Semantic Classification of Scientific Sentence Pair Using Recurrent Neural Network

Agung Besti
Department of Informatics
Universitas Jenderal Achmad Yani
Cimahi, Indonesia
bestiagung@gmail.com

Ridwan Ilyas
Department of Informatics
Universitas Jenderal Achmad Yani
Cimahi, Indonesia
ilyas@lecture.unjani.ac.id

Fatan Kasyidi
Department of Informatics
Universitas Jenderal Achmad Yani
Cimahi, Indonesia
fatan.kasyidi@lecture.unjani.ac.id

Esmeralda Contessa Djamal
Department of Informatics
Universitas Jenderal Achmad Yani
Cimahi, Indonesia
esmeralda.contessa@lecture.unjani.ac.id

***Abstract*— One development of Natural Language Processing is the semantic classification of sentences and documents. The challenge is finding relationships between words and between documents through a computational model. The development of machine learning makes it possible to try out various possibilities that provide classification capabilities. This paper proposes the semantic classification of sentence pairs using Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM). Each couple of sentences is turned into vectors using Word2Vec. Experiments carried out using CBOW and Skip-Gram to get the best combination. The results are obtained that word embedding using CBOW produces better than Skip-Gram, although it is still around 5%. However, CBOW slows slightly at the beginning of iteration but is stable towards convergence. Classification of all six classes, namely Equivalent, Similar, Specific, No Alignment, Related, and Opposite. As a result of the unbalanced data set, the retraining was conducted by eliminating a few classes member from the data set, thus providing an accuracy of 73% for non-training data. The results showed that the Adam model gave a faster convergence at the start of training compared to the SGD model, and AdaDelta, which was built, gave 75% better accuracy with an F1-Score of 67%.**

Keywords—NLP, Semantic Classification Similarity, Recurrent Neural Networks, LSTM.

I. INTRODUCTION

Writing scientific papers cannot be separated from direct quotations or paraphrases that have semantic sentences on each cited paper. This fact shows a theory to the reader what has been discovered by other scientists and shows the difference or similarity of opinion with other scientists in strengthening the novelty of research built. Citing sentence quotations have several categories [1]. However, the support depends on their implementation; one of them is to get more specific and similar information between cited scientific works [2].

Measuring Semantic Textual Similarity (STS) is the closeness of a sentence from each pair of texts; every two sentences are counted similar to other sentences. STS has the concept of how words can infer the level of equivalence of a couple of meaning sentences are compared to produce a score scale of 0 to 5. The type of relation consists of “Equivalent”, “Similar”, “Specific”, “No Alignment”, “Related” and “Opposite” [3]. Previous studies compared classification parameters in finding the best score between Semantic Textual Similarity (STS) relationships [4].

Semantic sentence similarity approaches can be analyzed through Natural Language Processing (NLP). NLP is a branch of artificial intelligence that focuses on natural language processing. Some of the benefits that can be used in NLP one of the information retrieval [5], question-answer [6], plagiarism detection [7], machine translation [8], sentiment analysis [9], text summarization [10], short answer scoring [11] and paraphrase identification [12].

Other studies used STS to understand the meaning based on the French corpus [13], the meaning of the Portuguese language [14], comparing distributed. One-hot representation to understand the meaning of clinical sentences [15], improve the quality of information from Electronic Health Records (EHRs) to the resulting data redundancy [16], see reliable information from a website [17], and get an understanding of news from the headlines delivered [18].

Other studies used citation sentences as a scientific sentence to define citation categories for filtering scientific references [2]. The categories are ‘problem’, ‘other’, ‘useModel’, ‘useTool’, and ‘useData’. For classification, Support Vector Machine (SVM) with a linear kernel is used to build the classification model and SMOTE as a sampling technique for imbalanced datasets. After the experiment, the result of the f-measure is achieved at 61.2%. However, some features did not have a good effect on some categories, and it is necessary to analyze every feature that can give a more significant effect to improve the performance of model classification.

Research about sentence pairs detection used Manhattan Recurrent Neural Network (MRNN) [12]. For checking the connection of each word, using Long Short Term Memory (LSTM). From the experiment, it achieved an accuracy of 82.54%. This research takes an opportunity to enhance the difficulty of detection into a semantic level, not only paraphrase.

Other studies used STS to predict input-response relationships in a conversation [19] where this is a common question from the SemEval 2017 Community Question Answer (CQA) evaluation. The prediction model is developed from the Deep Averaging Network (DAN) and Transformer. It achieves various resulted from 66.8-89.4% with many configurations. Another used STS for the multilingual word to find a similar word meaning in various languages [20]. The communications involved English, French, and Spanish. Recurrent Neural Network (RNN) with Long Short Term Memory (LSTM) was used to classify the similarity across multiple languages in word representation. After the experiment, RNN gave an accuracy of 70.74 % with LSTM. However, there is another method that is more sophisticated

to enhance the performance, especially for a specific language that has complicated grammar.

Recurrent Neural Networks (RNN) is one part of deep learning because data is processed through many layers. RNN can be applied in text data processing and data sequences where processing is called repeatedly to process sequential data input. There is a variant of the settings of RNN, namely Long Short-Term Memory (LSTM), which is used to manage information to be processed and information that is forgotten in a memory.

Research on textual semantic similarity is needed for various fields of NLP, one of which is to interpret the meaning of a sentence. As for previous studies measuring textual semantic similarity using the Vrep or UWB system [4], other studies measuring similarity using Siamese similarity to identify paraphrases in sentences of scientific work [12]. SemEval has held STS assignments in the last few years since 2012. Participants who joined competed in starting the STS assignment as the initial assignment of STS. In 2016 the task became an independent task [3].

This paper proposed a semantic similarity measurement system of scientific sentence pairs in English that can be used to understand the meaning in a literature review that supports or opposes each other. The steps taken to measure semantic similarities in scientific sentences are sentence extraction, sentence labeling, converting sentences to vectors, and calculating weights using RNN and LSTM. The results of this study can be used to measure the semantic level of a scientific sentence to understand the meaning in a literature review, and it is easier to obtain the knowledge to strengthen the argument of the ongoing research.

II. METHODS

A. Dataset

The dataset used is divided into training data and validation data. The training data used in this study were 1086 data sets, validation data were 535 data sets, and for test data were 406 data sets from 2027 scientific sentence pairs, which were divided into six classes—the distribution of data used in the model to be built as shown as Table I.

TABLE I. DATASET COMPOSITION

Score	Class	Number of Pairs
5	Equivalent	822
4	Similar	714
3	Specific	451
2	No Alignment	32
1	Related	6
0	Opposite	2

Labeling is carried out by annotators who are experts in English. The annotator predicts both sentences and concludes the similarity of meaning between pairs of scientific sentences. The annotator will give the label "Equivalent" if the sentences being compared convey the same meaning semantically, on the "Similar" label if the sentences being compared are mostly equal but there are some unimportant details, on the "Specific" label if the sentences being compared have a meaning that is comparable same. However, each sentence information is more specific semantic than the pair of sentences. Class of "No Alignment" if the sentences being compared are not equivalent, but provide some detailed information, on the "Related" label if the sentences being compared do not have equivalent meanings, but still on the same topic, on the name "Opposite" if the sentences being compared inform a different problem [3].

This research, through the stages of text processing, starts from text cleaning, case folding, tokenizing, and word embedding using Word2vec to extract sentences into vectors, then using RNN to provide training weights and Softmax to determine the semantic level of scientific sentences. Following is the computational model architecture built in Fig 1.

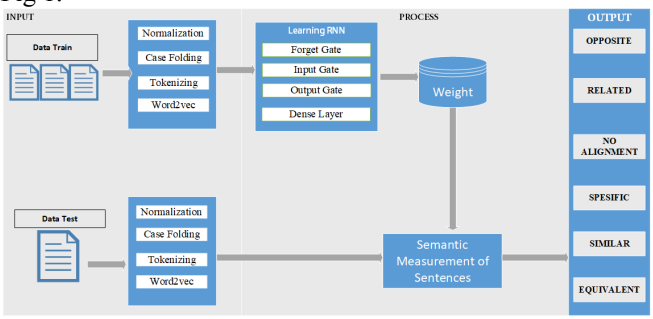

Fig. 1. Semantic classification using RNN

B. Pre-processing

Pre-processing contains text cleaning, case folding, and tokenizing. Text cleaning is the process of removing symbols to reduce noise in sentences so that it can easily detect patterns in text data. Case folding is a process to change the entire text in the corpus into a lower case so that the whole word is the same as the same entity. Tokenizing is the process of cutting or separating each sentence into several parts/tokens. So the input sentence becomes a collection of words in the list.

C. Word2vec

Word2vec is a method to convert each word of a sentence as a vector used the size of the Embedding Size. Word2Vec implements neural networks to calculate the contextual and semantic similarity of each word [21]. The results of contextual and semantic similarity become references in representing the relationship of a word to a vector [22]. The vector has a value between -1 to 1, which is the result of learning from the neural network algorithm. From these features, each word will have a vector that represents the meaning of the word.

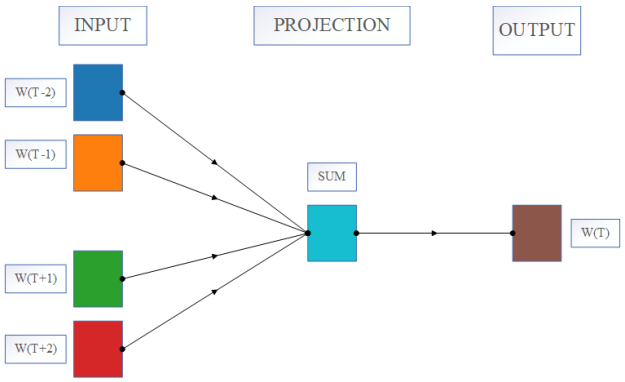

Fig. 2. Word Embedding CBOW

Word2vec has a Continuous Bag-of-Word (CBOW) and Skip-Gram architecture. The CBOW architecture will study the probability distribution of context with predetermined windows. Following CBOW architecture in Fig. 2. The Skip-Gram architecture will study the probability distribution of words in context with predetermined windows. Next is the Skip-Gram architecture in Fig. 3.

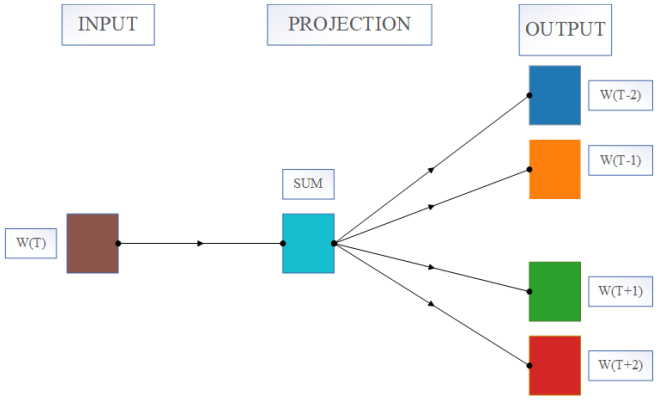

Fig. 3. Word Embedding Skip-Gram

After labeling, feature extraction is performed to get the vector value of each pair of sentences using Word2vec with a window size = 100. The representation of words into vectors is shown in Fig. 4.

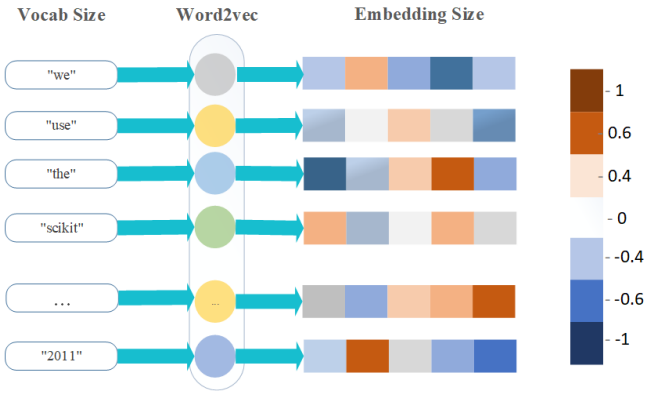

Fig. 4. Representation Word2vec

D. *Recurrent Neural Networks*

After the sentence goes through the Text Processing stage, it enters the Recurrent Neural Network (RNN) architecture model design stage. RNN is one of Deep Learning which is designed to process sequential data in recognizing specific patterns in a text. RNN can store memory (feedback loops) that make it possible to recognize data patterns well, then use them to make accurate predictions. The concept of RNN can store information from the past is by looping in its architecture that automatically keeps information from the past stored RNN has several types of processing. One of them is Many-to-One, which in this process provides several word inputs as input and produces one output as class representation. RNN has one variant, namely Long Short-Term Memory (LSTM), which is used to store important information in memory in processing data sequences so as to produce the weight used in classification calculations [23]. The following RNN architecture in Fig 5.

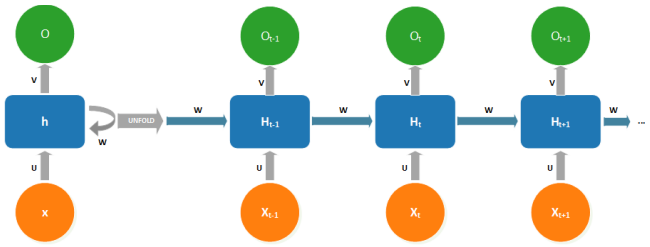

Fig. 5. RNN Architecture

LSTM is one way to calculate hidden states, where memory in LSTM is called cells that take input from the previous state and current information. The collection of cells can decide what will be stored in memory and what will be deleted from memory. LSTM combines the last state cell (C_{t-1}) hidden state memory (H_{t-1}) and input (X_t), which can be seen in Fig. 6 for LSTM cell architecture.

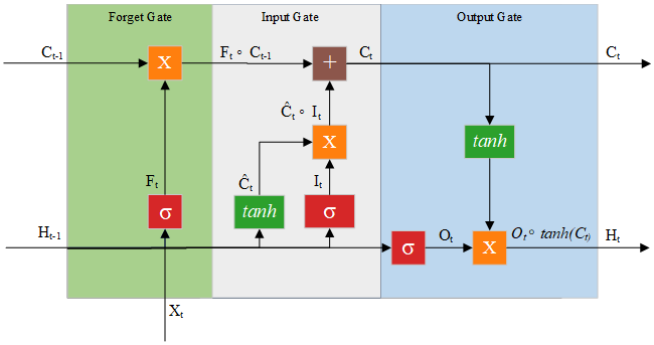

Fig. 6. Cell LSTM

LSTM has three gates, namely, forget gate as the first gate to determine the information to be removed or used from cells that have information from previous cells using the Sigmoid layer like (1).

$$F_t = \sigma(W_f.[H_{t-1}, X_t] + B_f) \quad (1)$$

The second gate is the input gate to use the sigmoid. The layer is a vector that the value is updated used (2) and (3).

$$\tilde{C}_t = \tanh(W_c.[H_{t-1}, X_t] + B_c) \quad (2)$$

$$I_t = \sigma(W_i.[H_{t-1}, X_t] + B_i) \quad (3)$$

Then cells from (1), (2), and (3) will be updated using (4).

$$C_t = F_t * C_{t-1} + I_t * \tilde{C}_t \quad (4)$$

The last gate is the output gate that will be calculated based on the renewal of the cell and sigmoid layer that determines the cell to be taken as the final result (5) and (6).

$$O_t = \sigma(W_o.[H_{t-1}, X_t] + B_o) \quad (5)$$

$$H_t = O_t * \tanh(C_t) \quad (6)$$

Where, I_t, F_t, O_t is the gate input, forget, output, C_t is the internal memory unit which is a combination of a previous memory, C_{t-1} is the previous memory, X_t is input at each time step t at this time, H_{t-1} is hidden state previously, W_f, W_c, W_i, W_o is the weight matrix, b_f, b_c, b_i, b_o are refractive vectors, σ is

a sigmoid activation function with a range (0, 1) and *tanh* is a tangent activation function with a range (-1, 1).

The output value of the gate output enters the Dense layer using the Softmax activation function of x_i, which aims to convert the output into a probability for each class of n unit that can be seen in (7).

$$S(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \quad (7)$$

After the probabilities of each class are obtained from Softmax activation results, then calculate the error of the predicted probability and the value on the class label at the output layer using cross-entropy as in (8).

$$Loss(S, L) = -\sum_i L_i \log(S_i) \quad (8)$$

Where *S* is the result of the Softmax value, *L* is the class label, and *i* is the unit index of each output or label.

This research extracted data in the form of sentences from scientific sentences in English into vectors, which were previously labeled by annotators. Extraction uses Word2vec to get the vector output of each word. Vector values will be trained and enter the weighting process using LSTM. The resulting weights are classified using Softmax activation.

E. Weight Correction

In increasing weight during training, there are several optimization methods in improving network weights based on loss functions. Where this method of learning by moving forward iteratively to find the optimal weight in order to get the minimum value of the cost function that represents the level of error when predicting the target. This method is a Gradient Descent method, which is derived through the Adaptive Moment Optimization (Adam), Stochastic Gradient Descent (SGD), and Adaptive Learning Rate (Adadelta).

1) Adaptive Moment Optimization

Adaptive Moment Optimization (Adam) is an adaptive learning method for calculating individual learning levels for each parameter that can reduce errors when making predictions. Adam uses the first-moment gradient estimation, v_t as the average exponential gradient in (9) and the second moment, s_t, as the average exponent squared in (10) to adapt the learning rate for each weight. Then, to determine the learning step by multiplying the learning rate by the gradient average and dividing it by the square root of the average of the exponential results of the gradient's square in (11) and updating the weights in (12).

$$v_t = \beta_1 * v_{t-1} - (1 - \beta_1) * g_t \quad (9)$$

$$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * {g_t}^2 \quad (10)$$

$$\Delta\omega_t = -\square \frac{v_t}{\sqrt{s_t + \epsilon}} * g_t \quad (11)$$

$$\omega_{t+1} = \omega_t + \Delta\omega_t \quad (12)$$

2) Stochastic Gradients Descent

Stochastic Gradient Descent (SGD) is a derivative learning method where SGD measures changes in functions along with changes in input values. The SGD method in evaluating gradients uses random data samples from several parts of the training data in one iteration. Whereas in Gradient Descent in finding local optimum it can be a waste of time because it uses all training data. Learning using SGD in (13) where θ is the parameter or weight of the function sought, $\square$ is the initial learning rate, x (i) and y (i) are the parameter labels on the training data. The advantage of SGD is that it did not use as much memory as the Gradient Descent so that it can converge faster than traditional Gradient Descent.

$$\theta = \theta - \square . \nabla_\theta J(\theta; x^i; x^i) \quad (13)$$

3) Adaptive Learning Rate

Adaptive Learning Rate (Adadelta) helps increase the probability of finding a solution in the next iteration by accumulating gradients, thus finding a fast way to achieve convergence. Learning with AdaDelta can be seen in (14). Where $E[g^2]_t$ expresses the accumulated gradient for iteration *t*, based on the previous gradient.

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1 - \gamma)\, g_t^2 \quad (14)$$

III. RESULT AND DISCUSSION

After getting the vector value in each pair of scientific sentences, it is then processed by RNN. This research extracted unique words available in the corpus, as many as 3446, to produce 344,600 neurons with the size of each word 100 vectors. The use of word embedding learning Word2vec uses CBOW and Skip-Gram, for the use of optimization models chosen to correct network weights, i.e., Adaptive Moment Estimation (Adam), Stochastic Gradient Descent (SGD), and Adaptive Learning Estimation (AdaDelta).

The experiment was carried out in four ways. First, testing with variations in learning models. The second effect is the distribution of data sets from each class. Third, examining the influence of the model for the conversion of sentences into vectors (Word2vec). Fourth is testing the validity of the model with the F1-Score of the best combination in the three previous ways. Variations in optimization models, distribution of data sets, and Word2Vec models give accuracy, as shown in Table II and Loss value in Table III.

TABLE II. ACCURACY OF OPTIMIZER, DATASET DISTRIBUTION, AND WORD2VEC MODEL

Optimizer	Epoch	*Six Classes*		*Four Classes*	
		CBOW	***Skip-Gram***	***CBOW***	***Skip-Gram***
Adam	100	70.44	73.65	71.78	74.01
	200	68.79	71.18	74.50	72.28
	300	66.26	71.67	71.78	75.25
SGD	100	56.40	46.55	67.33	51.98
	200	57.88	43.35	68.81	61.39
	300	70.20	57.64	71.53	63.86
AdaDelta	100	63.30	61.08	69.55	69.31
	200	63.79	67.40	73.27	65.59
	300	70.44	67.00	71.04	72.77

The experiments show the value of loss and accuracy of RNN learning outcomes on the results of CBOW and Skip-Gram word embedding using the Adam, SGD, and Adadelta optimization models on the test data gave the highest accuracy of 75% using the Adam optimization model with 300 epochs.

TABLE III. LOSS OF OPTIMIZER, DATASET DISTRIBUTION, AND WORD2VEC MODEL

Optimizer	Epoch	*Six Classes*		*Four Classes*	
		CBOW	***Skip-Gram***	***CBOW***	***Skip-Gram***
Adam	100	1.9245	1.4950	1.6796	1.5121
	200	2.0282	1.9106	1.7972	2.0036
	300	1.3577	1.5841	1.5529	1.7409
SGD	100	1.5342	1.4212	1.3044	1.1940
	200	1.2938	1.2545	1.6302	1.2697
	300	1.9002	1.7096	1.6943	1.3133
AdaDelta	100	1.1420	1.2091	1.6551	1.4850
	200	1.7166	1.6905	1.7864	1.8103
	300	1.8469	1.6529	1.9236	1.8287

This study examined the effect of epoch to see the consistency of the Adam, SGD, and Adadelta models in measuring semantic levels. When learning to use 200 epochs, there is an increase in the value of accuracy and loss. These results show that the time influences the convergence of optimization models. This study also examines the effect of word embedding in measuring semantic levels on the accuracy and loss values of the models built. The results in Table II show the Skip-Gram configuration is the best for this computing model. The Skip-Gram convergence model can be seen in Fig. 7 and Fig. 8.

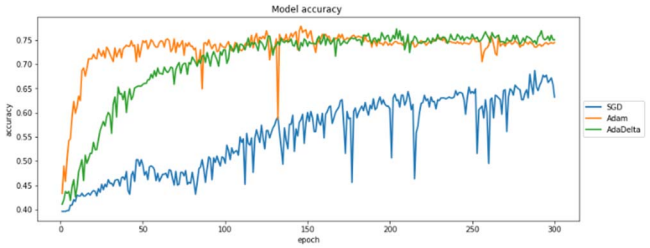

Fig. 7. Accuracy with Skip-Gram

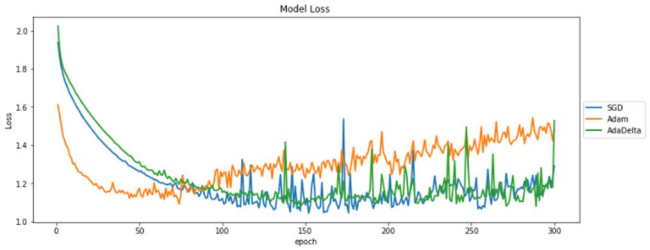

Fig. 8. Loss with Skip-Gram

As a result of word embedding testing in measuring semantic levels, CBOW and Skip-Gram have the best performance in pattern recognition in text. Confusion Matrix on the best CBOW and Skip-Gram models is shown in Table IV. The table shows that CBOW is less able to recognize patterns in words, so at the time of implementation, they are less precise in measuring semantic levels compared to Word Embedding Skip-Gram. Nevertheless, there is no significant difference between the CBOW and Skip-Gram models in the other classes. Three relations that have poor performance are OPPO, REL, and NOALI, where the composition in the corpus is the least.

TABLE IV. CONFUSION MATRIX OF SKIP-GRAM OPTIMIZER ADAM SIX CLASSES

A\P	OPPO	REL	NOALI	SPE	SIMI	EQUI
OPPO	**0**	0	0	1	0	0
REL	0	**0**	1	0	0	0
NOALI	0	0	**4**	1	3	1
SPE	0	0	0	**58**	13	7
SIMI	0	0	0	9	**99**	32
EQUI	0	0	0	15	24	**138**

Subsequent research was carried out by eliminating two classes that had the least composition in the corpus. This test is carried out to see how the accuracy of the two classes that have the least component of the corpus.

These results indicate that eliminating the two classes gives a high accuracy effect on the model built using SGD and AdaDelta optimization when compared with Table II testing. This study also shows the impact of word embedding in measuring semantic levels of accuracy and loss that are built from each model. The results in Table II illustrate the configuration for data that is not im-balance giving SGD optimization, and Adadelta can recognize patterns from any given text. The result is due to the ability of the model to be built to be able to study patterns with large amounts of data during training.

As a result of testing the word embedding in measuring the level of semantic configuration by eliminating the two classes, CBOW and Skip-Gram provide better performance compared to previous tests. Following this is the Confusion Matrix on the best Skip-Gram models.

TABLE V. CONFUSION MATRIX OF SKIP-GRAM OPTIMIZER ADAM FOUR CLASSES

A\P	NOALI	SPE	SIMI	EQUI
NOALI	**3**	0	0	4
SPE	0	**56**	17	17
SIMI	1	16	**109**	18
EQUI	1	10	16	**136**

In Table IV and Table V, we can see that reducing the number of classes does not have a significant effect on recognizing word patterns. The performance of built-in identifying sentence patterns is not too different from the tests in Table II. It was only given new knowledge that the data that is recognized will not be much different from the previous data even though by removing some data that is not im-balance.

In the Confusion Matrix, aspects of the scores are shown in Table IV. F1-Score explains that learning using RNN with Skip-Gram word embedding works better than using CBOW. The highest F1-Score results are contributed by class EQUI with 78%, class SIMI with 71%, class SPE with 72%, and class NOALI 52%. The worse classification is classes OPPO and REL according to the two smallest data classes on the corpus. Then the result of the F1-Score for the whole class was 46%.

Other tests examined F1-Score in the Confusion Matrix by removing the smallest data in the corpus shown in Table V. The biggest F1-Score is contributed by class EQUI, SIMI, and SPE.

These results indicate that eliminating two classes based on the data does not at least have a significant effect on the results obtained for the F1-Score of each class. The EQUI class F1-Score was obtained by 80%, the SIMI class 76%, the SPE class 65%, and the NOALI class 50%. The result of the overall F1-Score for the four classes is 67%. For accuracy testing, seen in 4 classes produces accuracy that is not too much different from the other optimizations. It is because the model can recognize features in the sentence with many variations in data compared with little data for each class.

IV. CONCLUSION

This research has shown how Word2vec works in representing words into vectors. Word2vec itself has CBOW and Skip-Gram variants that can be used to recognize meaning features in a word. When word representations become vectors, Window Size is configured to make the data very well recognized by the RNN pattern. RNN can conduct training to recognize patterns in scientific sentences so that these sentences can be measured semantically.

The results of this study found an influence of the Word Embedding word process on the Word2vec architecture. From the results of the study, it can be seen that the Skip-Gram architecture can recognize patterns in scientific sentences with Adam optimization model, which produces a good accuracy of 73%.

Other test results by eliminating two classes that have the least corpus composition using CBOW and Skip-Gram configurations with various optimizations give a better effect if using optimization SGD and Adadelta compared to the previously built model. Accuracy results did not have a significant impact by eliminating several classes. This result shows that the performance of built-in recognizing patterns in sentences is not too different from previous tests. So resulted obtained in the model built to provide better accuracy of 75% with the highest F1-Score of 67%.

REFERENCES

[1] S. Teufel, A. Siddharthan, and D. Tidhar, "An annotation scheme for citation function," *COLING/ACL 2006 - SIGdial06: 7th SIGdial Workshop on Discourse and Dialogue, Proceedings of the Workshop*, no. July, pp. 80–87, 2006.

[2] G. H. Rachman, M. L. Khodra, and D. H. Widyantoro, "Classification of citation sentence for filtering scientific references," *2019 4th International Conference on Information Technology, Information Systems and Electrical Engineering, ICITISEE 2019*, pp. 347–352, 2019.

[3] E. Agirre, A. Gonzalez-Agirre, I. Lopez-Gazpio, M. Maritxalar, G. Rigau, and L. Uria, "SemEval-2016 task 2: Interpretable semantic textual similarity," *SemEval 2016 - 10th International Workshop on Semantic Evaluation, Proceedings*, pp. 512–524, 2016.

[4] [R. C. Rajagukguk and M. Leylia Khodra, "Interpretable Semantic Textual Similarity for Indonesian Sentence," *ICAICTA 2018 - 5th International Conference on Advanced Informatics: Concepts Theory and Applications*, pp. 147–152, 2018.

[5] K. Krishnan, R. Krishnan, and A. Muthumari, "A semantic-based ontology mapping – information retrieval for mobile learning resources," *International Journal of Computers and Applications*, vol. 39, no. 3, pp. 169–178, 2017.

[6] S. Wan, Y. Lan, J. Guo, J. Xu, L. Pang, and X. Cheng, "A deep architecture for semantic matching with multiple positional sentence representations," *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, pp. 2835–2841, 2016.

[7] A. Abdi, N. Idris, R. M. Alguliyev, and R. M. Aliguliyev, "PDLK: Plagiarism detection using linguistic knowledge," *Expert Systems with Applications*, vol. 42, no. 22, pp. 8936–8946, Dec. 2015.

[8] T. Kano, S. Sakti, and S. Nakamura, "Neural Machine Translation with Acoustic Embedding," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019, pp. 578–584.

[9] A. Yousif, Z. Niu, J. Chambua, and Z. Y. Khan, "Multi-task learning model based on recurrent convolutional neural networks for citation sentiment and purpose classification," *Neurocomputing*, vol. 335, pp. 195–205, 2019.

[10] A. R. Pal and D. Saha, "An approach to automatic text summarization using WordNet," *Souvenir of the 2014 IEEE International Advance Computing Conference, IACC 2014*, pp. 1169–1173, 2014.

[11] M. Mohler, R. Bunescu, and R. Mihalcea, "Learning to grade short answer questions using semantic similarity measures and dependency graph alignments," *ACL-HLT 2011 - Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, pp. 752–762, 2011.

[12] A. A. Aziz, E. C. Djamal, and R. Ilyas, "Siamese Similarity between Two Sentences Using Manhattan's Recurrent Neural Networks," *Proceedings - 2019 International Conference on Advanced Informatics: Concepts, Theory, and Applications, ICAICTA 2019*, pp. 1–6, 2019.

[13] N. Grabar, "A French Corpus for Semantic Similarity," *Proceedings of The 12th Language Resources and Evaluation Conference*, no. May, pp. 6889–6894, 2020.

[14] H. Gonçalo Oliveira, A. Oliveira Alves, and R. Rodrigues, "Gradually Improving the Computation of Semantic Textual Similarity in Portuguese," vol. 10423, no. d, E. Oliveira, J. Gama, Z. Vale, and H. Lopes Cardoso, Eds. Cham: Springer International Publishing, 2017, pp. 841–854.

[15] Y. Xiong *et al.*, "Distributed representation and one-hot representation fusion with gated network for clinical semantic textual similarity," *BMC Medical Informatics and Decision Making*, vol. 20, no. Suppl 1, pp. 1–7, 2020.

[16] R. Antunes, J. F. Silva, and S. Matos, "Evaluating semantic textual similarity in clinical sentences using deep learning and sentence embeddings," *Proceedings of the ACM Symposium on Applied Computing*, pp. 662–669, 2020.

[17] E. Rejmund, W. Jaworski, and A. Wierzbicki, "Exploratory study of relationships among statement credibility, context, and semantic similarity," *Proceedings - 2014 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Workshops, WI-IAT 2014*, vol. 1, pp. 43–50, 2014.

[18] G. Majumder, P. Pakray, and D. E. P. Avendaño, "Interpretable Semantic Textual Similarity Using Lexical and Cosine Similarity," vol. 836, no. 2, J. K. Mandal and D. Sinha, Eds. Singapore: Springer Singapore, 2018, pp. 717–732.

[19] Y. Yang *et al.*, "Learning Semantic Textual Similarity from Conversations," in *Proceedings of The Third Workshop on Representation Learning for NLP*, 2018, pp. 164–174.

[20] M. Ahmed, C. Dixit, R. E. Mercer, A. Khan, M. R. Samee, and F. Urra, "Multilingual semantic textual similarity using multilingual word representations," *Proceedings - 14th IEEE International Conference on Semantic Computing, ICSC 2020*, pp. 194–198, 2020.

[21] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, pp. 1–12, 2013.

[22] T. Mikolov, W. T. Yih, and G. Zweig, "Linguistic regularities in continuous spaceword representations," *NAACL HLT 2013 - 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Main Conference*, pp. 746–751, 2013.

[23] X. Shi and R. Lu, "Attention-based bidirectional hierarchical LSTM networks for text semantic classification," *Proceedings - 10th International Conference on Information Technology in Medicine and Education, ITME 2019*, pp. 618–622, 2019.