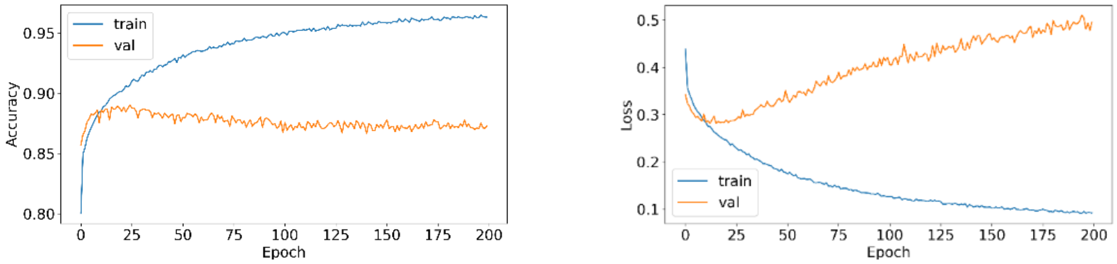
(a) (b)

Figure 7. Training (a) Accuracy and (b) Loss using LSTM and FastText at 50 epochs



(a) (b)

Figure 8. Training (a) Accuracy and (b) Loss using LSTM and FastText at 100 epochs



(a) (b)

Figure 9. Training (a) Accuracy and (b) Loss using LSTM and GloVe at 200 epochs

CONCLUSIONS AND SUGGESTIONS

Bag of Word, TF-IDF, word2vec, GloVe and FastText are some word embedding methods to display words in vector form. In this study, we compare GloVe and FastText word embedding, two state-of-the-art word representation algorithms that use deep learning LSTM architecture. According to the experiments' results, it seems that FastText is superior compared to the glove technique. The accuracy obtained reaches 90%. The number of epochs did not significantly improve the accuracy of the LSTM model with GloVe and FastText. For all the scenarios tested, the training and validation have a wide gap in the model's accuracy and loss. It seems that model improvement is needed for future research. Moreover, the early stop method for model training is crucial for overfitting and underfitting. The early stops technique can also achieve model convergence in the precise number of epochs.

REFERENCES

AlSurayyi, W. I., Alghamdi, N. S., & Abraham, A. (2019). Deep learning with word embedding modeling for a sentiment analysis of online reviews. *International Journal of Computer Information Systems and Industrial Management Applications*, *11*, 227–241. Retrieved from http://www.mirlabs.org/ijcisim/regular_papers_2019/IJCISIM_22.pdf

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Transactions of the Association for Computational Linguistics. *Transactions of the Association for Computational Linguistics*, *5*, 135–146. Retrieved from https://transacl.org/ojs/index.php/tacl/article/view/999

Botrè, C., Lucarini, C., Memoli, A., & D'Ascenzo, E. (1981). 397 - On the entropy production in oscillating chemical systems. *Bioelectrochemistry and Bioenergetics*, *8*(2), 201–212. https://doi.org/10.1016/0302-4598(81)80041-4

Brennan, P. M., Loan, J. J. M., Watson, N., Bhatt, P. M., & Bodkin, P. A. (2017). Pre-operative obesity does not predict poorer symptom control and quality of life after lumbar disc surgery. *British Journal of Neurosurgery*, *31*(6), 682–687. https://doi.org/10.1080/02688697.2017.1354122

Deho, O. B., Agangiba, W. A., Aryeh, F. L., & Ansah, J. A. (2018). Sentiment analysis with word embedding. *2018 IEEE 7th International Conference on Adaptive Science & Technology (ICAST)*, 1–4. https://doi.org/10.1109/ICASTECH.2018.8506717

Imaduddin, H., Widyawan, & Fauziati, S. (2019). Word embedding comparison for Indonesian language sentiment analysis. *2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT)*, 426–430. https://doi.org/10.1109/ICAIIT.2019.8834536

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, *28*(1), 11–21. https://doi.org/10.1108/eb026526

Kamiş, S., & Goularas, D. (2019). Evaluation of Deep Learning Techniques in Sentiment Analysis from Twitter Data. *2019 International Conference on Deep Learning and Machine Learning in Emerging Applications (Deep-ML)*, 12–17. https://doi.org/10.1109/Deep-ML.2019.00011

Kilimci, Z. H., & Akyokus, S. (2019). The Evaluation of Word Embedding Models and Deep Learning Algorithms for Turkish Text Classification. *2019 4th International*

Conference on Computer Science and Engineering (UBMK), 548–553. IEEE. https://doi.org/10.1109/UBMK.2019.8907027

Marukatat, R. (2020). A Comparative Study of Using Bag-of-Words and Word-Embedding Attributes in the Spoiler Classification of English and Thai Text. In *Studies in Computational Intelligence* (Vol. 847). Springer International Publishing. https://doi.org/10.1007/978-3-030-25217-5_7

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, 1–12. Retrieved from https://arxiv.org/abs/1711.08609

Rahman, M. Z., Sari, Y. A., & Yudistira, N. (2021). Analisis Sentimen Tweet COVID-19 menggunakan Word Embedding dan Metode Long Short-Term Memory (LSTM). *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *5*(11), 5120–5127. Retrieved from http://j-ptiik.ub.ac.id

Rezaeinia, S. M., Ghodsi, A., & Rahmani, R. (2017). Improving the accuracy of pre-trained word embeddings for sentiment analysis. *ArXiv*, 1–15. Retrieved from https://arxiv.org/abs/1711.08609

Wang, C., Nulty, P., & Lillis, D. (2020). A Comparative Study on Word Embeddings in Deep Learning for Text Classification. *Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval*, 37–46. https://doi.org/10.1145/3443279.3443304

Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent trends in deep learning based natural language processing [Review Article]. *IEEE Computational Intelligence Magazine*, *13*(3), 55–75. https://doi.org/10.1109/MCI.2018.2840738

Zaremba, W., Sutskever, I., & Vinyals, O. (2014). Recurrent Neural Network Regularization. *ArXiv*, (2013), 1–8. Retrieved from http://arxiv.org/abs/1409.2329