

Dual embedding with input embedding and output embedding for better word representation

Yoonjoo Ahn¹, Eugene Rhee², Jihoon Lee¹

¹Department of Smart Information and Telecommunication engineering, Sangmyung University, Cheonan, South Korea

²Department of Electronics engineering, Sangmyung University, Cheonan, South Korea

Article Info

Article history:

Received Sep 29, 2021

Revised May 24, 2022

Accepted Jun 10, 2022

Keywords:

Dual embedding

Natural language processing

Word embedding

Word representation

Word2Vec

ABSTRACT

Recent studies in distributed vector representations for words have variety of ways to represent words. We propose a various ways using input embedding and output embedding to better represent words than single model. We compared the performance in terms of word analogy and word similarity with each input and output embeddings and various dual embeddings which are the combination of those two embeddings. Performance evaluation results show that the proposed dual embeddings outperform each single embedding, especially with the way of simply adding input and output embeddings. We figured out two things in this paper, i) not only input embedding but also output embedding has such meaning to represent the words and ii) combining input embedding and output embedding as dual embedding outperforms the single embedding when we use input embedding and output embedding individually.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Jihoon Lee

Department of Smart Information and Telecommunication Engineering, Sangmyung University

Cheonan, South Korea

Email: vincent@smu.ac.kr

1. INTRODUCTION

A word embedding is a way to represent words using a dense vector representation. It is an improvement over more traditional bag-of-words model where they used large sparse vectors to represent each word in entire vocabulary. Since the size of the vocabulary was vast, these representations had to be sparse. So the given word or documentation would be represented with sparse vectors comprising mostly with zero values. However, in an embedding, words are expressed by dense vectors, in which a vector means projecting the word into a continuous vector space.

There has been a surge of work that propose word embedding using diverse training schemes based on neural-network language modeling like [1]-[3]. Distributed vector representations of words can capture meanings of the word. Word embedding, in other words, is crucial in learning algorithms to get higher improvement in natural language processing tasks like [4], [5]. There were various approaches to represent the word by distributed vector, we propose a new approach to make a distributed vector representation. In the Word2Vec model (Continuous Bag-of-Words (CBOW), Skip-gram) in [6], it outputs a feature matrix of words. While training, there are 2 matrices which is created between input layer and output layer. In several previous works, it has already been proven that output vector can acts as a word embedding and performs almost as good as an input vector. Note that we can call the input vector as input embedding and the output vector as output embedding. What we are going to utilize in this paper is input embedding and output embedding from the Word2Vec model. We get the input embedding matrix and output embedding matrix after training the words to have distributed vector representations. We propose a better embedding by combining input

embedding and output embedding in various ways. It outperforms the embedding that use only the input embedding as the Word2Vec presents. We know that there are many works that represent words almost perfectly with pre-trained vectors like [7]-[9], but the main part of the proposed scheme is that the simplest way is utilized to represent words using the basic Word2Vec model. Thus comparing with the basic model, this method's performances are remarkable. It may not be the state-of-the-arts performance in making word embeddings, but we are presenting various ways of utilizing input and output embeddings.

Input embedding and output embedding can both serve as word embedding. We use both of these embeddings to derive richer distributional relationships. It has been shown that combining embeddings results a better word embedding than using it individually. Different from other papers, we simply use only the embeddings from Word2Vec model, while they use other embeddings from the other models. In this paper, we tried various ways to combine input embedding and output embedding to better word embedding that represents words well. We compare the quality of each individual embeddings, input and output, and the combination of those embeddings by word analogy task, word similarity task and comparing nearest neighbors to see which method of combination performs better.

The main parts of this paper are as shown in:

- We propose various and efficient ways to represent the words better using input embedding and output embedding from Word2Vec model.
- We compare the performance of input embedding and output embedding with each of dual embeddings in various evaluation methods like word analogy task, word similarity task and nearest neighbors.
- Our idea of dual embedding is the simplest way of representing words comparing with recent works.

We will explain how our idea of dual embedding came from in section 2 with related works for this paper. And in section 3, we will talk about our dual embedding models one by one. Then we will use our various embeddings came from dual embedding models to evaluate and compare the embedding's quality with input embedding and output embedding in section 4. So the next section 5 will be the conclusion of this paper.

2. RELATED WORKS

2.1. Word representations

The Word2Vec model was first introduced by Sonkar *et al.* [6] to learn high-quality word representations from large data with billions of words. Their models are effective at capturing semantics and syntactics of the words measured in a word analogy task, which is useful for various natural language processing tasks. There were some trials to make Word2Vec a better model with various training methodologies like casting the Skip-gram with negative sampling (SGNS)'s training scheme as weighted matrix factorization [10]. Meanwhile, some works explained the Word2Vec model's negative sampling in details [11] and about parameter learning in details [12]. Hambi and Benabbou [12] mentioned about the "input vector" and "output vector" that comes from the Word2Vec model while training.

2.2. Awareness of the output embedding

There were some attempts to use both this input vector and output vector in [13], [14] to find out the usefulness of the output vector. Li and Summers-Stay [13] observed that output vector in a Word2Vec model can also be useful. They retrained both the embedding spaces to obtain more distributional relationships. They said Word2Vec model contains two separate embedding spaces(input and output) whose interactions capture additional meanings of words that cannot be found in each embeddings [15]. So they combined embeddings to leverage both the embeddings spaces and they used it for query and document ranking.

Similar to that, Nalisnick *et al.* [14] tried to improve the model for better improvement for information retrieval (IR). They said that for certain IR tasks, they postulate that they should combinedly use both the IN and the OUT embeddings. The meaning of dual embedding with input embedding and output embedding by [13] and [14] is that they mapped query words into the input domain and the document words into the output domain.

According to Press and Wolf [16], with the Word2Vec Skip-gram model, the quality of output embedding is almost as good as the quality of input embedding tested on five embedding evaluation methods. They suggested the tied model with input and output embeddings which leads to an improvement in the perplexity of various language models. While they use two embeddings for their papers, there were some several works that worked on utilizing input embedding and output embedding by demonstrating the effectiveness of the output embedding.

2.3. Combining embeddings

There were some methods that help to combine embedding vectors. Garten *et al.* [17] tried to combine vectors generated from different models such as distributed vector representation in sigma (DVRS) [18] and

Word2Vec. Tsuboi [19] showed the way to combine Word2Vec and GloVe embeddings into a part-of-speech (POS) tagging task. They demonstrated that using these two embedding sets together is beneficial than using them individually by improving the tagging accuracy. Also Jiao and Zhang [5] starts from the motivation that semi-supervised approaches can improve accuracy. For that, they combined two public embeddings, circular watermarking (CW) embedding [2] and hierarchical log-bilinear (HLBL) embedding [3], to show better performance than using these embeddings individually. A multi-view word embedding scheme using two-sided neural network was proposed [20]. They tried to make several embeddings by training CBOW model on various datasets like Wikipedia corpus, search click-through data and user query data. They combined these embeddings trained on different datasets and showed that using these embeddings together gives stronger results than using them individually.

Goiknetxea *et al.* [21] used concatenation of the word embeddings trained from different corpus and WordNet and improved the performance. Yin and Schütze [22] proposed various methods of combining five different public embedding sets like Word2Vec [6], [23]-[28], GloVe [29], and CW [2], HLBL [3], and Huang *et al.* [30]. They introduced concatenation (CONC), all known words (AVG), singular value decomposition (SVD), and 1ToN to combine these three embeddings to better represent the words. And similarly, Coates and Bollegala [31] introduced autoencoder method to combine those public embeddings. These previous works showed combining embeddings performs better than using one embedding alone.

3. PROPOSED METHOD

Word2Vec model introduced by Sonkar *et al.* [6] is a neural network-based technique which is based on distributional hypothesis that learns word embedding from the context words. The model comes from the situation that words in similar contexts hold similar meanings. The Word2Vec learns word representations through skip-gram model and continuous bag-of-words (CBOW) model. Continuous bag-of-words (CBOW) model is trained by predicting the target word based on the context words. This learns a word's embedding through maximizing the log probability of the word from the context words in the window. The Skip-gram model is similar, but completely opposite to the CBOW model, it predicts the context words founded on the target word. It learns word embedding for each word both in an input embedding matrix and in an output embedding matrix. There are two matrix in the model, first weight matrix is the one that is between an input layer and a hidden layer. In Figure 1, W_{in} is the input weight matrix of $V \times N$. Note that V is the vocabulary size of the embedding and N is the hidden layer size. W_{in} is the weight matrix that is returned as a word embedding. And the second weight matrix is generated in the middle of the output layer and the hidden layer. This is the output matrix W_{out} of $N \times V$ in Figure 1. We update these two matrices when we train context words and target words. Normally, the input weight matrix W_{in} is the returned vector to use as a word embedding of Word2Vec while output weight matrix W_{out} is abandoned. It means, by default, Word2Vec discards output embedding after training, and then outputs only the input embedding. However, in this paper, we used both the input weight matrix and the output weight matrix to better represent the word. Note that we call input weight matrix W_{in} as input embedding emb_{in} and output weight matrix W_{out} as output embedding emb_{out} .

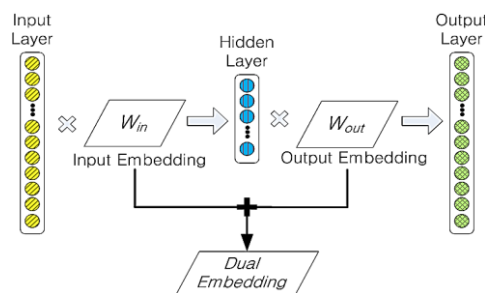


Figure 1. Learning process of the Word2Vec model

3.1. Concatenation

We simply came up with an idea of concatenation and sum to combine two embedding vectors into one embedding vector. We tried to combine input embedding and output embedding to capture both of their features. Actually the method of concatenating embeddings was used in [22] where they concatenated five public embeddings. They found out that concatenation of the embeddings is effective method for a particular word. We did it similarly, but the only difference is that we concatenated only two embedding vectors from

one model. In other words, we concatenated input embedding and output embedding from the Word2Vec like (1). And then we did L2 normalization for the emb_{CONC} . Combining those two embeddings into one vector, results the dimension size to be the double of each embeddings dimension. This causes a increase in training parameters than having an original dimension. We used this embedding as CONC embedding.

$$emb_{CONC} = emb_{in} \oplus emb_{out} \quad (1)$$

3.2. Sum

The sum embedding is the result of adding the input embedding and the output embedding element-wise like (2). The dimension size of this embedding is the same as the input embedding and output embedding. Bao and Bollegala [32] proposed the method of averaging the embeddings to combine in one vector. They proved that if word embeddings are shown to be approximately orthogonal, then, without increasing the dimensionality, averaging the embeddings will have the same information as concatenation. But in this paper, we tried both ways, averaging the embeddings like [32] and just adding the embeddings, not dividing into 2. Then compared the results of those embeddings, just adding two embeddings without dividing into 2 performed well. So we used this embedding as SUM embedding (emb_{sum}) in further experiments.

$$emb_{sum} = emb_{in} + emb_{out} \quad (2)$$

3.3. Auto encoder

Auto Encoder is an unsupervised way of finding data features only from the data input. This method was introduced in [31] which combines other word embeddings, e.g. Word2Vec and GloVe. However, in this paper, we use only input embedding and output embedding from Word2Vec.

We used the result of CONC embedding, i.e. concatenation of the input embedding and output embedding, as our input to the autoencoder. The goal is to make the reconstructed matrix in the output layer similar with the input layer's original matrix by minimizing the total reconstruction error. While we trained the concatenation embedding in an autoencoder, we randomly initialize the matrix at first, and we did not use any activation functions. As we train this model with (3), the matrix in the hidden layer learns from the input layer, which is the concatenation of the input embedding and output embedding.

$$loss = \sum \| (emb_{in} \oplus emb_{out})' - (emb_{in} \oplus emb_{out}) \|^2 \quad (3)$$

It learns both of the features from the both embedding. The matrix in the hidden layer, called compressed matrix, dimension size is half of the input dimension size because it extracts the data features from the input layer. As a result of the autoencoder embedding, we used the compressed matrix in the hidden layer as our dual embedding with autoencoder. This embedding has smaller dimension than CONC embedding, the original input to the autoencoder. So, we get the compressed dimension of the embedding while containing the input embedding and output embedding's information. We named this word embedding autoencoder based CONC (AE-CONC) because we used CONC embedding as our input.

We tried various different inputs to the autoencoder. First we tried CONC embedding to the input to get the same dimension of the input embedding and output embedding. For various experiment to get the better word embedding, we tried the SUM embedding as our input to the autoencoder. We named this embedding AE-SUM. Also we made SUM embedding using the weight ratio by 8:2 when adding input embedding and output embedding. We decided this weight ratio 8:2 heuristically. We named this embedding AE-SUMR. The dimension of this AE-SUM and AE-SUMR would be the half of the dimension of input embedding and output embedding. We can express words by smaller dimension with these embeddings.

3.4. Singular value decomposition

Singular value decomposition (SVD) is a way of decomposing the embedding matrix to such shapes. SVD has been utilized in diverse tasks in natural language processing like [33], [34] to get the reduced dimensionality of a feature space. The proposed method in combining vectors was introduced in [22]. They used the embedding of concatenation, CONC embedding, to the input to reduce the dimensionality. But instead, we used SUM embedding matrix compared to the better results. We used only the method to combine input embedding and output embedding. With $C = USV^T$ using the matrix of size $n \times k$, for the result, U gets unitary matrix of size $n \times k$, S gets diagonal matrix of size $n \times n$, and V gets unitary matrix of size $k \times k$. In this paper, n is the vocabulary size of the embedding and k is the embedding dimension size. We used the SUM matrix of the input embedding and output embedding as an input to C in this equation and we used U as our final embedding for SVD. We applied L2-normalization to the embeddings. SVD performs dimension reduction. For the various experiments, we also tried embedding matrix of adding input embedding and output embedding by the weight

ratio of 9:1 respectively. We named this embedding singular value decomposition ratio (SVD-R) embedding in further experiments.

3.5. 2to1

2to1 model is originated from 1toN model in [22]. 1toN embedding results fine-tuned meta embedding which contains knowledge from all individual embedding sets like word2vec [6], GloVe [29], class-weighted (CW) [2]. Different from the 1toN model, we train the word vectors from 2 embeddings, input embedding and output embedding. We first randomly initialize the embedding and then trained the vector from input embedding and output embedding with the loss function introduced in [35] to update the word embedding matrix efficiently which has enormous vocabulary size. Like in Figure 2, each loss $loss_{in}$, $loss_{out}$ from each embeddings emb_{in} , emb_{out} is used to train emb_{2to1} . This method successfully replaces the way the Softmax function is applied to all the values of the output layer.

We use this dual embedding to predict representations of the word in the individual embedding sets by projections. Also we used parameter α to find the best combination of the input embedding and the output embedding as shown in (4). This model makes the vector to have more meaningful embedding because they learn each knowledge from both embeddings.

$$loss_{total} = loss_{in} \times \alpha + loss_{out} \times (1 - \alpha) \tag{4}$$

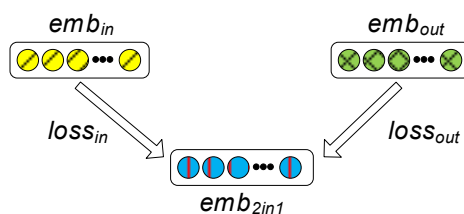


Figure 2. Visualization of using loss function in 2to1 model

4. RESULTS AND DISCUSSION

In this paper, we utilized input embedding and output embedding from the Word2Vec model, to put it concretely, the Skip-gram model, trained on dataset from “One Billion Word Language Modeling Benchmark” which consists of almost 1 billion words, and the text are already pre-processed. We set the vocabulary size to 229842, which will consist of words with high frequency, discarding the words that occur rarely. Input embedding and output embedding are both 300 dimensional vector.

The proposed dual embeddings are quantitatively evaluated on word analogy and similarity tasks, and then qualitatively on nearest neighbors of several words. We tried several ways to combine input embedding and output embedding as our dual embedding, and got several embeddings such as CONC embedding, SUM embedding, AE-CONC embedding, AE-SUM embedding, AE-SUMR embedding, SVD embedding, SVD-R embedding, and 2to1 embedding. We compared our dual embedding performances with each individual input embedding and output embedding as well as just concatenating and adding.

4.1. Word analogy task

We used semantic-syntactic word relationship test set from [6] to measure the quality of our embeddings. They have 8869 semantic and 10675 syntactic questions, which the semantic questions have categories like a male-to-female relationship. The questions is a list of 4 words which is 2 set of similar word pairs with 2 words like “he” : “she” :: “man” : “woman”. We need to find the last word in the closest word list computed with other 3 words. For example, we have to find the closest word to $vec(x)$ by cosine distance computed with $vec(“she”) - vec(“he”) + vec(“man”)$. The closest word needs to be exactly the last word in a set (the word “woman” in the above example) to count as a correct answer when we evaluate the accuracy.

The performance of the word analogy task is reported in Table 1. It is divided in semantic accuracy, syntactic accuracy, and total accuracy of the word analogy task for semantic-syntactic word relationship test set. The top 2 results are the input embedding and output embedding's results individually. We can see it at once that individual input embedding and output embedding as a word embedding perform poorly than any other dual embeddings. We observed that when we use output embedding in our experiment by combining with input embedding, the results of combined embeddings better perform than using only the input embedding. Surprisingly, when we combine input embedding in the ways of we proposed, the performance increased by

just only using output embedding with input embedding. These results in Table 1 demonstrate that our hypothesis, it would be efficient to use both of the embeddings, input and output, was right.

We found some interesting things in this experiment. Specifically, in the semantic part, SVD embedding performs the best in these embeddings. However, in the syntactic part, CONC embedding outperforms the others. It is interesting that CONC embedding and SUM embedding performs well in syntactic task with simply concatenating or adding the input embedding and output embedding. Especially, 2to1 embedding, made with the model we proposed, performs best in word analogy task among these embeddings including other dual embeddings which combine input embedding and output embedding. This show that 2to1 model has advantage on analogizing the word by forward propagating both the input embedding and output embedding.

Table 1. Accuracy on word analogy task

Embeddings	Semantic	Syntactic	Total
input	64.4	66.5	65.6
output	67.0	68.3	67.7
CONC	69.6	69.6	69.4
SUM	79.7	69.4	73.9
AE-CONC	71.6	67.3	69.2
AE-SUM	69.4	65.6	67.3
AE-SUMR	78.8	67.6	72.7
SVD	82.5	65.3	73.1
SVD-R	79.0	67.0	72.4
2to1	81.9	68.1	74.3

4.2. Word similarity task

We experimented the performance of the embeddings by Spearman rank correlation on word similarity task. A similarity score is obtained from the embedding vectors by calculating the cosine similarity after normalizing each feature across the vocabulary. Spearman's rank correlation coefficient is computed in the middle of this score and the human judgments. Table 2 shows the results by the percentage of the coefficient. We used Rubenstein-Goodenough (RG) dataset [36] with 65 word pairs, Miller-Charles (MC) dataset [37] with 30 word pairs, SimLex-999 (SL-999) dataset [38] with 999 word pairs, and rare word (RW) dataset [39] with 2034 word pairs in this word similarity task.

We tried word similarity task on individual input embedding and output embedding and other dual embeddings such as CONC, SUM, AE-CONC, AE-SUM, AE-SUMR, SVD, SVD-R, 2to1 embeddinngs as shown in Table 2. To see it generally, autoencoder embedding, especially autoencoder with SUM embedding (AE-SUM) outperforms the other dual embeddings in MC and RG dataset. Since MC and RG dataset have few word pairs compared to SL-999 dataset and RW dataset, AE-SUM embedding performs well because this embedding contains information of input embedding and output embedding in smaller dimensionality.

Table 2. Spearman rank correlationi coefficient for word similarity task

Embeddings	MC	RG	SL-999	RW
input	70.12	72.38	48.52	35.64
output	68.63	64.10	44.20	45.72
CONC	68.59	64.37	44.81	46.03
SUM	72.66	70.23	44.89	46.55
AE-CONC	78.58	78.59	44.53	43.76
AE-SUM	79.78	79.01	38.47	41.54
AE-SUMR	76.84	77.33	44.18	43.89
SVD	74.93	76.42	45.25	49.50
SVD-R	76.56	78.58	48.54	41.88
2to1	74.02	77.37	43.44	47.82

Interestingly, in the SimLex-999 dataset, SVD with ratio of 9:1 (SVD-R) embedding outperforms way better than other dual embeddings, needless to say, also better than only input embedding and output embedding. However, in RW dataset, SVD and 2to1 model generally performs better than each input and output embeddings and they are even better than simply adding input embedding and output embedding in word similarity task. Since RW dataset is consists of rare words, we can know that SVD embedding is good at capturing rare words features, i.e. powerful at representing the rare words.

4.3. Nearest neighbors

We selected several words and their nearest neighbors to show the qualitative results for input embedding, output embedding, and our dual embeddings. We did this experiment only on 3 dual embeddings, thinking that these are the representatives of the dual embeddings. In this Table 3, one of the embeddings, AutoEncoder means AE-SUMR embedding as a representative of all AutoEncoder embeddings.

The words in Table 3 are ‘language’, ‘eminem’, ‘unflagging’, and ‘remonstrate’, and ‘reprobate’. It consists of 2 frequent words (‘language’, ‘eminem’) which we all know, and 3 rare words (‘unflagging’, ‘remonstrate’, ‘reprobate’) that is hard to represent with the embedding. With 2 frequent words, in all embeddings like input embedding, output embedding, autoencoder, SVD, and 2to1 embedding have related words to each words in the results of nearest neighbors.

Table 3. Nearest neighbors with several words on dual embeddings

	language	eminem	unflagging	remonstrate	reprobate
input	languages	rapper	instilled	yelled	shortcake
	vocabulary	kanye	marvles	bargate	arand
	english	coldplay	unwavering	heurelho	sacramen
	dialect	rap	urbanity	kraig	guzzles
output	phrases	rappers	rediscovering	reasoning	loudmouth
	phonetic	outkast	unquenchable	remonstrated	7david
	aramaic	soulja	unswerving	remonstrating	turnblad
	dialects	timbaland	pasquerilla	rangana	guidenstern
AE	idioms	tinchy	untiring	olimpico	claireece
	dialect	jeezy	unstaining	skomina	deerhound
	languages	rapper	unwavering	ovrebo	philanderer
	arabic	interscope	tenacity	jeered	hissy
SVD	english	album	dedication	linesman	druggie
	dialect	timbaland	unfailing	whistled	alcoholic
	fluent	grammy	unswerving	referee	loveable
	afrikaans	rapper	unwavering	remonstrating	alcoholic
2to1	dialect	dre	unswerving	remonstrated	etonian
	fluent	interscope	unstinting	liaise	codger
	pashto	ludacris	unfailing	berserk	forma
	dialects	rihanna	unquenchable	unheeded	curmudgeon
2to1	english	rapper	unwavering	linesman	loudmouth
	languages	rap	dedication	remonstrated	druggie
	arabic	rappers	devotion	jeered	dim-witted
	vocabulary	album	tenacity	ovrebo	unlikeable
	the	kanye	unswerving	shouting	mutilates

The rare word ‘unflagging’ means never becomes weaker, ‘remonstrate’ means to complain, and ‘reprobate’ means a person of bad character and habits. For each rare words, input embedding’s nearest neighbors and output embedding’s nearest neighbors have words with totally unrelated meanings. With our dual embeddings, AutoEncoder, SVD and 2to1, however, their nearest neighbors have related words with similar meanings.

5. CONCLUSION

We found the way to better represent the word in distributed vector representation by using both the input embedding and the output embedding from training Word2Vec. Different from other works, we used embeddings from just only one model, Word2Vec, by simply combining their input embedding and output embedding. It is remarkable that we used both the input and output embeddings, especially output embedding, which Word2Vec model abandons. We know that there are incredible works in recent days to represent words almost perfectly (e.g. BERT), but this method is, in no doubts, the most simple and fast way of representing words. We demonstrated with word analogy task, word similarity task, and nearest neighbors of the dual embeddings. Proposing several dual embeddings such as CONC, SUM, AE, SVD, and 2to1 embeddings, we found various ways to represent the words. We leave it to further work to use these methods on various models. The state-of-the-arts models in word embeddings should have input embedding and output embedding when they train each model. It should be worth it to combine those two embeddings to get better performance than their own models.

ACKNOWLEDGEMENTS

This research was funded by a 2021 research Grant from Sangmyung University.





REFERENCES

- [1] K. Babić, S. M. Ipšić, and A. Meštrović, "Survey of neural text representation models," *Information (Switzerland)*, vol. 11, no. 11, pp. 1–32, Oct. 2020, doi: 10.3390/info11110511.
- [2] Y. Belinkov and J. Glass, "Analysis methods in neural language processing: A survey," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 49–72, Apr. 2019, doi: 10.1162/tacl_a_00254.
- [3] A. Madaan and Y. Yang, "Neural language modeling for contextualized temporal graph generation," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 864–881, doi: 10.18653/v1/2021.naacl-main.67.
- [4] M. Baroni, G. Dinu, and G. Kruszewski, "Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors," in *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, 2014, vol. 1, pp. 238–247, doi: 10.3115/v1/p14-1023.
- [5] Q. Jiao and S. Zhang, "A brief survey of word embedding and its recent development," in *IAEAC 2021 - IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference*, Mar. 2021, pp. 1697–1701, doi: 10.1109/IAEAC50856.2021.9390956.
- [6] S. Sonkar, A. Waters, and R. Baraniuk, "Attention word embedding," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2021, pp. 6894–6902, doi: 10.18653/v1/2020.coling-main.608.
- [7] M. E. Peters *et al.*, "Deep contextualized word representations," in *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 2018, vol. 1, pp. 2227–2237, doi: 10.18653/v1/n18-1202.
- [8] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, 2018, vol. 1, pp. 328–339, doi: 10.18653/v1/p18-1031.
- [9] M. D. Cookson and P. M. R. Stirk, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North*, 2019, pp. 4171–4186, doi: 10.18653/v1/N19-1423.
- [10] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," *Advances in Neural Information Processing Systems*, vol. 3, no. January, pp. 2177–2185, 2014, doi: 10.5555/2969033.2969070.
- [11] Z. Yang, M. Ding, C. Zhou, H. Yang, J. Zhou, and J. Tang, "Understanding negative sampling in graph representation learning," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2020, pp. 1666–1676, doi: 10.1145/3394486.3403218.
- [12] E. M. Hambi and F. Benabbou, "A deep learning based technique for plagiarism detection: A comparative study," *IAES International Journal of Artificial Intelligence*, vol. 9, no. 1, pp. 81–90, Mar. 2020, doi: 10.11591/ijai.v9.i1.pp81-90.
- [13] D. Li and D. Summers-Stay, "Dual embeddings and metrics for word and relational similarity," *Annals of Mathematics and Artificial Intelligence*, vol. 88, no. 5–6, pp. 533–547, Jun. 2020, doi: 10.1007/s10472-019-09636-8.
- [14] E. Nalisnick, B. Mitra, N. Craswell, and R. Caruana, "Improving document ranking with dual word embeddings," in *WWW 2016 Companion - Proceedings of the 25th International Conference on World Wide Web*, 2016, pp. 83–84, doi: 10.1145/2872518.2889361.
- [15] A. C. Kozlowski, M. Taddy, and J. A. Evans, "The geometry of culture: Analyzing the meanings of class through word embeddings," *American Sociological Review*, vol. 84, no. 5, pp. 905–949, Oct. 2019, doi: 10.1177/0003122419877135.
- [16] O. Press and L. Wolf, "Using the output embedding to improve language models," in *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference*, 2017, vol. 2, pp. 157–163, doi: 10.18653/v1/e17-2025.
- [17] J. Garten, K. Sagae, V. Ustun, and M. Dehghani, "Combining distributed vector representations for words," in *1st Workshop on Vector Space Modeling for Natural Language Processing, VS 2015 at the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2015*, 2015, pp. 95–101, doi: 10.3115/v1/w15-1513.
- [18] V. Ustun, P. S. Rosenbloom, K. Sagae, and A. Demski, "Distributed vector representations of words in the sigma cognitive architecture," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8598 LNAI, 2014, pp. 196–207, doi: 10.1007/978-3-319-09274-4_19.
- [19] Y. Tsuboi, "Neural networks leverage corpus-wide information for part-of-speech tagging," in *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2014, pp. 938–950, doi: 10.3115/v1/d14-1101.
- [20] Y. Luo, J. Tang, J. Yan, C. Xu, and Z. Chen, "Pre-trained multi-view Word embedding using two-side neural network," in *Proceedings of the National Conference on Artificial Intelligence*, 2014, vol. 3, pp. 1982–1988, doi: 10.5555/2892753.2892828.
- [21] J. Goikoetxea, E. Agirre, and A. Soroa, "Single or multiple? Combining word representations independently learned from text and word net," in *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, 2016, pp. 2608–2614, doi: 10.5555/3016100.3016266.
- [22] W. Yin and H. Schütze, "Learning word meta-embeddings," in *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*, 2016, vol. 3, pp. 1351–1360, doi: 10.18653/v1/p16-1128.
- [23] S. Sivakumar, L. S. Videla, T. Rajesh Kumar, J. Nagaraj, S. Itnal, and D. Haritha, "Review on Word2Vec word embedding neural net," in *Proceedings - International Conference on Smart Electronics and Communication, ICOSEC 2020*, Sep. 2020, pp. 282–290, doi: 10.1109/ICOSEC49089.2020.9215319.
- [24] P. P. Joby, "Expedient Information retrieval system for web pages using the natural language modeling," *Journal of Artificial Intelligence and Capsule Networks*, vol. 2, no. 2, pp. 100–110, Jun. 2020, doi: 10.36548/jaicn.2020.2.003.
- [25] M. A. Fauzi, "Word2Vec model for sentiment analysis of product reviews in Indonesian language," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 1, p. 525, Feb. 2019, doi: 10.11591/ijece.v9i1.pp525-530.
- [26] D. E. Cahyani and I. Patasik, "Performance comparison of tf-idf and word2vec models for emotion text classification," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 5, pp. 2780–2788, Oct. 2021, doi: 10.11591/eei.v10i5.3157.
- [27] L. K. Ramasamy, S. Kadry, and S. Lim, "Selection of optimal hyper-parameter values of support vector machine for sentiment analysis tasks using nature-inspired optimization methods," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 1, pp. 290–298, Feb. 2021, doi: 10.11591/eei.v10i1.2098.
- [28] N. S. A. Yasmin, N. A. Wahab, A. N. Anuar, and M. Bob, "Performance comparison of SVM and ANN for aerobic granular sludge," *Bulletin of Electrical Engineering and Informatics*, vol. 8, no. 4, pp. 1392–1401, Dec. 2019, doi: 10.11591/eei.v8i4.1605.





- [29] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2014, pp. 1532–1543, doi: 10.3115/v1/d14-1162.
- [30] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng, "Improving word representations via global context and multiple word prototypes," in *50th Annual Meeting of the Association for Computational Linguistics, ACL 2012 - Proceedings of the Conference*, 2012, vol. 1, pp. 873–882, doi: 10.5555/2390524.2390645.
- [31] J. N. Coates and D. Bollegala, "Frustratingly easy meta-embedding-computing meta-embeddings by averaging source word embeddings," in *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 2018, vol. 2, pp. 194–198, doi: 10.18653/v1/n18-2031.
- [32] C. Bao and D. Bollegala, "Learning word meta-embeddings by autoencoding," in *COLING 2018-27th International Conference on Computational Linguistics, Proceedings*, 2018, pp. 1650–1661.
- [33] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, Sep. 1990, doi: 10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASII>3.0.CO;2-9.
- [34] P. D. Turney, "Measuring semantic similarity by latent relational analysis," in *IJCAI International Joint Conference on Artificial Intelligence*, 2005, pp. 1136–1141, doi: 10.5555/1642293.1642475.
- [35] S. Rida-E-Fatima *et al.*, "A multi-layer dual attention deep learning model with refined word embeddings for aspect-based sentiment analysis," *IEEE Access*, vol. 7, pp. 114795–114807, 2019, doi: 10.1109/ACCESS.2019.2927281.
- [36] H. Rubenstein and J. B. Goodenough, "Contextual correlates of synonymy," *Communications of the ACM*, vol. 8, no. 10, pp. 627–633, Oct. 1965, doi: 10.1145/365628.365657.
- [37] G. A. Miller and W. G. Charles, "Contextual correlates of semantic similarity," *Language and Cognitive Processes*, vol. 6, no. 1, pp. 1–28, Jan. 1991, doi: 10.1080/01690969108406936.
- [38] F. Hill, R. Reichart, and A. Korhonen, "Simlex-999: Evaluating semantic models with (Genuine) similarity estimation," *Computational Linguistics*, vol. 41, no. 4, pp. 665–695, Dec. 2015, doi: 10.1162/COLI_a_00237.
- [39] M. T. Pilehvar, D. Kartsaklis, V. Prokhorov, and N. Collier, "CARD-660: Cambridge rare word dataset - A reliable benchmark for infrequent word representation models," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, 2018, pp. 1391–1401, doi: 10.18653/v1/d18-1169.

BIOGRAPHIES OF AUTHORS







Yoonjoo Ahn     received B.S. degrees in Computer Science from Dongduk Women's University in 2018. She is a graduate student in Sangmyung University. She is currently working in Department of IT at KIS Pricing Inc. Her research interests include artificial intelligence, natural language processing, and finance engineering. She can be contacted at email: yoonjoo30@naver.com.



Eugene Rhee     received a Ph.D. degree in electronics from Hanyang University, Korea, in 2010. He was a visiting professor at Chuo University, Japan from 2010 to 2011. Since 2012, he has been with Sangmyung University, Korea, where he is currently an associate professor in the Department of Electronic Engineering. His research area includes microwaves, electromagnetic compatibility, electromagnetic interference, and reverberation chambers. He can be contacted at email: eugenerhee@smu.ac.kr.



Jihoon Lee     received B.S., M.S, and Ph.D. degrees in electronics engineering from Korea University in 1996, 1998, and 2001, respectively. From 2002 to 2011, he worked at Samsung Electronics as a senior research member. He is currently an associate professor in the Department of Smart Information and Telecommunication Engineering, Sangmyung University, Korea. His research interests include edge computing, secure M2M, and network security. He can be contacted at email: vincent@smu.ac.kr.