Research Article

International Journal of Advanced Computer Research, Vol 8(39)
ISSN (Print): 2249-7277 ISSN (Online): 2277-7970
http://dx.doi.org/10.19101/IJACR.2018.839002

Word similarity score as augmented feature in sarcasm detection using deep learning

Joseph Tarigan* **and Abba Suganda Girsang**
Computer Science Department, BINUS Graduate Program-Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia

Received: 31-August-2018; Revised: 09-November-2018; Accepted: 14-November-2018

Abstract
Sarcasm detection is an important task in natural language processing (NLP). Sarcasm flips the polarity of a sentence and will affect the accuracy of sentiment analysis task. Recent researches incorporate machine learning and deep learning methods to detect sarcasm. Sarcasm can be detected by the occurrence of context disparity. This feature can be detected by observing the similarity score of each word in the sentence. Word embedding vector is used to calculate word similarity score. In this work, the word similarity score is incorporated as an augmented feature in the deep learning model. Three augmenting schemes in deep learning models are observed. Results show that in general, a word similarity score boosts the performance of the classifier. The accuracy of 85.625% with F-Measure of 84.884% was achieved at its best.

Keywords
Sarcasm detection, Word incongruity, Deep learning, Augmented feature.

1.Introduction
Sarcasm is an intended mockery that inverts the literal meaning of an utterance. The Oxford online dictionary defines sarcasm as *"The use of irony to mock or convey contempt"* [1]. Sarcasm transforms the sentiment of utterance into its opposite. It is used to harass or to express contempt to someone. Sarcasm detection is a difficult task. It must detect whether a positive utterance is having negative sentiment or vice versa.

Word is part of language that doesn't occur arbitrary. It has a relation with other words that occur together, for example, semantic relationship [2]. This relation is formed by the frequency of words that frequently appear together [3], thus, pairs of words used in general and normal situation will have high similarity scores.

Sarcasm can be detected through context incongruity [2, 4]. This feature can be detected by observing the similarity between the words in the sentence. Words that frequently appear together tend to be in the same context. Therefore, words in sarcasm text tend to have uneven distribution of word similarity score.

In this research, word similarity is introduced as a feature in sarcasm detection process. This feature is used as an augmented feature in the deep learning model. Deep learning has made impressive advances in computer vision field. Following the trend, many NLP researchers use deep learning as the classifier [5−7].

Traditional NLP techniques require features to be defined first, such as POS tags, term frequency, and TF-IDF. In contrast, deep learning will automatically detect patterns as features, enabling multilevel feature representation learning. The use of deep learning enables the classifier to find patterns that cannot be manually defined.

Word is a discrete unit of characters combination. Traditional NLP techniques use features extracted from the word, such as morphological feature, syntactic feature, and semantic feature. Those features are not numerical features; they cannot be directly used as the input in the deep learning model. There are several techniques to transform words into numerical features, such as co-occurrence matrix, but this method is inefficient and is too slow when the size of the corpus is getting larger [8]. Another technique that can be used is word embedding. Word embedding or distributed word representation [3] is a

*Author for correspondence

technique to transform the word into the vector of real number $\mathbb{R}^d$. The vector is formed using a neural network, trained by windowing the words, it captures the relationship between words that appear closer together. The relationship between words explains the normality of phrases.

1.1Background

The role of context incongruity in sarcasm processing has been studied in [9]. The effect of context incongruity on human interpretation and the processing of verbal irony by observing the participant's reaction and reading time have been discussed [9]. It concludes that literal statements are perceived to be more sarcasm when there is a disparity or contrast between contexts in a statement [9]. The greater the disparity or contrast between the context and the statement, the greater it led to be perceived as irony or sarcasm.

Words in similar contexts tend to have similar meanings [2] and are expected to appear closely together [10, 11]. In the context of NLP, words that appear in similar context can be estimated by word embedding technique. One of the most popular technique is Word2Vec [3]. This technique transforms words into the distributed vector representation and preserves syntactic and semantic word relationship. The general idea of this technique is to train a classifier model to classify context words given a target word by moving the windows through words as seen in *Figure 1*.

Figure 1 Illustration of sliding window in Word2Vec

The green window is the target word and the yellow windows are the context words. In this way, the disparity of context can be observed in the text by observing the similarity of each word in it.

Sarcasm can be detected in several ways. The most prominent way to detect sarcasm is by detecting context disparity [2]. Context disparity can be observed by evaluating the relation of words in a text. This relation can be observed by scoring each word for their similarity. Word similarity has been exploited as a feature to increase sarcasm detection performance in [4, 12]. It utilises similarity score as one of the features through the WordNet similarity score, which utilises *is-a* and *has-part* relation between words [4]. In [12], word similarity is used to detect context incongruity. They use the maximum and minimum score of the most similar and dissimilar word pair as the features of context incongruity. They also tried distance-weighted score of those features. Support vector machine (SVM) model was used in [12]. Augmented feature boosts at most 5% in the f-measure score.

Deep learning has been exploited as the classifier in a sarcasm detection task. In [5] CNN model has been used as the classifier. Several features are introduced to detect sarcasm, they are sentiment, emotion, and personality features. All features are concatenated together to form a final feature vector. They also incorporate SVM into the CNN model as the part of the classifier model. The model achieves impressive average accuracy of 95.27% when tested with 3 different datasets.

There are only a few researches of sarcasm detection conducted in the Indonesian language. [12] is the only paper we can find in the internet repositories. This mainly happens because the lack of dataset availability. In [12], sentiment analysis is done before the sarcasm detection takes part. The sarcasm detection task is done only on data with positive sentiment polarity.

1.2Contribution

In this research, we propose a sarcasm detection model that uses a common feature of sarcasm. The CNN deep learning model is used as the classification model and word similarity score is introduced as an augmented feature in the model. Several ways to augment the word similarity score is observed and the results are observed. This research is conducted in Indonesian Language.

2.Materials and methods

2.1Data materials

Twitter is a popular social media platform with more than 400 million active users [13]. It lets user posts message up to 140 characters length. Apart from

alphanumeric characters, user can use @ (at) to refer the tweet to another user, and # (hashtag) to identify the topic or to signify the context of the tweet. An example of a tweet is as follows: *"@username I watched the #tv and suddenly I want to eat #hungry"*.

Twitter API is used to collect tweet that contains hashtags. We relied on hashtags that twitter provides for users to use to signify their tweets. In our observation, users usually don't tweet sarcasm tweets and use #sarcasm tag in their tweets. Therefore, searching for sarcasm tweets by using only #sarcasm hashtag is not effective. We use other hashtags that often being used together in sarcasm tweets, such as #eh and #lah to obtain potential sarcasm tweets. Manual review is then performed on collected tweets by labelling the tweet as sarcasm or not sarcasm. These tweets then will be pre-processed. The pre-processing steps are described in section 2.5.

2.2Similarity as feature vector

Word2Vec is one of word embedding techniques [3]. Word embedding is a technique to compute distributed representation of words in the form of continuous vectors. Word vector from word embedding technique can capture semantic relationship between words. In this way, similarity between words can be estimated by calculating cosine similarity between word vectors. *Cosine similarity* is calculated using Equation 1.

$$cos(\theta) = \frac{A.B}{\|A\|\|B\|} \quad (1)$$

where A and B are vectors. The $cos(\theta)$ will be 0 if the angle between vector A and B is orthogonal, which means those vectors are linearly independent. The more $cos(\theta)$ close to 1, the more of both vectors are dependent or similar with each other.

Semantic similarity in word2vec is estimated by how closely words appear together and how many times these the words appear together. In normal situation, words that are occurring appear together form a sentence. This combination of words will have high similarity if the combination appears frequently. Similarity semantic similarity also concludes whether the words are considered to appear in the same context, as if words don't appear close together in normal situation, each of those words might have a different context of use.

Sarcasm is a way of mockery formed using words that don't occur appear together and closely in normal situation. In this way, if Word2Vec is trained with formal and daily conversation corpus such as Wikipedia or news corpus, some words in sarcasm text will have a low value in its word pair's similarity score calculation results shown in *Table 1*.

Table 1 Similarity score chart

	Saking	**Pintar**	**Tidak_Lulus**
Saking	1	0.278477	0.081239
Pintar	0.278477	1	0.191633
Tidak_Lulus	0.081239	0.191633	1

The pair of "Pintar" or "Smart with "Tidak_Lulus" or "Didn't Pass" has a low similarity score. This happens because these two words are rarely appearing together in normal context. In a normal context, "Smart" relates to "Pass the test". This value shows that there is a context incongruity between words in the tweet.

In this research, word similarity scores are used as feature vectors. All of the word similarity scores are concatenated together, as in (2), forming a vector. We call it similarity score vector.

$$x_{1-n} = \text{sim}(w_1, w_{1-n}) \oplus \ldots \oplus \text{sim}(w_n, w_{1-n}) \quad (2)$$

Where sim denotes similarity function. Words from pre-processed tweet are limited to 25 words. Similarity score vector will be padded with 0 for tweet that has been pre-processed with lower than 25 words.

To validate our intuition, a fully connected neural network with 256 hidden neurons is built and similarity score vector is used as the input, not as augmented feature. K-fold validation with k=5 is used as the validation method. The result is shown in *Table 2*.

Table 2 Performance of word similarity score used as input

	Percentage
Average Accuracy	80.080%
Recall	80.868%
Precision	78.917%
F-measure	79.828%

Similarity score as the input has been tested by Joshi

et al. [11]. The highest F-Measure they got was 69.49%, and it was concluded that their proposed similarity score cannot be used alone as the input. In our case, with f-measure of 79.828%, our proposed similarity score vector is sufficient to use as the input alone and validates that similarity score vector is a useful feature to use for sarcasm detection.

2.3Deep learning setup

Deep learning is defined as a model consists of many layers that can learn complicated concepts [14]. In this research, the CNN deep learning model is used. Each convolutional layer consists of convolution operation and pooling operation. *Max-pooling* is used in pooling layer, while *Leaky ReLU* function is used in convolutional layer. *Sigmoid function* is used in the output layer, yielding a binary output for the classification task. The formula can be seen in Equation 3,4 and 5 respectively.

$$f(X) = \arg max\,(X) \quad (3)$$

$$f(x) = \begin{cases} x, & x > 0 \\ ax, & x < 0 \end{cases} \quad (4)$$

$$f(x) = \frac{1}{1+e^{-x}} \quad (5)$$

Word is a combination of alphanumeric characters. Traditional NLP techniques usually represent words as indices of vocabulary. Words need to be converted into a representation that deep learning model can use as its input, which is a vector of real numbers $\mathbb{R}^d$. This task is accomplished by using the word embedding technique. In this research, words are transformed into vectors using Word2Vec. Word2Vec model is trained on a *Wikipedia* Bahasa Indonesia corpus. Each word will be converted into a vector with 300 dimensions. Each of the resulting word vectors is concatenated together, forming a vector. A tweet having *n* words is represented as follows in (6):

$$x_{1-n} = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \ldots \oplus x_n \quad (6)$$

As the *n* is limited to 25 words, pre-processed tweets that have words lower than 25 words will be padded with 0.

The architecture of the deep learning model used in this research can be seen in *Table 3*.

Table 3 Architecture of CNN model

	Neuron count	Kernel size	Kernel count	Padding	Stride
Input Layer	300×25	-	-	-	-
1st Conv	-	11	300	VALID	1
Pooling Layer	-	2	-	VALID	2
2nd Conv	-	11	200	VALID	1
Pooling Layer	-	2	-	VALID	2
3rd Conv	-	10	100	VALID	1
Pooling Layer	-	2	-	VALID	2
4th Conv	-	10	100	VALID	1
Pooling Layer	-	2	-	VALID	2
Dense Layer	100	-	-	-	-
Output Layer	1	-	-	-	-

2.4Feature augmentation

Augmentation in the machine learning model has been widely used in computer vision and speech recognition [15−17]. Augmentation can be applied to the dataset or in the feature space. Augmentation on dataset is done by creating a new instance of data by transforming the original data or by adding noise to the original data. Researches refer this method as data augmentation [17]. This method is used in a condition where the required labelled dataset is not available in large quantity. Synthetic data is formed with features that are not so different from the original dataset. Augmentation in the feature space is done in the feature space produced by the classifier model, for example, the deep feature of CNN [18]. With this scheme, external features can be incorporated with the main classifier, increasing the ability of the model to classify better.

In this paper, similarity score vector is used as augmented feature. At the forward pass, Word2Vec model takes in one-hot vector of each word. The representational vector of each word is used as the distributed representation of word and is used as the input by the CNN. Those vectors are then used to calculate the similarity of each word, concatenated together, forming the similarity score vector. Augmenting the similarity score vector is the original aspect of this research. The similarity score vector will be augmented in the feature space. There are several possible ways of augmenting the similarity score vector in the feature space. The optimality of

the feature augmentation methods is unknown in this context. Through this research, the most optimal methods of augmenting the similarity score vector will be obtained. Three ways of augmenting similarity vector with deep learning model are observed. They are described in the following sub-sections.

2.4.1 Word similarity score vector augmented with the deep feature

In this scheme, similarity score vector is augmented through concatenating the vector with deep feature vector, as seen in *Figure 2*. Deep feature is the flatten output vector of the last convolution layer in the deep learning model, before the fully connected layer. This method is commonly used by researchers to incorporate features from more than one model, as seen in [15, 18]. The concatenated vector will be the input of the fully connected layer. After the augmentation, the dimension of the deep feature will be 256 + (25×25).

2.4.2 Word similarity score vector augmented as a channel in the input layer

In computer vision task, deep learning model takes colour image with more than one channel as the input. The image will commonly have 3 channels that consist of red, green, and blue channels. In NLP tasks, deep learning usually will only take one channel of input, which is the sparse d dimension encoding of words (d is the number of concatenated words dimension). In this case, d will be the word embedding vectors. In this scheme, similarity score vector is embedded as the second channel in the input layer. The input layer is now having 2 channels, as seen in *Figure 3*. It consists of word embedding vector as the first layer and the similarity score vector as the second channel.

The second channel needs to have the same dimension with the first channel. Linear interpolation is used to resize the second channel. The similarity score vector will be processed together, in parallel, with the word embedding vector by the convolutional layers.

2.4.3 Word similarity score vector augmented with the input vector

In this scheme, similarity score vector is concatenated with the input vector, which is a word embedding vector, as seen in *Figure 4*. The input vector will have dimensions of (300×25) + (25×25).

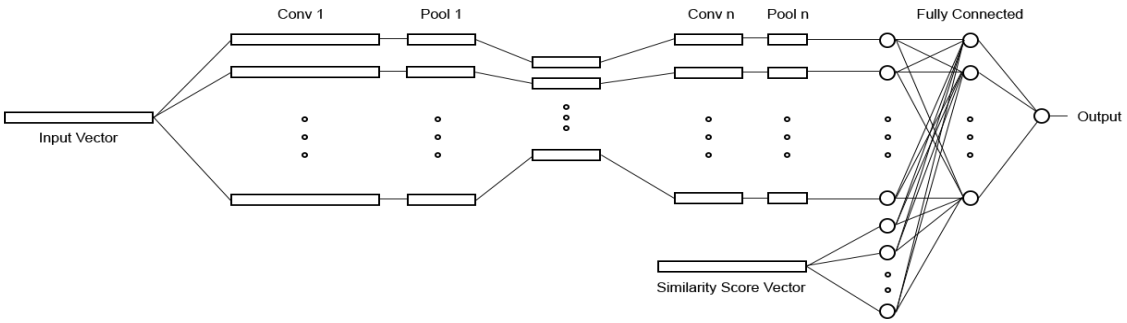

Figure 2 Illustration of similarity score vector augmented with the deep feature

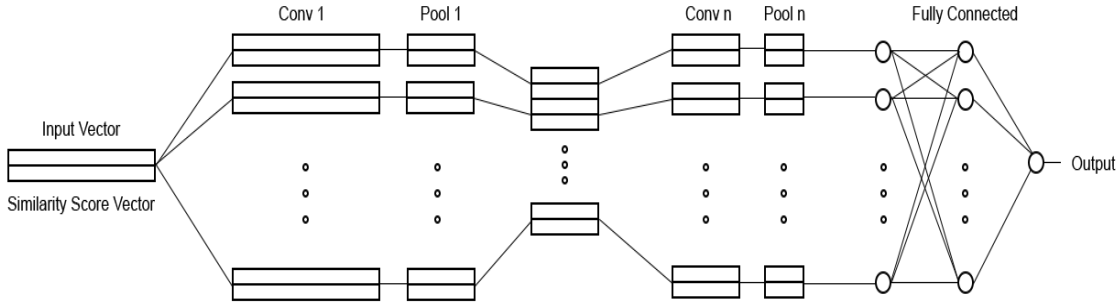

Figure 3 Illustration of similarity score vector augmented as the second channel

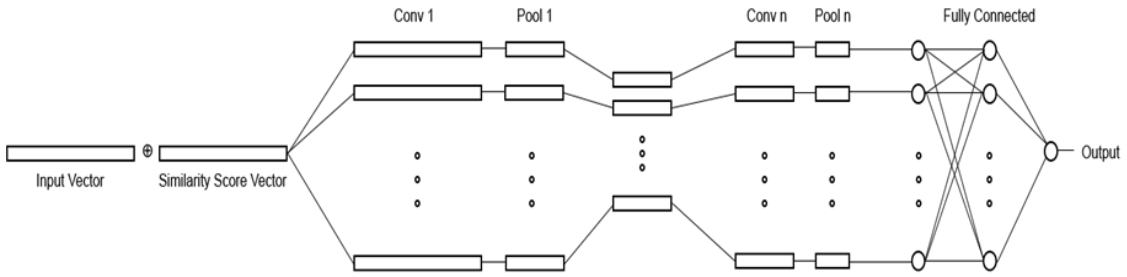

Figure 4 Illustration of similarity score vector augmented with the input vector

2.5Data pre-processing

Tweets collected from twitter are noisy and need to be cleaned [19]. Users usually don't use standard language in twitter. Constrained by character limitation, twitter users are usually abbreviating word such as "b4", "brb", and "lol". Smiley is popular too among the twitter users. Its short and expressive nature makes it popular in character count limited social media like the twitter. These kinds of paradigm make the processing becomes ineffective. They need to be normalised first.

Pre-processing takes the role in this phase. It will normalise the tweet, so it can be processed by the classifier. There are several processes in this step; each of them will be described as follows.

RT Filter: Retweeted tweets are considered not useful, as they have the same content with the original tweets. Retweeted tweets can be detected by character "RT" at the beginning of the tweets. We remove tweets that started with "RT" characters.

URL Removal: User can type anything in their tweet, including URL. URL is considered not useful, as it doesn't convey any meaningful feature unless we follow, open the URL, and process the information from the referred URL. URL is discarded from the tweet by using regular expression to detect the URL.

Twitter Special Tags Removal: Twitter uses some tags as special tags, such as "@" to mention other user, and "#" to signify or to show the context of tweet. These characters need to be eliminated as they are considered not a feature.

Tokenization: Tokenization is a process of chunking words into a collection of single word. In this process, some words are filtered and tokenised as a concatenate bigram. Words such as "Tidak suka" has the opposite meaning with "Suka". It will be tokenised as "tidak_suka", so it will not lose its negation feature.

Abbreviation Expansion: As described before, users in twitter tend to abbreviate word. Those words need to be expanded before they are used as input for the classification task. Abbreviations are expanded by using dictionary we build from our observation on the collected tweets.

Duplicated Letter Normalization: Twitter users usually exaggerate message by duplicating letters in the word, such as 'woooow' and 'waahhh'. 'Waw' and 'waaaaw' words are not standard. "Waw" and "waaawww" have the same meaning and intention, they should be treated in the same way. These kinds of words need to be normalised, so the model will process them as the same words. These kinds of words are normalised by removing duplicated consecutive letters in the word. In this way, word such as "waaaww" will be transformed to "waw" and will be treated as the same word.

Reversed Word Normalization: Slang words are used extensively in twitter. Reversing letter order in word is popular in Indonesia. For example, "Yuk" means "Come on" in English, and twitter users sometime reverse the letter order to "Kuy". These kinds of word need to be normalised. Normalising is accomplished by using dictionary we built from our observation on collected tweets.

Reduplicated Word Normalization: Word like "hahahaha" will be normalised by removing duplicated word and leave only 2 repetitions. In this way, "hahahahaha" will be transformed to "haha". Reduplicated words will be treated in the same way.

Unused Punctuation Removal: Punctuation is considered as feature in Natural Language Processing. Punctuation can be used to intensify the semantic meaning, such as "!" and "?". There are some punctuations that are not used or don't have contribution in forming word's semantic meaning. In this step, unused punctuations are eliminated, such as braces, ";", "*", and some more unused punctuations.

Stemming: In this step, words are normalised into their dictionary form. Modified enhanced confix stripping [20] is used as stemming algorithm.
Stopword Removal: Stopword is considered unimportant in text mining [21]. Therefore, it is usual to remove stopwords from corpus before further analysis. Stopword dictionary developed by Tala [22] is used in this step.

The output of these steps above is a collection of classification-ready word tokens. These tokens are transformed into vectors with Word2Vec. These vectors are concatenated together, forming a classification-ready vector.

2.6Experimental setup

Accuracy, precision, recall, and f-measure are used as the metric and 5-folds cross validation is used for evaluating the models. The vanilla CNN model has the same architecture as described in *Table 3*. SVM model is built with RBF kernel and penalty term or *C* value is 1. Vanilla CNN and SVM models use the same dataset, but with no augmented feature. The models will do binary classification on the dataset. The dataset consists of 4000 tweet samples with 2 labels, sarcasm and not sarcasm. The polarity distribution in the dataset is balanced, means the dataset consists of 2000 sarcasm samples (positive) and 2000 non-sarcasm samples (negative). Training runs for 50 epochs in each CNN model. For the SVM model, training runs with the stopping criterion of 10^{-4}.

3.Experimental results

The results of the experiments are shown in this section. To measure the potential of our method, we consider the method proposed by Joshi et al. [11] as our baseline. In addition, result from Lunando and Purwarianti [12] has also included a comparison of sarcasm detection in Indonesian language. Model 1, 2, and 3 are CNN models with augmented similarity score vector, described in section 2.4 respectively. For comparison purpose, vanilla CNN model and SVM model are also built. The vanilla CNN model is shown as model 4, SVM with no augmented similarity score vector is shown as model 4, SVM without augmented similarity score vector is shown as model 5, SVM with augmented feature is shown as model 6, SVM model by Joshi et al. [11] is shown as model 7, and result from Lunando and Purwarianti [12] is shown as model 8. The detail of each model is shown in *Table 4*.

Table 4 Table of model description

Model	Model description
1	CNN with word similarity score vector augmented with the deep feature.
2	CNN with word similarity score vector augmented as a channel in the input layer.
3	CNN with word similarity score vector augmented with the input vector
4	Vanilla CNN without augmented feature.
5	SVM without augmented feature.
6	SVM with augmented feature
7	SVM by Joshi et al. [11]
8	Lunando and Purwarianti [12]

3.1Accuracy results

All models are trained and tested with setup as described in section 2.6. The classification accuracy results are presented in *Table 5*.

Results show that in general, deep learning CNN models with augmented similarity score vector have better accuracy than the vanilla CNN model, though it only gains 2% accuracy from the vanilla CNN model. The vanilla CNN model achieves 81.950% average accuracy, while model 1 achieves 83.950% average accuracy.

The interesting thing is that the SVM models in our experiment outperform the accuracy of vanilla CNN by 0.825%. In our experiments, we concluded that this happens because of the dataset size and the structure of the CNN models. The SVM constructs decision boundary between the data, it relies on support vectors that divides the label on the data in the dataset. Dataset with a small number of labels such as a binary labelled dataset is assumed to have linearly separated data, thus, SVM perform better than deep learning methods. SVM will underperform deep learning models when the data in the dataset is scattered and has many labels. This can be observed by the accuracy of deep learning models and SVM model between binary labelled and 3 class labelled dataset in [23], the accuracy gap is getting smaller as the dataset has more labels. The word similarity vectors clearly boost the accuracy of CNN model. Compared with vanilla CNN model, augmenting

word similarity vectors into the CNN model boost the accuracy by 2% at its best. The accuracy of CNN models with augmented feature is also better than the accuracy of SVM model. For better understanding, other metrics are calculated; they are shown in the next section.

Table 5 Classification accuracy of each model

Model	Min	Max	Avg.	Std. Dev.
1	82.375%	85.625%	**83.950%**	1.171%
2	79.250%	85.625%	82.775%	2.665%
3	82.38%	85.13%	83.78%	1.04%
4	79.625%	83.250%	81.950%	1.449%
5	79.250%	85.625%	82.775%	2.665%
6	79.950%	86.840%	83.210%	2.205%
7	79.250%	85.625%	82.775%	2.665%
8	-	-	54.1%	-

3.2Precision, recall, and F-measure

Precision, recall, and f-measure of each model as described in section 2.6 are shown in *Table 6*.

Vanilla CNN achieves an impressive 93.250% recall score, but it also has the minimum precision score among other CNN models, which affect the usability of the model. Despite the SVM models achieved good average accuracy as shown *Table 5*, they have the lowest precision scores.

Table 6 Precision, recall, and f-measure of each model

Model	Precision	Recall	F-Measure
1	81.299%	88.800%	**84.884%**
2	78.19%	91.60%	84.37%
3	79.732%	90.650%	84.841%
4	76.187%	93.250%	83.859%
5	64.897%	90.176%	75.461%
6	67.082%	89.500%	76.684%
7	65.784%	90.500%	76.179%
8	-	-	-

4. Discussion

Recent researches in NLP show that deep learning methods are performing better than machine learning models in NLP tasks. Researchers in [6] explored deep learning model in twitter sentiment analysis task. They use a CNN model to analyse tweet sentiment using Semeval-15 datasets. The model achieves accuracy that could rank in the first two positions in Semeval-15 tasks.

In the context of using word similarity score as the feature of sarcasm detection task, experiments have been conducted in [12]. In [12], similarity score is used as augmented features by calculating four scores, the maximum score of most similar and dissimilar word pair and the minimum score of similar and dissimilar score. They also add weights to the scores by measuring the distance between each word pair. SVM model was used in [12]. Results show that augmenting these features boost the performance of classification models, although it decreases the performance in some of the experimental models. The analysis shows that contextual sarcasm can't be captured with features used in [12] as they only use score of similar and dissimilar word pair. This problem is solved by using the whole word similarity score as the feature, as it contains all similarity scores of word pair in the phrase. By incorporating not only the most similar and dissimilar word similarity score, it captures the complete context of the conversation.

In our experiment, SVM model scores better than vanilla CNN. This occurrence also happens in [23]. The performance of deep learning methods and machine learning methods in the twitter sentiment analysis are evaluated in [23]. They used CNN and Elman RNN as deep learning models and SVM model as machine learning model. The SVM model uses unigram, bigram, and a combination of them as term frequency features, while word vector representation is used as the feature of the deep learning models. The models are evaluated with binary and 3 class classification tasks. All of the

SVM model variants perform better than deep learning model variants in the classification task. The average performance gap between SVM and deep learning models is 11.65% of binary classification tasks. The average performance gap is getting smaller in 3 class classification tasks, it decreases to 8.52%. We conclude that this happens because of the size of the dataset and the number of dataset's labels as described in section 3.1.

Model 2 and 3 have the augmented feature augmented with input vector, while the model 1 has it augmented with the deep feature vector. The dimension of augmented feature is 625 and is a ready-to-be classified feature, as we have concluded in section 2.2. The augmented feature in model 2 and 3 is sampled down by the convolution process in CNN model, thus, diminish the similarity score information. While the convolution process will extract important features from given input, the results show that the similarity score vector is better when augmented with the deep feature rather than with the input vector. Given the finding in section 2.2, this concludes that the similarity score vector is better when it is classified directly with a fully connected neural network, as in the model 1. We want to investigate more on how to utilise the similarity score feature in future research.

5.Conclusion

By experimenting different scheme of augmentation, we have 2 conclusions. First, augmenting word similarity score as a whole with deep learning model gives the model additional knowledge to classify sarcasm better, thus, increasing its accuracy. Second, augmenting similarity score vector with the deep features in CNN gives the best result. This scheme also creates a model with balanced precision and recall over other augmenting schemes. In the future, we plan to evaluate RNN deep learning model augmented with word similarity score. We also want to gather more Indonesian language data and publish it online. Future work will also focus on how to extract and utilise word similarity associated with other features, such as word intensifier, punctuations, and deep learning hyper parameter optimization.

Acknowledgment

We thank you Binus University AI Research Center for the computing facility.

Conflicts of interest

The authors have no conflicts of interest to declare.

References

[1] Oxford Online Dictionaries. https://en.oxforddictionaries.com/definition/sarcasm. Accessed 9 October 2018.
[2] Harris ZS. Distributional structure. Word. 1954; 10(2-3):146-62.
[3] Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. In advances in neural information processing systems 2013 (pp. 3111-9).
[4] Joshi A, Sharma V, Bhattacharyya P. Harnessing context incongruity for sarcasm detection. In proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing 2015 (pp. 757-62). Association for Computational Linguistics.
[5] Poria S, Cambria E, Hazarika D, Vij P. A deeper look into sarcastic tweets using deep convolutional neural networks. In proceedings of international conference on computational linguistics 2016 (pp. 1601-12). The COLING.
[6] Severyn A, Moschitti A. Twitter sentiment analysis with deep convolutional neural networks. In proceedings of the international ACM SIGIR conference on research and development in information retrieval 2015 (pp. 959-62). ACM.
[7] Majumder N, Poria S, Gelbukh A, Cambria E. Deep learning-based document modeling for personality detection from text. IEEE Intelligent Systems. 2017; 32(2):74-9.
[8] Lin J. Scalable language processing algorithms for the masses: a case study in computing word co-occurrence matrices with MapReduce. In proceedings of the conference on empirical methods in natural language processing 2008 (pp. 419-28). Association for Computational Linguistics.
[9] Ivanko SL, Pexman PM. Context incongruity and irony processing. Discourse Processes. 2003; 35(3):241-79.
[10] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781. 2013.
[11] Joshi A, Tripathi V, Patel K, Bhattacharyya P, Carman M. Are word embedding-based features useful for sarcasm detection?. arXiv preprint arXiv:1610.00883. 2016.
[12] Lunando E, Purwarianti A. Indonesian social media sentiment analysis with sarcasm detection. In international conference on advanced computer science and information systems 2013 (pp. 195-8). IEEE.
[13] Allcott H, Gentzkow M. Social media and fake news in the 2016 election. Journal of Economic Perspectives. 2017; 31(2):211-36.
[14] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015; 521:436-44.
[15] Liu B, Wang X, Dixit M, Kwitt R, Vasconcelos N. Feature space transfer for data augmentation. In

proceedings of the conference on computer vision and pattern recognition 2018 (pp. 9090-8). IEEE.
[16] Volpi R, Morerio P, Savarese S, Murino V. Adversarial feature augmentation for unsupervised domain adaptation. In proceedings of the conference on computer vision and pattern recognition 2018 (pp. 5495-504). IEEE.
[17] Salamon J, Bello JP. Deep convolutional neural networks and data augmentation for environmental sound classification. IEEE Signal Processing Letters. 2017; 24(3):279-83.
[18] Valkonen M, Kartasalo K, Liimatainen K, Nykter M, Latonen L, Ruusuvuori P. Dual structured convolutional neural network with feature augmentation for quantitative characterization of tissue histology. In proceedings of the conference on computer vision and pattern recognition 2017(pp. 27-35). IEEE.
[19] Han B, Baldwin T. Lexical normalisation of short text messages: Makn sens a# twitter. In proceedings of the annual meeting of the association for computational linguistics: human language technologies (pp. 368-78). Association for Computational Linguistics.
[20] Tahitoe AD, Purwitasari D. Implementation of modified enhanced confix stripping stemmer for Indonesian language using corpus based stemming method. Institut Teknologi Sepuluh (ITS). 2010.
[21] Meyer D, Hornik K, Feinerer I. Text mining infrastructure in R. Journal of Statistical Software. 2008; 25(5):1-54.
[22] Tala FZ. A study of stemming effects on information retrieval in bahasa Indonesia. Institute for Logic, Language and Computation, Universiteit van Amsterdam, The Netherlands. 2003.
[23] Lu Y, Sakamoto K, Shibuki H, Mori T. Are deep learning methods better for twitter sentiment analysis?. In proceedings of the 23rd annual meeting of natural language processing (Japan) (pp. 787-90).

Joseph Tarigan was born on June 11th 1990. He completed his undergraduate study at Binus University, Jakarta, Indonesia, majoring Computer Science and continued Master Degree in Information Engineering at Binus University, Jakarta Indonesia. He is currently a Software Engineer in Jakarta, Indonesia. His research interests are Artificial Neural Network, Computer Vision, and Information Engineering.
Email: joseph.tarigan@binus.ac.id

Abba Suganda Girsang is currently a lecturer at Master in Computer Science, Bina Nusantara University, Jakarta, Indonesia Since 2015. He got Ph.D. degree in 2015 at the Institute of Computer and Communication Engineering, Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan. He graduated bachelor from the Department of Electrical Engineering, Gadjah Mada University (UGM), Yogyakarta, Indonesia, in 2000. He then continued his masters degree in the Department of Computer Science at the same university in 2006–2008. He was a staff consultant programmer in Bethesda Hospital, Yogyakarta, in 2001 and also worked as a web developer in 2002–2003. He then joined the faculty of the Department of Informatics Engineering in Janabadra University as a lecturer in 2003-2015. His research interests include Swarm, Intelligence, Combinatorial Optimization, and Decision Support System.
Email: agirsang@binus.edu