Journal of Theoretical and Applied Information Technology
31st January 2022. Vol.100. No 2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

THE ACCURACY COMPARISON AMONG WORD2VEC, GLOVE, AND FASTTEXT TOWARDS CONVOLUTION NEURAL NETWORK (CNN) TEXT CLASSIFICATION

EDDY MUNTINA DHARMA[1], FORD LUMBAN GAOL[2], HARCO LESLIE HENDRIC SPITS WARNARS[3], BENFANO SOEWITO[4]

[1,2,3,4] Computer Science Department, BINUS Graduate Program – Doctor of Computer Science, Bina Nusantara University, Jakarta 11480, Indonesia

E-mail: [1]eddy.dharma@binus.ac.id, [2]fgaol@binus.edu, [3]shendric@binus.edu, [4]bsoewito@binus.edu

ABSTRACT

Feature extraction in the field of Text Processing or Natural Language Processing (NLP) has its own challenges due to the characteristics of unstructured text. Thus, the selection of the right feature extraction method can affect the performance of the classification. This study aims to compare the accuracy of 3 word embedding methods namely Word2Vec, GloVe and FastText on text classification using Convolutional Neural Network algorithm. These three methods were chosen because they are able to capture semantic, syntactic, sequences and even context around words. Therefore, the accuracy of these three methods was compared on the classification of news from the data set taken from the UCI KDD Archive, which contains 19,977 news stories and is grouped into 20 news topics. The results show that the word embedding with the Fast Text method performs the best accuracy in the classification process. In fact, the difference in accuracy of the three methods is not crucially significant, so, it can be concluded that its usage depends on the applied data set.

Keywords: *Word2Vec, Glove, Fasttext, Word Embedding, Convolution Neural Network, Text Classification*

1. INTRODUCTION

Extraction of features in the area of Text Processing or Natural Language Processing (NLP) presents unique difficulties owing to the unstructured nature of text. Thus, feature extraction is a critical step in text classification because it converts unstructured textual forms to structured textual formats that can be processed by learning model algorithms for further categorization into preset classes. Because the technique of feature extraction that is chosen has an effect on classification performance [1] , there have been many research on feature extraction targeted at increasing classification performance develepod nowdadys. Compressing data into vector space representation is one of the methods used in NLP for feature extraction. This method converts unstructured text to structured data using a term-frequency matrix. [2][3].

Historically, the Bag of Words model, which included term frequencies (TF), term frequency-inverse document frequency (TF-IDF), and n-grams, was the most often employed method for feature extraction. It is a model for decomposing text into a collection of terms based on their frequency of occurrence in a document [4] . The disadvantage of bag of words, on the other hand, is its inability to capture semantic connections between words in individual documents.

Following that, the Word Embedding Technique was created, which is the process of turning alphanumeric characters into vector shapes [5]. Each word is a vector that represents a certain dimension of a point in space. Word embedding places words that have certain characteristics, such as being in the same context or having the same semantic meaning, close together in the space.[6], [7]. In summary, word embedding may capture both the semantic and syntactic meaning of a word.

Then, in 2013, a Google team headed by Tomas Mikolov developed and released the Word2Vec technique for word embedding, which includes two models: Skip-gram and Continous Bag of Words (CBOW).[8]–[10]. Some studies using Word2Vec models include Zhang et al.(2015), using word2Vec models for sentiment analysis about clothing products in China. Performance

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

measurement of sentiment classification is based on part-of-speech and based on lexicon. The results of the experiment showed that word2vec had good performance [18]. Yang et al. (2018), using theWord2Vec model for Twitter election classification. They conducted the assessment by using convolution neural network (CNN) as its classification method while Word2Vec was aplied as its word embedding method. Their experimental results showed better results compared to classification methods such as SVM with TF-IDF and SVM with Word Embedding [19]. Jang et al. (2019), using theWord2Vec method to classify news articles and tweets, with the highest accuracy of 93.41% for news and 90.81% for tweets [20].

The next year (2014), Pennington, Socher, and Manning published a paper titled GloVe: Global Vectors for Word Representation. The paper introduces a novel word embedding technique called GloVe, which employs a co-occurrence probability ratio between words.[11]. Some studies using the GloVe model include Dhiman & Toshniwal (2020), using the GloVe and Word2Vec models to detect an event using twitter data. In addition to that, JoSE which utilizes statistics of the semantic emergence of words and word-paragraph was applied to capture contextual information in the data Their experiments resulted in better Precision, Recall and F1-Score than graph clustering-based methods [21]. Next Eke et al. (2021) conducted a study to identify Sarcasm from twitter data by using the GloVe model as a word embedding feature coupled with the Bidirectional Encoder representation and Transformer (BERT) feature which related to sentiment and syntax. Their experiments showed the highest precision at 98.5%. [22]. Then, Roman et al.(2021) conducted a citation intent classification using word embedding. They compared three models such as Glove, BERT and Infersent [23].

Then, in 2017, Bojanowski et al. developed the Fast Text technique, a refinement of the Skip-Gram model used in the word2vec method. Thus, this Fast Text approach investigates word representation by taking into account the word's subword information.[12]. Some studies using the FastText model include Hasanah et al. (2021) who conducted studies to identify people awarness of COVID-19 through citizen conversations on Twitter. It aims to help relevant parties make policies to develop appropriate emergency response strategies in the face of changes in people's behavior due to pandemics. From the data obtained, the classification process was carried out using a combination of word embedding (FastTest and Word2Vec) with deep learning methods. The classification results showed the highest accuracy of 99.4% [24]. Imtiaz et al.(2020) use a 300-dimensional FastText model to detect duplicate questions on Quora. Quora is a growing platform that consists of a collection of user-generated questions and answers. Effectively detecting duplicate questions will make it easier to find high-quality answers as well as saving time. The model

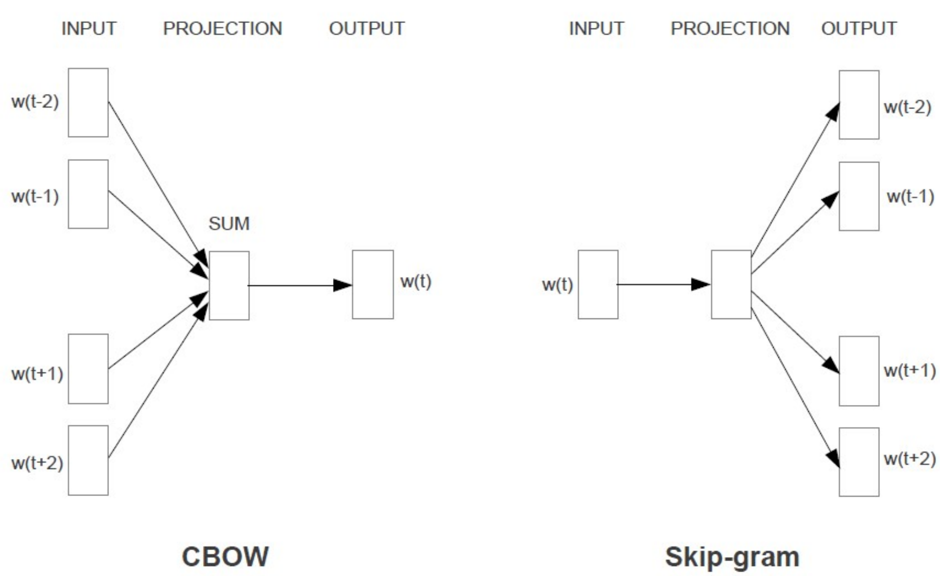


Figure 1 : CBOW Architectural Models and Skip-Gram
Source : "Efficient estimation of word representations in vector space" paper [8]

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

proposed in their study, was testing 100,000 pairs question. Then, the results of the brand experiment showed that the proposed model achieved 91.14% accuracy [25]. Then in the Le et al. (2019) was presented an innovative approach by interpreting DNA sequences as a combination of continuous FastText N-grams, which were then incorporated into the Deep Neural Network to classify them. Their approach was able to achieve cross-validation accuracy of 85.41 and 73.1% respectively at two layers [26].

Finally, the widespread usage of the word embedding model in the area of natural language processing provides an opportunity to compare the performance of the Word2Vec, GloVe, and FastText word embedding models.

2. WORD EMBEDDING

2.1 Word2Vec

The word2vec algorithm was created by Tomas Mikolov et al. in 2013. This algorithm is one of the word embedding algorithms that mapping every word in the text into vectors that can carry the semantic meaning of the word. Word2vec is one of the unsupervised learning applications using a neural network. It consists of a hidden layer called the projection layer and a fully connected layer which is trained using stochastic gradient descent with a backpropagation algorithm. The projection layer is the mapping of the word in the context of n-gram into the form of continuous vectors. Words that appear simultaneously or repeatedly in the context of N-gram have a tendency to be activated by the same weight, resulting in a correlation between words.

Weights connect the input layer with the projection layer as well as connect the projection layer with the output layer. Weights between the input layer and projection layer are represented by the V x N size W matrix, where V is the dimension of the input layer and N is the dimension of the projection layer.. Between the projection layer and the output layer, matrix W is represented by a matrix measuring N x D, where N is the dimension of the projection layer and D is the dimension of the output layer.

Word2vec relies on local information from the language [13]. The semantics learned from a particular word are influenced by the surrounding words. Wod2vec demonstrates the ability to study linguistic patterns as linear relationships between word vectors.

Two architectural models that can be used in word2vec, namely Continuous Bag-of-Word (CBOW) and Skip-Gram. Both models can be seen in figure 1.

In the CBOW model, word2vec uses words that are in preceding and following the target word and is limited to a window predicting the target word. While skip-gram uses a word to predict words that are before and after the word that is limited by the window. A window is used as a kernel to obtain input and target words. The window is shifted from the beginning to the end of the wording. As an example, when the window size is given at 2, then word2vec will consider 2 words in before and 2 words after a word associated with it. Illustrations from the window can be seen in figures 2 and 3.

2.2 Glove

Glove, global vector, a technique that takes advantage of two different approaches: count-based (e.g. PCA, principal component analysis) and direct prediction such as word2vec.

Unlike word2vec which relies solely on local information from words with local context windows, the GloVe algorithm also combines word co-occurrence information or global statistics to obtain semantic relationships between words in the corpus. GloVe uses the global matrix factorization method, a matrix that represents the appearance or absence of words in a document [11]. Word2vec is a feedforward neural network model, so it is often referred to as neural word embeddings, while GloVe is a log-bilinear model or often referred to as a count-based model. It means that GloVe learns the relationship of words by calculating how often words appear with each other in a given corpus. The probability ratio of the appearance of words has its potential to encode some form of meaning and help to improve the performance on the problem of word analogy.

The GloVe model aims to study the vector of words in such a way that the dot product of those words is equal to the logarithm of the probability of words to appear together or the probability of their co-occurence. The Glove Model can be written down as follows. [11] :

$$w_i^T + \vec{w}_k + b_i + \vec{b}_k = log(X_{i_k}) \quad (1)$$

where w is the word vector, $\vec{w}$ is the context word vector, bi and bk are scalar biases for the i-word

Journal of Theoretical and Applied Information Technology
31st January 2022. Vol.100. No 2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

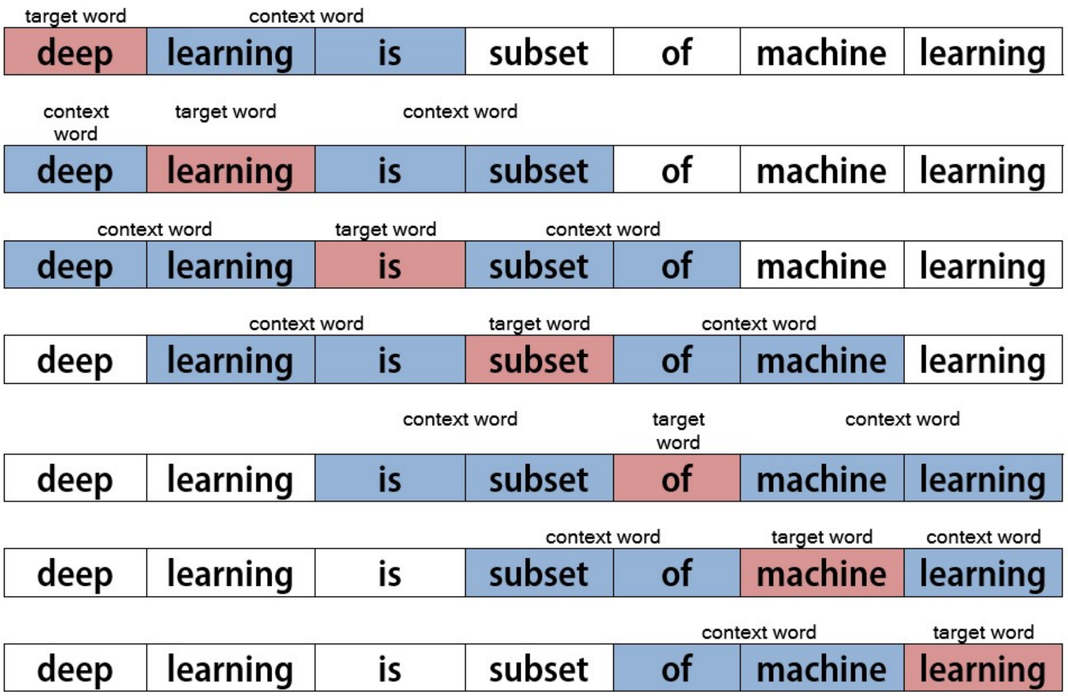


Figure 2 : Illustration of CBOW Architecture with window size 2

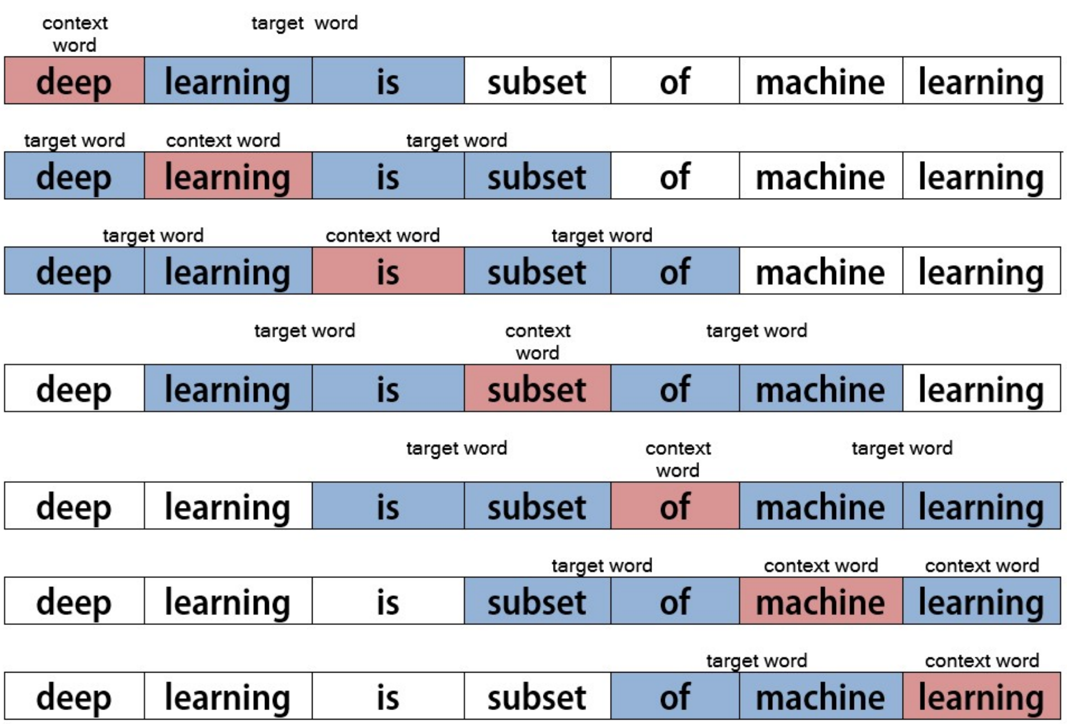


Figure 3 : Skip-Gram Architecture illustration with window size 2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

and the k-word context. X is a word co-occurrence matrix in which each Xik element represents the number of times the word i appears in the k-word context. Context word itself is a collection of words consisting of words that are in before and after the word i as much as windows size given. Then each word will be given weighting by way of 1 / distance. the distance in this case belongs to the distance between the context word and the position of the word. To calculate the function of the weight f(Xik), can be conducted within an equation as follows [11]:

$$f(X_{i_k}) = \begin{cases} (\frac{X_{i_k}}{xmax})^{\alpha}; if X_{i_k} < xmax \\ 1; lainnya \end{cases} \quad (2)$$

Then by incorporating equations (1) and (2) into a cost function produce a model as follows [11]:

$$J = \sum_{i,k=1}^{V} f(X_{ik})\left(w_i^T \overrightarrow{w_k} + b_i + b_k - \log(X_{ik})\right)^2 \quad (3)$$

2.3 FastText

FastText can be defined as a word embedding method which is part of the word2vec development [12], Developed by Facebook's AI research team, this model was inspired by research conducted by Mikolov et al. and managed to show that their model was able to train at 1 billion words within 10 minutes, within performing quality of results compared to other models. [14]. The FastText model architecture is similar to the CBOW architecture in Word2Vec, however, it has a hierarchical structure and represents words in dense vector form. In addition, there is a hidden layer between the input layer and the output layer, as shown in figure 4. [14]

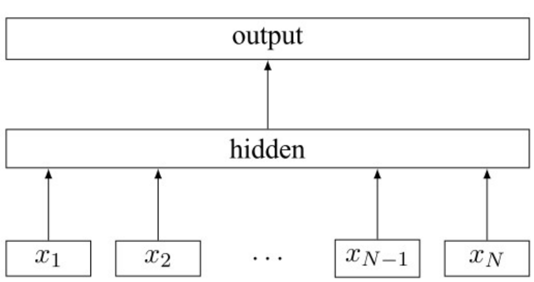


Figure 4 : Model architecture offastText for a sentence with n-gram feature x_1, x_2, ... x_N
Source : "Bag of tricks for efficient text classification" paper [14]

This approach investigates word representation by including sub word information. Each word is represented by an n-gram sequence of characters. Thus, it may aid in the comprehension of shorter words and enable people to comprehend the suffixes and prefixes of the word. Each n-gram character has a vector representation, and words are represented as the total of those vector representations. After the word is represented using the n-gram character, it is trained using the CBOW architecture to determine the word's embedding vector.

By and large, models that examine the vectorization of words disregard the morphology of the word, since each word has a unique vector. As a result, it imposed a restriction on the representation of terms from languages with a big vocabulary and a high proportion of uncommon words.

FastText is an excellent performer, capable of rapidly training models on huge datasets and providing representations for words that do not exist in the training data. When a word does not occur during model training, its vector embedding may be determined by breaking it down into n-grams.

3. METHOD

The stages in this study is described in Figure 5, starting from dataset collection then continued to training process, model testing, and finally it is ended by calculating the accuracy.

3.1 Data Set

The data set used in this study was a data set taken from the UCI KDD Archive [15], which contained 19,977 news stories and was grouped into 20 newsgroups or topics, thus, each news topic consisted of an average of 1,000 news stories. This data was be divided into 11,986 training data and 7,991 test data.

3.2 Text Classification

Deep Learning has shown its dependability in the area of Natural Language Processing by successfully solving a variety of categorization difficulties. Consider the Convolutional Neural Network (CNN), which is capable of effectively representing meaningful representations of sentences in language categorization and modeling [16].

In this research, one-dimensional CNN is used to model the categorization of news articles into a

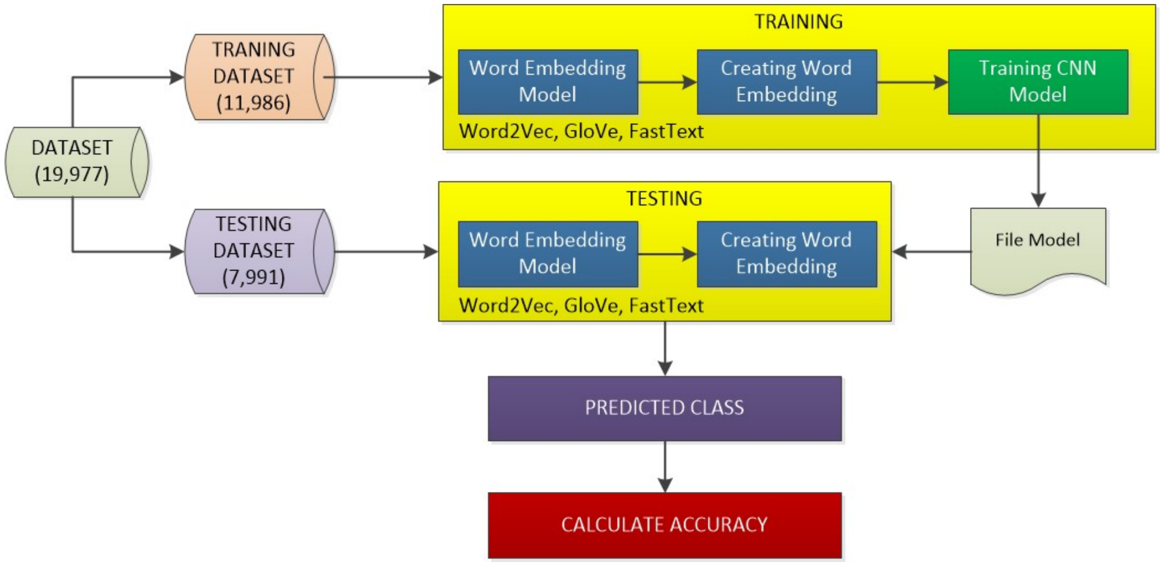


Figure 5 : Block diagram of the overall architecture of our method

variety of themes in the dataset since it is very successful at reducing the characteristics of fixed-length segments throughout the whole dataset and works well for Natural Language Processing (NLP) issues.

Figure 6 shows the used of CNN architecture, consisting of input layer, convolutional, max pooling, and fully connected.

a. Input Layer
To begin, the text of each news item was converted into a 300-dimensional word vector representation using word embedding (Word2vec, GloVe, FastText), and then presented as a document vector. Each news subject includes 1000 news articles; therefore, the input matrix will be 1000 x 300.

b. Convolutional Layer
The convolutional layer is composed of neurons that are organized in a pattern to create a filter. This layer included 128 filters with a window size of 5 that were arranged vertically over the whole input matrix. After performing a "dot" operation between the filter weight and the weight of the input matrix, as well as a non-linear operation with the ReLU activation function, an activation map or feature map is produced that includes significant lower-dimensional features in the first hidden layer. Then, the first hidden layer's feature map will be used as the input for the second convolutional layer, and so on. Three convolutional layers are used to create the CNN model.

c. Max Pooling Layer
The Max Pooling Layer will extract the greatest value from components within a five-dimensional window, ensuring that the most critical information is extracted from the feature map of the convolution results.

d. Fully Connected Layer
The preceding hidden layer's output, a reshaped feature map, is linked to the output layer to be categorized. At this layer, softmax activation functions and loss functions are employed, as multiclass output variables are encoded using a one-hot encoding consisting of the values 0 and 1, respectively.

In addition to utilizing Adam's optimization algorithm, the CNN was trained with batch size=128 and epoch=20. To prevent overfitting, each hidden layer would use a dropout regulation technique [17] with a rate of 0.5, which would randomly deactivate 50% of neurons during the training phase.

3.3 Accuracy Calculation

A small application was developed using the python 3.9 programming language and the jupyter notebook 6.4.0 IDE, with a dataset of 20 newsgroups, to evaluate the accuracy of three word embedding techniques, namely Word2Vec, GloVe, and FastText categorized using the Convolutional Neural Network (CNN) algorithm. The hardware

Journal of Theoretical and Applied Information Technology
31st January 2022. Vol.100. No 2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

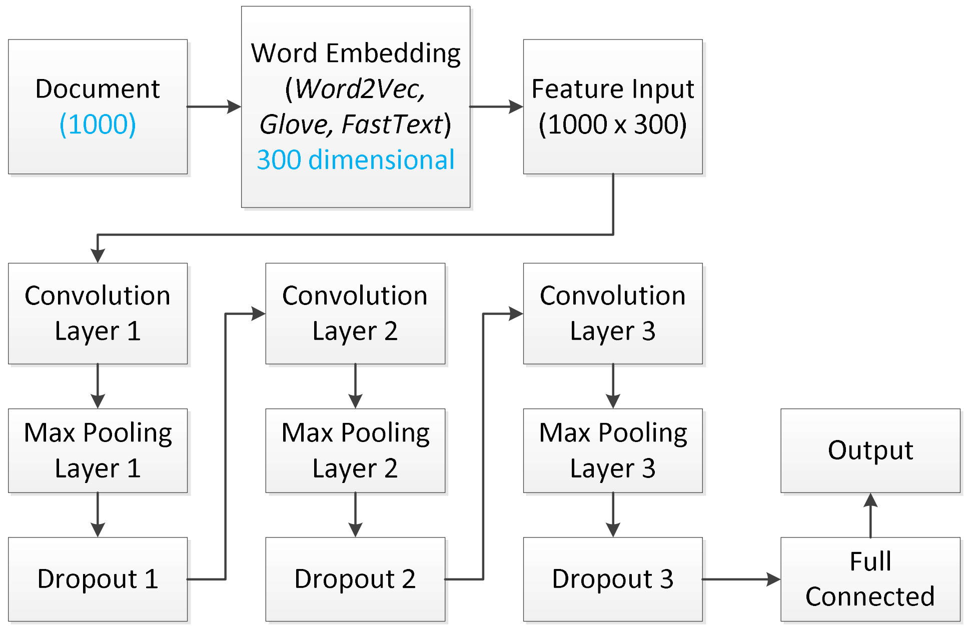


Figure.6 : CNN Archihetcure for Text Classification

specifications used at the time of the experiment were shown in Table 1..

Table 1: Hardware Specifications used by experiments

Item	Detail
OS	Microsoft Windows 10 Pro, 64-bit
CPU	Intel(R) Core(TM) i7-1065G7
RAM	16GB
GPU	NVDIA GeForce MX350
CUDA	11

Table 1 shows the hardware specifications for running the CNN algorithm we use. We're trying to maximize deep learning speed through Compute Unified Device Architecture (CUDA) yang dikembangkan NVIDIA [27]. CUDA is a General-Purpose computing Graphics Processing Units technology that enables the use of parallel processing algorithms in the Graphics Processing Unit (GPU). We used CUDA to accelerate the research process via processing CPU-processed operations on the GPU.

While, in order to testing the performance of Word2Vec, GloVe, and FastText, the accuracy formula was used as follows

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

where True Positive (TP) denotes the number of real positives among the predicted positives, and True Negative (TN) denotes the number ofreal negatives ofthe predicted negatives. Similarly, False Negative (FN) denotes the number ofreal positives among predicted negatives, and False Positive (FP) denotes the number ofreal negatives among predicted positives. Therefore, accuracy denotes the proportion of documents classified correctly by CNN among all documents [28].

4. RESULT AND DISCUSSION

The training process uses 11,986 training data and one target with 20 newsgroup classifications. While the parameters of the CNN model used are shown in Table 2.

Table 2 : CNN Parameters used

Parameter	Value
Epochs	1 - 20
Learning Rate	0.001
CNN dropout probability	0.5
Optimization Algorithm	Adam

After experimenting, the results of training accuracy and testing data were obtained in the dataset of 20 newsgroups shown in Table 3 and Figure 7..

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 3 : Word Embedding Accuracy Measurement Results

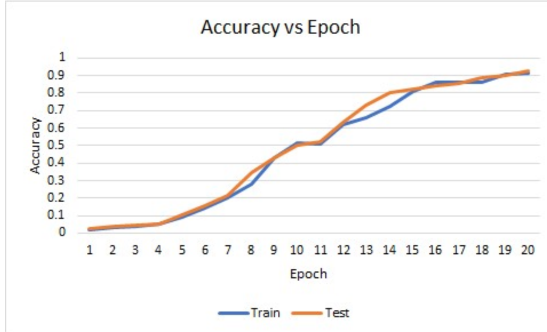
Word Embedding	Accuracy (%)
Word2Vec	92.5
GloVe	95.8
FastText	**97.2**

Based on the results of the performance evaluation of the three word embedding (Word2Vec, GloVe and FastText), it was obtained that FastText performance is superior. FastText has the ability to provide word representations that do not appear in training data or are able to overcome *out of vocabulary problems.* Words that are not found during the *training*process, the word is broken down into *n-grams* in the form of a collection of syllable sequences to get *embedding* the vector. This advantage is evident from the results of experiments that showed better FastText performance compared to Word2vec and GloVe for both datasets of 20 news groups.

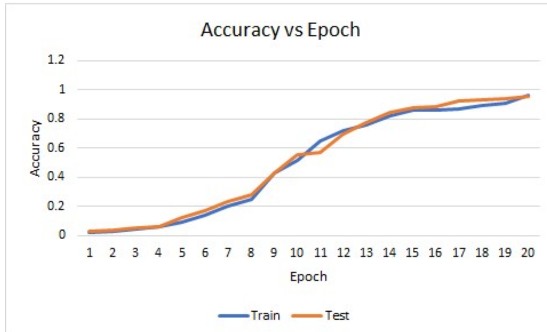
The results of this experiment are in line with some studies showing word embedding FastText works more effectively than in some datasets [29]. Kang et al. (2016) found that Word2Vec provided the best performance on word embedding comparisons of Word2vec CBOW, GloVe, and Collobert & Weston for English and Cross-Lingual Chinese Word Sense Disambiguation [30].

Another study conducted by Li et al. (2018), where they conducted a study comparing three word embeddings (Word2Vec, GloVe and FastText)in the case of tweet crisis classification. Word embedding used is pre-trained from the corpus of Google News, Wikipedia or Twitter, and word embedding built from the crisis-specific domain of the tweet corpus. . The result is that crisis-specific embedding is more suitable for more specific classifications of crisis-related tweets, while pre-trained embedding is more suitable for more general classifications. Based on those three types of word embedding (word2vec, GloVe and FastText), GloVe performed the best for all three datasets used (CrisisLexT6, CrisisLexT26 and 2CTweets) [31].

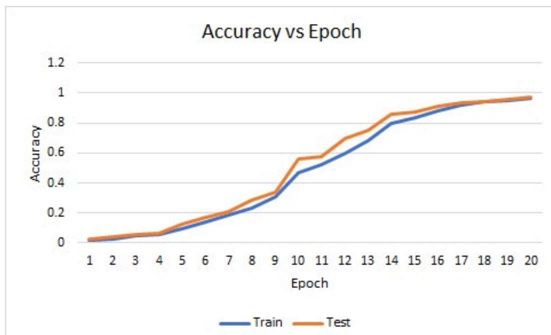
Naili et.al (2017) investigated the topic segmentation in Arabic and English and found that word2vec and GloVe were more effective than LSA for both languages. Compared to GloVe, Word2Vec produces the best word vector representations with small dimensional semantic space. While the quality of the topic segmentation depends on the language used. The quality of segmentation in Arabic is worse than English due to its high complexity[32].



(a) Word2Vec



(b) Glove



(c) FastText

Figure 7: Graph accuracy training and testing data in dataset 20 newsgroups, with Word Embedding: Word2Vec(a), Glove(b) and FastText(c)

Based on the results of experiments and several studies comparing the performance of Word2vec, GloVe, and FastText, it is proven that these three word embeddings have competitive performance. Word embedding performance depends on the dataset used and the domain of the problem solved.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

5. CONCLUSION

There are several methods that can be used to represent text into vector form, one of which is by using word embedding. Word2vec is a word embedding in the form of a weight matrix obtained from the results of unsupervised neural network learning training. Wod2vec relies on local information from the language. The semantics learned from a particular word are influenced by the surrounding words. Unlike Word2vec, GloVe shows how to engage the global statistical information contained in a document. The semantic meaning of a word is influenced not only by the words around it but also the global statistical information of the document. GloVe uses the ratio of co-occurance probability between words. FastText is built from a word2vec model that maps subword/syllables. .

According to the results of tests of the accuracy of the three word embedding, FastText outperforms Glove and Word2vec for the dataset of 20 newsgroups, the accuracy is 97.2% for FastText, 95.8% for Glove and 92.5% for Word2Vec. Word2vec and GloVe are unable to represent vectors of words that are not in the corpus (out of vocabulary). Unlike FastText which can be relied upon for this out of vocabulary problem. The best performance of the experiment was obtained using Word Embedding FastText. However, the difference in accuracy between the three techniques is not statistically significant, demonstrating that these three methods have competitive performance. The accuracy of these three word embeddings is dependent on the data set used and the domain of the issue addressed, therefore other data sets and domain problems to be solved may be added for future study.

REFRENCES:

[1] M. Eklund, "Comparing Feature Extraction Methods and Effects of Pre-Processing Methods for Multi-Label Classification of Textual Data," 2018.

[2] D. J. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*. Cambridge, MA, USA: MIT Press, 2001.

[3] C. D. Manning, P. Raghavan, and H. Schütze, *An Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2009.

[4] P. Rosado, F. Reverter, E. Figueras, and M. Planas, "Semantic-Based Image Analysis with the Goal of Assisting Artistic Creation," in *International Conference, ICCVG 2014 Warsaw, Poland, September 15–17, 2014 Proceedings*, 2014, pp. 526–533, doi: 10.1007/978-3-319-11331-9.

[5] D. Dessì, D. R. Recupero, and H. Sack, "An Assessment of Deep Learning Models and Word Embeddings for Toxicity Detection within Online Textual Comments," in *Electronics*, Mar. 2021, vol. 10, no. 7, p. 779, doi: 10.3390/electronics10070779.

[6] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A Neural Probabilistic Language Model," in *Journal of Machine Learning Research*, 2003, vol. 3.

[7] Y. Bengio, H. Schwenk, J. S. Senécal, F. Morin, and J. L. Gauvain, "Neural probabilistic language models," in *Studies in Fuzziness and Soft Computing*, 2006, vol. 194, no. 2006, pp. 137–186, doi: 10.1007/10985687_6.

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, 2013, no. October.

[9] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations ofwords and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, 2013, pp. 1–9.

[10] T. Mikolov, Q. V. Le, and I. Sutskever, "Exploiting Similarities among Languages for Machine Translation," 2013, [Online]. Available: http://arxiv.org/abs/1309.4168.

[11] J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, vol. 31, no. 6, pp. 1532–1543, doi: 10.3115/v1/D14-1162.

[12] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," in *Transactions of the Association for Computational Linguistics*, Dec. 2017, vol. 5, pp. 135–146, doi: 10.1162/tacl_a_00051.

[13] T. Ganegedara, “Intuitive Guide to Understanding GloVe Embeddings,” *towards data science*, 2019. https://towardsdatascience.com/light-on-math-ml-intuitive-guide-to-understanding-glove-embeddings-b13b4f19c010 (accessed Jul. 02, 2021).

[14] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, “Bag of tricks for efficient text classification,” in *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference*, 2017, vol. 2, pp. 427–431, doi: 10.18653/v1/e17-2068.

[15] The UCI KDD Archive, “Information and Computer Science University of California, Irvine.” https://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html (accessed Jul. 02, 2021).

[16] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, “A Convolutional Neural Network for Modelling Sentences,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014, pp. 655–665, doi: 10.3115/v1/P14-1062.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting Nitish,” in *Journal of Machine Learning Research*, 2014, vol. 15.

[18] 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, p. e0220976, Aug. 2019, doi: 10.1371/journal.pone.0220976.

[19] D. Zhang, H. Xu, Z. Su, and Y. Xu, “Chinese comments sentiment classification based on word2vec and SVMperf,” *Expert Syst. Appl.*, vol. 42, no. 4, pp. 1857–1863, Mar. 2015, doi: 10.1016/j.eswa.2014.09.011.

[20] X. Yang, C. Macdonald, and I. Ounis, “Using word embeddings in Twitter election classification,” *Inf. Retr. J.*, vol. 21, no. 2–3, pp. 183–207, Jun. 2018, doi: 10.1007/s10791-017-9319-5.

[21] M. Roman, A. Shahid, S. Khan, A. Koubaa, and L. Yu, “Citation Intent Classification Using Word Embedding,” *IEEE Access*, vol. 9, pp. 9982–9995, 2021, doi: 10.1109/ACCESS.2021.3050547.

[22] C. I. Eke, A. A. Norman, and L. Shuib, “Context-Based Feature Technique for Sarcasm Identification in Benchmark Datasets Using Deep Learning and BERT Model,” *IEEE Access*, vol. 9, pp. 48501–48518, 2021, doi: 10.1109/ACCESS.2021.3068323.

[23] A. Dhiman and D. Toshniwal, “An Approximate Model for Event Detection From Twitter Data,” *IEEE Access*, vol. 8, pp. 122168–122184, 2020, doi: 10.1109/ACCESS.2020.3007004.

[24] N. A. Hasanah, N. Suciati, and D. Purwitasari, “Identifying Degree-of-Concern on COVID-19 topics with text classification of Twitters,” *Regist. J. Ilm. Teknol. Sist. Inf.*, vol. 7, no. 1, p. 50, Feb. 2021, doi: 10.26594/register.v7i1.2234.

[25] Z. Imtiaz, M. Umer, M. Ahmad, S. Ullah, G. S. Choi, and A. Mehmood, “Duplicate Questions Pair Detection Using Siamese MaLSTM,” *IEEE Access*, vol. 8, pp. 21932–21942, 2020, doi: 10.1109/ACCESS.2020.2969041.

[26] N. Q. K. Le, E. K. Y. Yapp, N. Nagasundaram, and H.-Y. Yeh, “Classifying Promoters by Interpreting the Hidden Information of DNA Sequences via Deep Learning and Combination of Continuous FastText N-Grams,” *Front. Bioeng. Biotechnol.*, vol. 7, Nov. 2019, doi: 10.3389/fbioe.2019.00305.

[27] Kirk D. NVIDIA CUDA software and GPU parallel computing architecture. ISMM. 2007. pp. 103–104

[28] M. Junker, R. Hoch, and A. Dengel, “On the evaluation of document analysis components by recall, precision, and accuracy,” in Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR ’99 (Cat. No.PR00318), 1999, pp. 713–716, doi: 10.1109/ICDAR.1999.791887.

[29] E. S. Qorina, A. Zamhari, H. Hasan, K. Hulliyah, and D. Saepudin, “Comparative Analysis of the Performance of the Fasttext and Word2vec Methods on the Semantic Similarity Query of Sirah Nabawiyah Information Retrieval System: A systematic literature review,” in 2020 8th International Conference on Cyber and IT Service Management (CITSM), Oct. 2020, pp. 1–4, doi: 10.1109/CITSM50537.2020.9268827.

[30] H. J. Kang, T. Chen, M. K. Chandrasekaran, and M.-Y. Kan, "A Comparison of Word Embeddings for English and Cross-Lingual Chinese Word Sense Disambiguation," in Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA2016). Osaka, Japan, 2016, pp. 30–39, [Online]. Available: http://arxiv.org/abs/1611.02956.

[31] H. Li, X. Li, D. Caragea, and C. Caragea, "Comparison of Word Embeddings and Sentence Encodings as Generalized Representations for Crisis Tweet Classification Tasks," Proc. ISCRAM Asian Pacific 2018 Conf., no. November, pp. 1–13, 2018.

[32] M. Naili, A. H. Chaibi, and H. H. Ben Ghezala, "Comparative study of word embedding methods in topic segmentation," Procedia Comput. Sci., vol. 112, pp. 340–349, 2017, doi: 10.1016/j.procs.2017.08.009.