Semantically Readable Distributed Representation Learning for Social Media Mining

Ikuo Keshi
Nara Institute of Science and Technology
keshi.ikuo.ka9@is.naist.jp

Yu Suzuki
Nara Institute of Science and Technology
ysuzuki@is.naist.jp

Koichiro Yoshino
Nara Institute of Science and Technology
koichiro@is.naist.jp

Satoshi Nakamura
Nara Institute of Science and Technology
s-nakamura@is.naist.jp

ABSTRACT

The problem with distributed representations generated by neural networks is that the meaning of the features is difficult to understand. We propose a new method that gives a specific meaning to each node of a hidden layer by introducing a manually created word semantic vector dictionary into the initial weights and by using paragraph vector models. Our experimental results demonstrated that weights obtained based on learning and weights based on the dictionary are more strongly correlated in a closed test and more weakly correlated in an open test, compared with the results of a control test. Additionally, we found that the learned vector are better than the performance of the existing paragraph vector in the evaluation of the sentiment analysis task. Finally, we determined the readability of document embedding in a user test. The definition of readability in this paper is that people can understand the meaning of large weighted features of distributed representations. A total of 52.4% of the top five weighted hidden nodes were related to tweets where one of the paragraph vector models learned the document embedding. Because each hidden node maintains a specific meaning, the proposed method succeeds in improving readability.

KEYWORDS

Distributed representation learning, semantic vector, semantic lexicon, paragraph vector, word2vec, Twitter, sentiment analysis

ACM Reference format:
Ikuo Keshi, Yu Suzuki, Koichiro Yoshino, and Satoshi Nakamura. 2017. Semantically Readable Distributed Representation Learning for Social Media Mining. In *Proceedings of WI '17, Leipzig, Germany, August 23-26, 2017,* 7 pages.
https://doi.org/10.1145/3106426.3106521

1 INTRODUCTION

Distributed representations named word2vec and paragraph vectors, computed using simple neural networks with context information as features, have been widely used [8, 11–13]. The paragraph vectors achieved state-of-the-art results on sentiment analysis at the time of publication [8]. The problem in engineering with the distributed representations of words and paragraphs is that the meaning of the distributed representations is difficult to understand. Thus, tests and improvement in the quality of the application system are necessary because the distributed representations are not readable as is.

WordNet [14], FrameNet [1], both of which are in English, and a word semantic vector dictionary [5] in Japanese using manual construction have been proposed. The word semantic vector expresses the relationship between a word and 266 feature words as a binary value that is related or unrelated. The word semantic vector dictionary includes feature words related to each core word comprising 20,330 important words in Japanese. We proposed an integration method to learn words expanded using the word semantic vector dictionary with a paragraph vector model to solve the problem of word sparsity in Twitter [6]. The integration of the word semantic vector and paragraph vector learning showed that the accuracy of sentiment analysis improves by learning context information of a particular domain even if words are sparse. We showed that expanded feature words for Tweets can be used for error analysis of sentiment analysis, but it was still difficult to read the features of distributed representations.

This paper proposes a new method of automatically learning readable distributed representations using the word2vec and paragraph vector models based on the word semantic vector dictionary [5]. Word semantic vector dictionaries are more like distributed representations rather than semantic lexicons like WordNet [14] and FrameNet [1] because each core word is defined as a fixed-length dense vector. More specifically, 266 feature words are taken as the hidden nodes of each model. Then, the initial weights between the input word and each hidden node, which is a seed vector, are given based on the dictionary. We investigated whether or not the meaning of each hidden node is maintained, even after learning is done by the neural networks related to core words. Also, the meaning of the hidden nodes is maintained to some extent in new words. If the learning is insufficient, then the weights after learning will naturally correlate with the initial weights. However, the accuracy of sentiment analysis using distributed representations after learning by the proposed method is better than the paragraph vector performance.

Finally, we present our evaluation of the readability of document embedding in a user test conducted through crowdsourcing. The definition of readability in this paper is that people can understand the meaning of large weighted features of distributed representations. A total of 66.1% and 52.4% of the given feature words were related to tweets where one of the paragraph vector models learned the document embedding for the top weighted hidden node and the top five weighted hidden nodes, respectively. Therefore, our method improves the readability of the distributed representations, which are the weights of each hidden node for words and paragraphs. Also, it can be applied to mining of social media such as Twitter by using each feature word as a conceptual axis.

2 RELATED WORK

Research on the integration of external semantic lexicons and distributed representation learning has been active. The following three research directions are being studied.

- **Pre-processing.** Tweet2vec trained the CNN-LSTM encoder-decoder model on 3M randomly selected tweets populated using data augmentation techniques, which are useful for controlling generalization error for a deep learning model [20]. Data augmentation techniques refer to replicating tweets and replacing some words with their synonyms using WordNet [14].
  Feature word expansion on Twitter of our previous proposal is part of this direction [6].
- **Learning process.** RC-NET [21] is built upon the Skip-gram model [12], the objective function of which is extended by incorporating both the relational knowledge (like is-a, etc) and the categorical knowledge (like synonyms) as regularization functions. Bollegala et al. proposed a global word co-occurrence prediction method [15] using the semantic relations in WordNet as a regularizer [3].
  Our proposal in this paper is part of this direction.
- **Post-processing.** Retrofitting [4] is a technique for fitting learned word vectors to semantic lexicons.
  In this paper, we used this technique to create initial weights of core words.

Experiments have shown that the precision of distributed representations of words has qualitatively improved the best in [3] and that the accuracy of sentiment analysis has improved in [20]. Both showed that they were state-of-the-arts techniques using standard datasets on word similarity and sentiment analysis. Similar studies have been done using topic models based on LDA [2]. Topical word embeddings(TWE) [10], in which "topical word" refers to a word taking a specific topic, have been proposed to measure contextual word similarity by extending the Skip-gram model [12]. Also TWE outperformed the Skip-gram model in word similarity tasks. TWE is also applied to tweet topic classification tasks and performs better than paragraph vectors [9]. However, no reports on the relevant literature describe an attempt to give meaning to each hidden node.

One model of paragraph vectors (PV-DBOW) [8] uses pre-trained word embeddings that reportedly improve task performance [7].

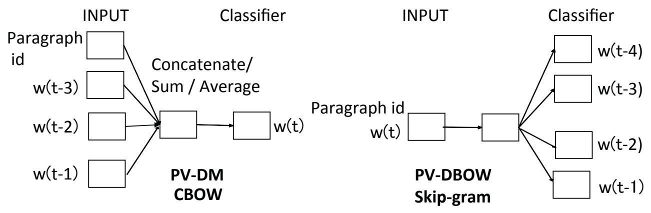


Figure 1: Word2vec and paragraph vector models.

Although this paper shows the possibility of learning proper document embedding with good initialization of word embeddings, it does not demonstrate the possibility of interpretation of hidden nodes.

The point to emphasize is that topic models, for example, can be used to assign topic numbers and related keywords to each tweet, but people cannot understand the meaning of large weighted features of tweet embedding using any of the conventional methods.

3 PROPOSED METHOD

3.1 Hypothesis

This paper presents a test of the hypothesis that the meaning of each hidden node is maintained using three approaches even after learning: using word2vec and paragraph vector models, assigning specific meaning to each hidden node, and giving the strength of the semantic and associative relationship with each hidden node as the initial weights of important words.

The neural network learns the concept automatically for the hidden layer. Thus, we thought the weights would adapt to the context with the concept maintained by pre-setting the appropriate conceptual classification to be learned to the nodes of the hidden layer and by giving the suitable initial value.

3.2 Word2vec and Paragraph Vector Models

Figure 1 presents two variants of word2vec and paragraph vector models [8]. The distributed memory model of paragraph vectors (PV-DM) predicts the target word vector of the next word w (t) from the context vector obtained by adding a paragraph ID to input words within the context window. The continuous bag of words (CBOW) of word2vec does not add the paragraph ID to the input layer, but it is fundamentally the same as the PV-DM.

The paragraph vector with a distributed bag of words (PV-DBOW) learns the paragraph vector to predict the context word vectors of randomly selected surrounding words within the context window. Skip-gram of word2vec is used to learn the vectors of the target words in the PV-DBOW. In Skip-gram, the target word vector is learned so that the inner product of the target word vector and the context word vector of the surrounding words is larger than the inner product of the context word vector of words other than the surrounding words.

3.3 Word Semantic Vector Dictionary

We selected 266 conceptual classifications that belong to six major classes and 29 upper concepts as feature words in the word semantic vector dictionary [5], as presented in Table 1. For core words, we

Table 1: Classification of feature words.

Six large classifications	Examples of 29 upper concepts	Examples of 266 feature words
Human· Life	Human Creature	Human, Name, Male, Female, Child Animal, Bird, Insect, Microbe, Plant
Human environment	Artificiality Traffic·Communication	Tool, Mechanical·Component, Building Communication, Traffic·Transportation
Natural environment	Area Nature	Place name, Country name, Japan, City Land, Mountain, Sky, Ocean
Abstract concept	Spirit·Psychology Abstract concept	Sense, Emotion, Happiness, Sadness State·Aspect, Change, Relationship
Physics· Substance	Motion Physical characteristics	Motion, Halt, Dynamic, Static Warmth, Weight, Lightness, Flexible
Civilization· Information	Humanities Science	Race, Knowledge, Speech Mathematics, Physics, Astronomy

Table 2: Grant criteria by logical relationship.

Logical relationship	Core words ãĂĂ	Feature words
Class inclusion	Autumn	Season
Synonym relationship	Idea	Thought
Part–whole relationship	Leg	Human body

Table 3: Grant criteria by associative relationship.

Core words	Feature words
Love	Kindness, Warmth
Up	Economy, Video
Leg	Car, Traffic·Transportation

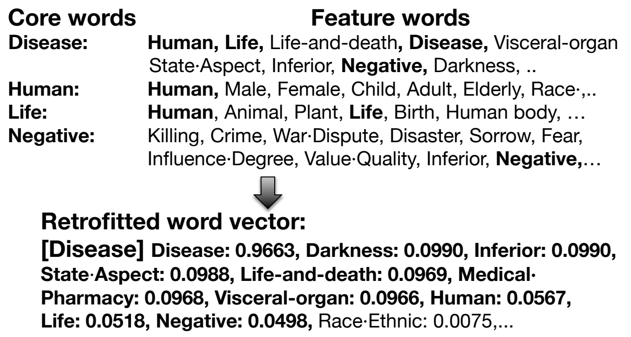
selected 20,330 words from encyclopedias, newspapers, websites, instruction manuals, and Kansei words.

A human expert assigned feature words to each core word based on the following criteria. Feature words were assigned from a logical and associative relationship. The logical relationship refers to those in which the feature words have direct relevance for core words, as shown in Table 2. The associative relationship refers to those in which feature words are related to core words by association, as shown in Table 3.

3.4 Model setting for testing the hypothesis

In this section, we describe the setting to encode the initial weights of the core words based on the strength of the relationship with each feature word, and we describe our test of the hypothesis using Skip-gram as an example.

A method has been proposed for generating a word vector by recursively expanding a definition sentence for a word in a dictionary [19]. The word semantic vector dictionary can be regarded as defining a core word with 266 types of feature words. Because feature words are also core words, recursive extension is necessary. However, convergence occurs when the feature word is expanded several times because the definition sentence of the core word is limited to 266 words. Also, a method has been proposed for retrofitting word vectors according to related words in a dictionary [4]. In that method, we generate a seed vector of the core word by recursively expanding the dictionary using retrofitting tools[1].



Core words **Feature words**
Disease: **Human, Life,** Life-and-death**, Disease,** Visceral-organ State·Aspect, Inferior, **Negative,** Darkness, ..
Human: **Human,** Male, Female, Child, Adult, Elderly, Race·,..
Life: **Human**, Animal, Plant, **Life**, Birth, Human body, …
Negative: Killing, Crime, War·Dispute, Disaster, Sorrow, Fear, Influence·Degree, Value·Quality, Inferior, **Negative,**…

Retrofitted word vector:
[Disease] Disease: 0.9663, Darkness: 0.0990, Inferior: 0.0990, State·Aspect: 0.0988, Life-and-death: 0.0969, Medical· Pharmacy: 0.0968, Visceral-organ: 0.0966, Human: 0.0567, Life: 0.0518, Negative: 0.0498, Race·Ethnic: 0.0075,...

Figure 2: Example of retrofitting "Disease."

When building a vocabulary from the corpus, the initial vectors of the following two kinds are created first.

- The 266 feature words are added to the vocabulary as one-hot vectors with dimensions corresponding to respective feature words set to 1.
- Other initial word vectors including core words extracted from the corpus are 266-dimensional zero vectors.

The algorithm aims at bringing word vectors closer to the relationship of the word entries of the lexicon as post-processing of learning of word vectors. We applied this algorithm for retrofitting the aforementioned initial word vectors, which are 266-dimensional one-hot or zero vectors, into the word semantic vector dictionary. The retrofitting algorithm is shown as the following online update [4]:

$$\mathbf{q_i} = \frac{\sum_{j:(i,j)\in E} \beta_{ij}\mathbf{q_j} + \alpha_i \hat{\mathbf{q}}_\mathbf{i}}{\sum_{j:(i,j)\in E} \beta_{ij} + \alpha_i} \tag{1}$$

$\mathbf{q_i}$ is the retrofitted word vector for the core word w_i, $\hat{\mathbf{q}}_\mathbf{i}$ is the aforementioned initial vector for w_i, and α_i is the weight of the initial vector; currently it is set to the number of given feature words w_j for w_i. $\mathbf{q_j}$ is the retrofitted word vector for the given feature word w_j, and β_{ij} is the weight of the given feature word w_j for the core word w_i; currently, the weight β_{ij} is set to 1. Equation 1 multiplies the initial vector $\hat{\mathbf{q}}_\mathbf{i}$ of the core word w_i by the weight α_i, adding the vectors obtained by multiplying the retrofitted vector $\mathbf{q_j}$ of the given feature word w_j by the weight β_{ij} and by dividing it by the sum of both weights. Running the procedure for about ten iterations increases the relationship between each core word and 266 feature words from an average of 9 to an average of 100. The relationship is increased for each core words to expand the feature words given to the core word recursively.

Figure 2 presents an example of retrofitting "Disease," which is a feature word and core word in the dictionary. The points of this algorithm are the following.

- The retrofitted word vector is close to the original vector. In the case of "Disease," the original vector is a one-hot vector.
- When the feature words assigned to a retrofitted core word are not expanded as core words, the weights of the feature

[1]https://github.com/mfaruqui/retrofitting

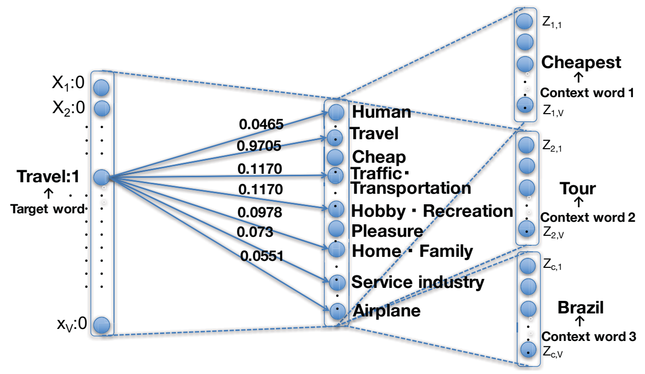


Figure 3: Skip-gram model setting for testing.

words are almost equal. When expanded, the weights decrease according to the number of feature words to be expanded.

In the vocabulary, each word has two vectors. One is an input vector, which is the weights between the input node and each hidden node, and the other is an output vector, which is the weight between each hidden node and the output node. The retrofitted word vector was used as the seed vector of the input vector. The initial weights of the output vector were set to 0, which is the default setting of gensim's doc2vec library[2].

Figure 3 presents an example of the Skip-gram setting for testing the hypothesis. The input layer specifies the target word. The output layer consists of three context words appearing around the target word. The hidden layer comprises the nodes corresponding to 266 feature words. The weights of the target word for each hidden node are retrofitted weights. Each weight is updated by back propagation so that the probability of predicting the context words increases when the target word is input. The objective function is the following [12, 16].

$$E = -\log\sigma\left(\mathbf{v}'_{\mathbf{w}}{}^{T}\mathbf{h}\right) - \sum_{w_j \in W_{neg}} \log\sigma\left(-\mathbf{v}'_{\mathbf{w_j}}{}^{T}\mathbf{h}\right) \tag{2}$$

The hidden layer outputting $\mathbf{h}$ is $\mathbf{v}_{\mathbf{w_i}}{}^{T}$. $\mathbf{v}_{\mathbf{w}}$ is an input vector with an initial vector that is generated by Equation 1, and $\mathbf{v}'_{\mathbf{w}}$ is the output vector of the word w. W_{neg} is the set of words for negative sampling. The output vector is updated as follows [16].

$$\mathbf{v}'_{\mathbf{w_j}}{}^{(new)} = \mathbf{v}'_{\mathbf{w_j}}{}^{(old)} - \eta\left(\sigma\left(\mathbf{v}'_{\mathbf{w_j}}{}^{(old)T}\mathbf{h}\right) - t_j\right)\mathbf{h} \tag{3}$$

where t_j is 1 when w_j is the context word and 0 otherwise. The initial output vector $\mathbf{v}'_{\mathbf{w}}$ is 0. Thus, the output vectors of the context words become close to the input vector, which is the seed vector, of the target word.

4 EXPERIMENTS

In these experiments, we examined the relationship between sentiment analysis using a single domain benchmark and readability of tweet embedding in a user test. We also tested the hypothesis on whether or not weights obtained based on learning and weights based on the dictionary are correlated in a closed test and an open test, compared with a control test.

Table 4: Configuration of the corpus.

Dataset	Positive	Negative	Neutral
Training set	3654	2375	2802
Dev. set	608	396	467
Test set	609	396	467
Unlabeled tweets		560,853	

Table 5: Hyper-parameter settings for learning word vectors.

Hyper-parameters	Values
Dimensionality of the feature vectors	266
Number of iterations over the corpus	20
Learning rate	Initial:0.025, Minimum:0.0001
Window size	5
Downsample threshold for words	1e-5
Number of negative sampling words	15

Table 6: Example of retrofitted and learned word vectors for a core word that is a feature word itself.

Generation method	feature words and weights arranged in descending order
Retrofitted vector for "**travel**"	**travel**:0.97, traffic·transportation:0.12, hobby·recreation:0.1, home ·family:0.1, service industry:0.1, airplane:0.06, human:0.05, car:0.05, overseas:0.05, Japan:0.05,
learned vector for "**travel**" by PV-DM	**travel**:1.41, **machine**:0.65, **image**:0.61, company:0.55, state·aspect:0.52, traffic·transportation:0.5, hobby·recreation:0.43, education:0.40, facility:0.38, behavior:0.36,
learned vector for "**travel**" by PV-DBOW	**travel**:1.45, time:0.45, custom:0.44, clothes:0.43, state·aspect:0.43, Europe:0.42, low:0.42, **image**:0.41, public system:0.40, **machine**:0.40,

4.1 Corpus

We collected Twitter data about two product brands of smartphones, as shown in Table 4. For the 560,853 unlabeled tweets, only noises such as the URL and the account name were deleted. The evaluation benchmark of sentiment analysis consisted of 11,774 tweets of one product brand labeled using crowdsourcing as either positive, negative, or neutral [6]. Japanese morphological analysis, MeCab[3] and its dictionary mecab-ipadic-NEologd[4], which expanded it by millions of new words and named entities from language resources on the Web, were used to extract words from tweets. The number of words extracted from the corpus five or more times was 30,468, while the number of retrofitted core words was 6,814 words.

4.2 Learning Word Vectors by Our Method and Evaluation of Correlation Coefficients

First, the word vector was updated using two variants of paragraph vector models with unlabeled tweets only using gensim's doc2vec library. On the basis of the accuracy of the sentiment analysis of the final stage, we decided the values of hyper-parameters for

[2]https://radimrehurek.com/gensim/models/doc2vec.html

[3]http://mecab.googlecode.com/svn/trunk/mecab/doc/index.html
[4]https://github.com/neologd/mecab-ipadic-neologd

Table 7: Example of retrofitted and learned word vectors for a core word that is not a feature word.

Generation method	feature words and weights arranged in descending order
Retrofitted vector for "**screen**"	**computer**:0.42, machine:0.42, communication tech.:0.42, mass media:0.42, electronics:0.40, **plane**:0.22, **color**:0.22, communication:0.22, advertisement:0.04,
learned vector for "**screen**" by PV-DM	**computer**:0.83, machine:0.78, **image**:0.78, state·aspect:0.73, company:0.68, **plane**:0.61, education:0.55, book:0.55, facility:0.52, **color**:0.50,
learned vector for "**screen**" by PV-DBOW	state·aspect:0.68, **plane**:0.50, **computer**:0.45, **image**:0.34, human body:0.32, relationship:0.31, chemistry:0.31, civil engineering·architecture:0.29, tool:0.26, **color**:0.26,

Table 8: Evaluation results 1: correlation coefficients between initial and learned word vectors.

	Control Test	Closed Test	Open Test
PV-DM/CBOW	0.224	0.608	0.340
PV-DBOW/Skip-gram	0.211	0.642	0.395

paragraph vector learning of the conventional method. The hyper-parameter settings for learning the corpus are shown in Table 5. Our method used the same hyper-parameter settings. Here, the size of the feature vectors was adjusted to the number of feature words, 266. When the number of dimensions of the feature vectors exceeded 266, we could set the initial value 0 or the random number for the part exceeding 266 in our method. However, no difference occurred in accuracy between 266 dimensions and 300 dimensions for the corpus in the paragraph vector of the conventional method. Thus, we utilized 266 dimensions. Both the PV-DM and PV-DBOW have the same hyper-parameter settings. We used the sum of the input vectors for the hidden layer of the PV-DM for the same reason as with the hyper-parameter settings.

Table 6 shows an example of retrofitted and learned word vectors for a core word "travel," which is a feature word itself. The retrofitted word vector for "travel" was similar to a one-hot vector. The weight of the feature word "travel" of the learned word vector "travel" was more than twice the weight of other feature words in the PV-DM and more than three times the weight of other feature words in the PV-DBOW. Because our method learned the word vectors with a smartphone corpus, "machine" and "image" had higher weights in both of the learned word vectors. Table 7 shows an example of retrofitted and learned word vectors for a core word "screen," which is not a feature word. Feature words with the top eight weights of retrofitted vectors for "screen" are those given to the core word "screen." Of these, the feature word "computer," "plane" and "color" had higher weights in both of the learned word vectors. Although "image" is not a feature word assigned to the core word "screen," it gained a high score in both learning methods with the smartphone corpus.

Table 8 presents correlation coefficients between retrofitted vectors and learned vectors in the closed and open test, compared with those of the control test. The control test shows the correlation between the word vectors after learning by the conventional method and the initial vectors, the closed test shows the correlation for the core words used for learning by the proposed method, and the open test shows the correlation for the core words not used for learning by the proposed method as follows.

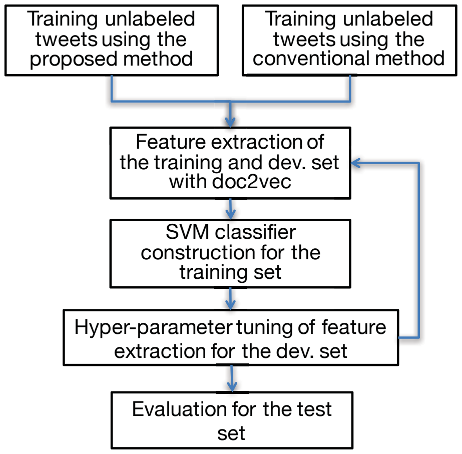

Training unlabeled tweets using the proposed method
Training unlabeled tweets using the conventional method
Feature extraction of the training and dev. set with doc2vec
SVM classifier construction for the training set
Hyper-parameter tuning of feature extraction for the dev. set
Evaluation for the test set

Figure 4: Procedures of sentiment analysis.

- **Control test:** We selected the core words (814 words) in the top 2% high-frequency words (2343 words) for the evaluation because high-frequency words had a stronger influence on tweet vector learning than low-frequency words. We evaluated the correlation coefficients between the initial vectors as a control test using default random initialization and the learned vectors.
- **Closed test:** For the 814 core words, we combined all elements of retrofitted word vectors with 0.013 or more as one vector and similarly learned word vectors. A feature word with a value of 0.013 or less corresponded to a relationship according to a two-step recursion with the core word. Therefore, we excluded feature words having a value less than 0.013 from the calculation of the correlation coefficient because the relationship with the core word is not high. Then, we calculated the correlation coefficients of the two vectors. The results showed a stronger correlation compared with that of the control test.
- **Open test:** Let the word vectors learn for the unlabeled tweets excluding the aforementioned 814 retrofitted core word vectors. Subsequently, we calculated the correlation coefficient between the aforementioned 814 retrofitted core word vectors with 0.013 or more and the corresponding 814 learned word vectors. The results showed a weak correlation.

4.3 Evaluation of Sentiment Analysis

Second, we evaluate the tasks of sentiment analysis using the paragraph vector of the conventional method and our method. The experimental procedures are presented in Figure 4. The only difference between the methods was the initial weights when learning the unlabeled tweets. Specifically, the evaluation steps were the following.

Table 9: Evaluation results 2: F-score for predicting positive and negative tweets in 3-class sentiment analysis.

	Dev. Set	Test Set
Conventional Method	68.6	68.8
Our Method	70.5*	70.2**

vs. Conventional Method *p=0.0006<0.05 **p=1.9e-05<0.05

Table 10: Evaluation results 3: Readability of hidden nodes and F-score of the corresponding 2-class sentiment analysis.

	Readability of hidden nodes			Sentiment Analysis
Tweet Vectors	Top1	Top5	Top10	F-score
PV-DBOW Positive	64.5%	56.1%	46.6%	86.8
PV-DBOW Negative	66.8%	47.7%	41.5%	78.3
PV-DBOW All	66.1%	52.4%	44.1%	82.5
PV-DM Positive	56.4%	46.2%	36.9%	80.2
PV-DM Negative	67.3%	43.8%	37.8%	74.7
PV-DM All	61.9%	45.0%	37.3%	77.5
Control Test for Positive		16.1%	15.4%	80.2
Control Test for Negative		13.9%	13.9%	74.7
Control Test for All		15.0%	14.6%	77.5

- **[Step 1]** In our method, we updated word vectors by having it learn unlabeled tweets with the PV-DM and PV-DBOW based on the retrofitted word vectors, as described in the previous section. In the conventional method, we used the same settings of the hyper-parameters shown in Table 5 for the PV-DM and PV-DBOW based on standard random initialization.
- **[Step 2]** In learning the paragraph vector of the training and dev. set, let the word vectors learned in Step 1 be the initial value of the word vectors. We combined the PV-DBOW and PV-DM for each tweet and created a tweet feature vector. We built a tweet classifier with a support vector machine (SVM) using the labels of each training tweet as teacher data.
- **[Step 3]** Each tweet vector and its label of the development set was entered into the classifier, and the error rate [=100-(F_{pos}+F_{neg})/2] was measured. Bayesian optimization automatically adjusted the parameters of the paragraph vector learning so that the output of the objective function, which is the error rate, was minimized [18].

After Step 2 and Step 3 were repeated until the error rate converged, the hyper-parameter of the paragraph vector learning was determined.

As presented in Table 9, we found that the evaluation results of our method were better than those of the conventional method in the macro-average F-score [=(F_{pos}+F_{neg})/2] of positive and negative prediction in three-class classification, which is a rating measure utilized by SemEval [17].

4.4 User Test for Readability

Finally, we conducted a readability user test of hidden nodes for the learned tweet vectors. We prepared feature words for hidden nodes with top ten weights for 30 tweets, which were estimated to be positive or negative by using the PV-DM and PV-DBOW individually. The top ten feature words were given for each tweet, and 30 user testers were asked through crowdsourcing whether or not each tweet was associated with the ten feature words. Table 10 shows what percentage of the Top 1, 5, and 10 weighted hidden nodes of PV-DBOW or PV-DM tweet vectors that were classified as either positive or negative were related to the tweets. For the PV-DBOW, the percentage of the given feature words were related to the tweets where one of the paragraph vector models learned the document embedding was 66.1% for the top weighted hidden node and was 52.4% for the top five weighted hidden nodes. The PV-DM results were 61.9% for the top weighted hidden nodes and 45.0% for the top five weighted hidden nodes. Overall, the PV-DBOW readability was better than that of PV-DM, though the PV-DM results tended to be more readable for negative tweets compared with those of the PV-DBOW. The third results show a control test. The control test assessed how user testers scored on five or ten feature words randomly chosen for the tweets classified as either positive or negative using PV-DM. The reason is that none of the conventional methods which give the meaning of features of tweet embedding. A comparison with the control test showed that our method apparently improves the readability of distributed representation learning.

Table 10 also shows the F-score in 2-class sentiment analysis using the corresponding positive and negative vectors for reference. A common trend was evident in the readability of the hidden nodes and sentiment analysis.

For the five cases of the paragraph vector (PV-DBOW) in the aforementioned readability user test, each tweet and a list in descending order of the weight of the feature words are shown below.

Product B is amazing!

[**power, strong, human**, worth, state·aspect, facility, education, positive,]

Product B is a godsend!

[state·aspect, thought, **human**, relationship, life and death, existence, **power, strong**,]

Oh, after all, the sound of Product B is something good and deep.

[sound, advertisement, state·aspect, image, **emotion**, worth, **music**, power, sense,]

While listening to the music with Product B, I was surprised with how good the sound quality was.

[state·aspect,**music**, facility, action, ethics, service industry,**emotion**, quantity,]

Even if Product B is fully charged, the LED remains lit.

[**brightness**, machine, **luminescence**, state·aspect, computer, activity, essence,]

For the first two similar tweets, the common feature words "power," "strong," and "human" had higher weights. Other tweets with the PV-DBOW tweet vectors that were classified as positive and the three feature words with the top ten weights were as follows.

"Product B is the best." "Product B is the most attractive."

In the following two tweets on the sound quality of smartphones, the common feature words "music" and "emotion" had higher weights. In the last tweet on charging and LED lighting, the feature words "brightness" and "luminescence" had higher weights. Also, importantly, clear differences emerged in the top feature words of these three groups' tweets.

The results of this readability user test revealed the following points.

- By looking up the top feature words of the learned tweet vectors, you can determine whether or not the learning is proceeding well.
- The conceptual axis can further filter the results of the sentiment analysis for social media mining. Each feature word itself or a combination of feature words becomes a conceptual axis.

5 CONCLUSION

We proposed a new method to give specific meaning to each node of a hidden layer in neural networks using a word semantic vector dictionary to enable the readability of word and document embeddings. We tested a method of maintaining the meaning of hidden nodes and found that the macro-average F-score of sentiment analysis was better than that of the conventional method. The meaning was maintained to some extent not only for core words but even for new words. We also tested the readability of hidden nodes in a user test. A total of 52.4% of the top five weighted feature words were related to tweets.

The proposed method improved the readability of distributed representations because these distributed representations of words and paragraphs learned by neural networks are weights for each hidden node with a specific meaning.

Future research will be conducted to optimize 264 feature words. We will also determine whether or not our method is universal and applicable to other languages using a standard English dataset of a sentiment analysis task, word similarity task, and word analogy task. We can also evaluate related work [3, 21] using WordNet as a regularizer of the learning process in our method.

We believe that the performance of the task of social media mining could be improved by our method without any annotated data because the performance of sentiment analysis and the readability of document embedding show similar trends.

ACKNOWLEDGMENTS

The research results have been achieved by "Research and Development on Fundamental and Utilization Technologies for Social Big Data" the Commissioned Research of National Institute of Information and Communications Technology (NICT) and NAIST Bigdata Project. This study was partially supported by CREST JST. It was also supported by Sharp Corporation.

REFERENCES

[1] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*. 86–90.

[2] Blei, David M. and Ng, Andrew Y. and Jordan, Michael I. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3 (2003), 993–1022.

[3] Danushka Bollegala, Alsuhaibani Mohammed, Takanori Maehara, and Ken-ichi Kawarabayashi. 2016. Joint Word Representation Learning Using a Corpus and a Semantic Lexicon. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. 2690–2696.

[4] Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting Word Vectors to Semantic Lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1606–1615.

[5] Ikuo Keshi, Hiroshi Ikeuchi, and Ken'ichi Kuromusha. 1996. Associative image retrieval using knowledge in encyclopedia text. *Systems and Computers in Japan* 27, 12 (1996), 53–62.

[6] Ikuo Keshi, Yu Suzuki, Koichiro Yoshino, Graham Neubig, Kazuto Ohara, Toshiro Mukai, and Satoshi Nakamura. 2017. Reputation Information Extraction from Twitter Using a Word Semantic Vector Dictionary. *IEICE TRANSACTIONS on Information and Systems (Japanese Edition)* J100-D, 4 (2017), 530–543.

[7] Jey Han Lau and Timothy Baldwin. 2016. An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. In *Proceedings of the 1st Workshop on Representation Learning for NLP*. Association for Computational Linguistics, 78–86.

[8] Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the 31th International Conference on Machine Learning*. 1188–1196.

[9] Quanzhi Li, Sameena Shah, Xiaomo Liu, Armineh Nourbakhsh, and Rui Fang. 2016. Tweet Topic Classification Using Distributed Language Representations. In *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. 81–88.

[10] Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical Word Embeddings. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2418–2424.

[11] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013).

[12] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *CoRR* abs/1310.4546 (2013).

[13] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association of Computational Linguistics : Human Language Technologies*. 746–751.

[14] George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[15] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation.. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*,, Vol. 14. 1532–1543.

[16] Xin Rong. 2014. word2vec Parameter Learning Explained. *CoRR* abs/1411.2738 (2014).

[17] S. Rosenthal, P. Nakov, S. Kiritchenko, S. M Mohammad, A. Ritter, and V. Stoyanov. 2015. SemEval-2015 Task 10: Sentiment analysis in Twitter, In Proceedings of the 9th International Workshop on Semantic Evaluation. *SemEval-2015*, 451–463.

[18] J. Snoek, H. Larochelle, and R. P. Adams. 2012. Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems* (2012), 2951–2959.

[19] S.Suzuki. 2003. Probabilistic Word Vector and Similarity based on Dictionaries. In *Proceedings of International Conference on Intelligent Text Processing and Computational Linguistics*. 564–574.

[20] Soroush Vosoughi, Prashanth Vijayaraghavan, and Deb Roy. 2016. Tweet2Vec: Learning Tweet Embeddings Using Character-level CNN-LSTM Encoder-Decoder. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1041–1044.

[21] Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. RC-NET: A General Framework for Incorporating Knowledge into Word Representations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. 1219–1228.