# Composing Word Vectors for Japanese Compound Words Using Dependency Relations

Kanako Komiya[1] Takumi Seitou[1] Minoru Sasaki[1] and Hiroyuki Shinnou[1]

Ibaraki University, 4-12-1 Nakanarusawa, Hitachi-shi, Ibaraki, 316-8511 JAPAN,
{kanako.komiya.nlp, 14t4037r,
minoru.sasaki.01,hiroyuki.shinnou.0828}@vc.ibaraki.ac.jp

**Abstract.** The use of distributed representations, e.g., via word2vec, has become popular in recent years. However, Japanese has many compound words and we often face the situation where meanings of a word and a compound word should be compared. Therefore, in the current study, we composed compound word vectors from those of constituent word vectors. We took into consideration the dependency relations of compound words to compose word vectors of them. The experiments revealed that, when we consider dependency relations, (1) we could obtain better representations for compound words when we separately learn models for each dependency relation, (2) each model could obtain good representations with fewer epochs, and (3) the learned weights for a model of compound words with one dependency relation could be used for fine-tuning for models for compound words of other dependency relations.

## 1 Introduction

The use of distributed representations, e.g., via word2vec [6–8], has become popular in recent years. Distributed representations are vector representations of meanings, which are calculated on the basis of their contexts, and are used to investigate the similarity in meaning of two individual language units. There is much research on distributed representations of various language units, such as on a word, a phrase or a document level (see Section 2).

However, Japanese has many compound words and sometimes it makes it difficult to compare two words. For example, UniDic[1] [4], a dictionary developed by National Institute for Japanese Language and Linguistics, defined“いちご狩り, ichigo-gari, strawberry picking”as one word (short-term unit) and“ぶどう狩り, budou-gari, grape picking”as a compound word (long-term unit) [2]. In this case, a morphological analyzer using UniDic treats“いちご狩り, ichigo-gari, strawberry picking”as one word and“ぶどう狩り, budou-gari, grape picking”as two words, which makes it impossible to directly compare the word meanings of these two words via word vectors: the distributed representation of words.

[1] http://unidic.ninjal.ac.jp/(In Japanese)

[2] いちご means strawberries, ぶどう means grapes, and 狩り means picking or hunting in Japanese.

In addition, word boundaries in Japanese are unspecific because Japanese does not have word delimiters between words. Therefore, Japanese dictionary individually defines words; Japanese has different definitions of words according to each dictionary or each tagger. That is why a word for a dictionary or a tagger could be a compound word for another dictionary or another tagger. Therefore, we believe that a method to compare meanings of a word and a compound word is necessary.

In the current study, we composed a compound word vector from those of constituent word vectors. We focused on UniDic that defined two language units, the short-term unit and the long-term unit, and composed word vectors of long-term units from word vectors of two short-term units. We took into consideration the dependency relations of compound words to compose word vectors of them. Specifically, we classified the compound words by the dependency relations and separately trained models for each dependency relation (see Section 3). We utilized 13 dependency relations (described in Section 4).

The experiments revealed that we could obtain better representations for compound words when we separately learn models for each dependency relation than when we learn a model for all the relations together. In addition, the experiments showed that each model could obtain good representations with fewer epochs when we took into consideration the dependency relations (see Sections 5 and 6).

We examined other classifications of dependency relations and investigated the effectiveness of fine-tuning (described in Section 7). Finally, we conclude our work in Section 8.

## 2 Related Work

There has been much research on composing phrase representations from multiple word representations in recent years [1, 9, 2, 3]. [1, 9] used dependency relations. [9] showed that the model using different composition matrices for different dependency relations was the best. [2] proposed an implicit tensor factorization method for learning the word vectors of transitive verb phrases. [3] proposed a method for jointly learning compositional and noncompositional word vectors for phrases by adaptively weighting both types of word vectors using a compositionality scoring function.

At the same time, as described in Section 1, it is necessary to compose distributed representations of compound words. We believe that we can compose word vectors of compound words from those of constituent word vectors using the method to compose word vectors of phrases. Therefore, in the current study, we compose word vectors of Japanese compound words following a method of composing phrase vectors.

## 3 Composing Word Vectors of Japanese Compound Words Using Dependency Relation

We followed the method of [9], the method for composition of phrase vectors, to compose word vectors of compound words. We employed UniDic that defined two language units, the short-term unit and the long-term unit and composed word vectors for long-term units from two word vectors of short-term units. We only utilized the long-term units that consists of two short-term units although some long-term units consist of more than two short-term units, because we believe that these long-term units can be recursively composed using the same way to compose long-term units that consist of two short-term units.

[9] composed word vectors for phrases taking word vectors as inputs. They utilized the method proposed by [10, 11], *Relfunc*, the method that can compose a vector depending on the relation $r$ between two inputs. We used this model to compose word vectors for long-term units from two vectors of short-term units. A word vector of a long-term unit, $l$, can be composed from two word vectors of short-term units, $s_1$ and $s_2$, using the following model.

$$l = f(s_1, s_2, r) = \sigma\left(W_r \begin{bmatrix} s_{1r} \\ s_{2r} \\ b_r \end{bmatrix}\right) \tag{1}$$

where, $\sigma(.)$ denotes an element-wise sigmoid function, and $W_r \in \mathbb{R}^{d\times(2d+1)}$ and $b_r \in \mathbb{R}$ are parameters trained for each relation $r$. Therefore, the weights and biases of neural networks varies according to the dependency relations.

We employed the same loss function as [9]. It is the square errors between composed vectors and gold word vectors:

$$J(\theta) = \frac{1}{T}\sum_{i=1}^{N}\frac{1}{2}||l_i - t_i||^2 + \frac{\lambda}{2}||\theta||^2 \tag{2}$$

where, vector $l_t$ denotes a long-term-unit vector composed by Equation 1 and vector $t_i$ denotes a gold word vector for a long-term unit. $\theta$ is all the other parameters.

We classified the compound words by the dependency relations and separately trained models for each dependency relation, following [9].

## 4 Compound Words and Dependency Relations

We defined the classes of dependency relations based on the structure of Japanese compound words. First, we extracted the long-term units that consists of two short-term units from 23,000 compound words in Balanced Corpus of Contemporary Japanese (BCCWJ) [5]. They were 3,500 examples and we manually classified them. A class of dependency relation has at least 30 examples in 3,500 compound words [3]. The definitions of the classes are as follows.

[3] The relations that had less than 30 examples were stuck in 'others' class

| **Dependency** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Number of Data** | 1010 | 107 | 34 | 38 | 650 | 259 | 49 | 422 | 307 | 428 | 71 | 37 | 88 |

**Table 1.** Dependency relations and number of examples.

1. Combinations where the former short-term unit explains the latter short-term unit, c.f. “講習会, kousyu- kai, lecture-class” when “講習” means lectre and “会” means class or meating.
2. Combinations of an object and a predicate, c.f. “債務放棄, saimu-houki, debt-waiver” when “債務” means debt and “放棄” means waive, waiver or waiving,
3. Combinations of a complement and a predicate, c.f. “法的整理, houteki-seiri, legal-liquidation (liquidating)” when “法的” means legal and “整理” means liquidating,
4. Combinations of a subject and a predicate, c.f. “画面割れ, gamen-ware, screen-cracking” when “画面” means screen and “割れ” means cracking,
5. Combinations includes a unit, c.f. “1 ドル, 1 doru, 1 dollar” when “ドル” means dollar,
6. Combinations of a main word and a suffix, c.f. “具体的”, gutai-teki, concrete (combination of a word concrete (具体) and a suffix that makes an adjective verb (的)),”
7. Combinations of a prefix and a main word, c.f. “副代表, fuku-daihyou, sub-delegate” when “副” means sub and “代表” means delegate,
8. Combinations that includes a particle, c.f. “ための, tame-no, for (combination of for(ため) and a particle “no(の), ”)”
9. Combinations of a proper noun and a general noun, c.f. “茨城県, Ibaraki-ken, Ibaraki-prefecture ” when “茨城” means Ibaraki (place name) and “県” means prefecture.
10. Combinations of a noun and a verb. The combination makes a verb, c.f. “応募する, oubo-suru, apply (combination of do(する) and application(応募),)”
11. Numbers, c.f. “三二, sanjyu-ni, 32 (combination of 3(三) and 2(二),)”
12. Combinations where each short-term unit has no meaning, c.f. “だが, daga, however, ” and
13. Others, c.f. a four-character idiom like “意気揚々, iki-youyou, high-spirits.” Here, “意気” means spirits, “揚” means upraise and “々” means the same charcter as the last character, or repetation [4].

Table 1 shows the number of examples of long-term units by each dependency relation in manually-classified 3,500 examples.

We trained a model with manually-tagged examples using support vector machine (liblinear[5]). After that, SVM model classified the dependency relations of the other compound words in BCCWJ.

[4] “揚々” is the same as “揚揚.”

[5] https://www.csie.ntu.edu.tw/ cjlin/liblinear/

## 5 Experiment

We generated word vectors of long-term units and short-term units as follows.

1. Separate long-term-unit words in Japanese in a corpus with spaces.
2. Separate short-term-unit words in Japanese in a corpus with spaces.
3. Add a file of long-term-unit words to a file of short-term-unit words.
4. Generate word vectors from a file that contains long-term-unit words and short-term-unit words.

We employed BCCWJ as a corpus. We used word2vec[6] to generate word vectors. We employed a skip-gram algorithm and the vector size was set to 100 to generate the word vectors. We used default settings for other parameters. We utilized only long-term units that consists of two short-term units. Word vectors of 169,736 long-term units and 339,472 short-term units, those of constituent word vectors, were obtained in this way.

We employed a feed-forward neural network to compose word vectors for long-term units. Here, inputs are two word vectors for short-term units and an output is a word vector for long-term unit. We set maximum epoch number to 2,000 and tried three types of unit number of a hidden layer: 100, 300, and 500. The initial weights of the network were randomly generated.

We carried out two-fold cross validation, specifically, we used half of word vectors we generated as the training data and used the rest of them as the test data. We evaluated the cosine similarities between a composed word vector and a word vector for a long-term unit directly generated by the word2vec program. Here, the composed vector is better when the cosine similarity is higher.

## 6 Results

### 6.1 Unit Number of a Hidden Layer

We tried three types of unit number of a hidden layer, 100, 300, and 500, to find the best settings for the neural network to compose the word vectors for compound words. We examined the best unit number using the network without considering dependency relations, namely, the model trained by all the word vectors. Table 2 shows the result of the experiment.

Table 2 shows that the best unit number for a hidden layer is 500. However, it takes time to train a model with 500 and the training data decrease when the models were separately learned. Therefore, we used 300 units for a hidden layer when we separately learn the models.

### 6.2 Effect of Dependency Relations

We compared the composed word vectors for compound words with and without taking into consideration dependency relations. Micro-averaged cosine similarities over the whole dataset of word vectors were evaluated. Table 3 summarizes

[6] https://radimrehurek.com/gensim/

| Epochs | 100 | 300 | 500 |
|---|---|---|---|
| 200 | 0.263 | 0.284 | 0.337 |
| 400 | 0.300 | 0.328 | 0.393 |
| 600 | 0.333 | 0.387 | 0.435 |
| 800 | 0.412 | 0.425 | 0.475 |
| 1,000 | 0.433 | 0.468 | 0.501 |
| 1,200 | 0.460 | 0.491 | 0.522 |
| 1,400 | 0.476 | 0.515 | 0.538 |
| 1,600 | 0.498 | 0.528 | 0.552 |
| 1,800 | 0.510 | 0.545 | 0.562 |
| 2,000 | 0.523 | 0.555 | 0.569 |

**Table 2.** Comparison of performances by unit numbers for a hidden layer.

| Epochs | - Dependency | + Dependency |
|---|---|---|
| 200 | 0.337 | 0.415 |
| 400 | 0.393 | 0.500 |
| 600 | 0.435 | 0.548 |
| 800 | 0.475 | 0.575 |
| 1,000 | 0.501 | 0.596 |
| 1,200 | 0.522 | 0.607 |
| 1,400 | 0.538 | 0.615 |
| 1,600 | 0.552 | 0.621 |
| 1,800 | 0.562 | 0.626 |
| 2,000 | 0.569 | 0.629 |

**Table 3.** Comparison of cosine similarities of models with and without taking into consideration dependency relations.

the cosine similarities between the composed word vectors and the gold word vectors for the compound words. In this table, we compared word vectors of compound words when the dependency relations were taken into consideration and those when the dependency relations were not taken into consideration. The cosine similarities were calculated from 200 epochs to 2000 epochs. Table 3 shows that cosine similarity is 0.060 higher when the dependency relations were taken into consideration at 2,000 epochs. In addition, the difference between cosine similarities tend to become higher when the numbers of epochs are smaller. These results indicate that we can obtain the better composed word vectors with fewer epochs when we separately learn models for each dependency relation than when we learn a model for all the relations together.

### 6.3 Cosine similarities of each dependency relation

Table 4 shows the cosine similarities of each dependency relation. The table also shows the number of epochs when the best performances were obtained. This table shows that the dependency relation with the highest performance

| Dependency | Cos-Sim | Epochs |
|---|---|---|
| 1 | 0.608 | 2,000 |
| 2 | 0.656 | 1,600 |
| 3 | 0.610 | 600 |
| 4 | 0.668 | 800 |
| 5 | 0.617 | 2,000 |
| 6 | 0.630 | 2,000 |
| 7 | 0.619 | 2,000 |
| 8 | 0.620 | 400 |
| 9 | 0.619 | 2,000 |
| 10 | 0.722 | 2,000 |
| 11 | 0.666 | 600 |
| 12 | 0.470 | 200 |
| 13 | 0.632 | 2,000 |

**Table 4.** Cosine similarities of each dependency relation.

was *Dependency Relation 10*, the combination of a noun and a verb, and the dependency relation with the lowest performance was *Dependency Relation 12*, the combination where each short-term unit has no meaning. we compared the difference between the cosine similarity averaged over all the dependency relation at the best epoch and the cosine similarity averaged over all the dependency relation at 2,000 epochs. They were rarely different from each other (difference was only 0.001) because word vectors were the best at 2,000 epochs for most of dependency relations.

## 7 Discussion

### 7.1 Performances of the Models and Classification Accuracy of SVM

In the current study, the classification accuracy of SVM greatly affects the performances of the models to compose word vectors for compound words. Therefore, we evaluated the classification accuracy of SVM. We extracted 100 compound words of each dependency relation and manually checked if they were right or wrong. Table 5 shows the accuracy and most frequent errors of each dependency relation. **MFE** in the table stands for most frequent errors.

Table 5 shows the classification accuracies of *Dependency Relation 3* and *Dependency Relation 4* are much lower than other classes of dependency relations. *Dependency Relation 3* is combinations of a complement and a predicate and *Dependency Relation 4* is combinations of a subject and a predicate. The most frequent error of these classification was misclassification to *Dependency Relation 1*, combinations where the former short unit explains the latter short unit, and they are similar each other. Therefore, we combined *Dependency Relation 3* and *Dependency Relation 4* to *Dependency Relation 1*, refer to it as *Dependency Relation 1'*, and learned the models again. Table 6 summarizes the results

| Dependency | MFE | Accuracy |
|---|---|---|
| 1 | 6 | 84 |
| 2 | 1 | 42 |
| 3 | 1 | 8 |
| 4 | 1 | 16 |
| 5 | 6 | 99 |
| 6 | 13 | 91 |
| 7 | 1,6,13 | 94 |
| 8 | 1,6 | 85 |
| 9 | 1 | 91 |
| 10 | 1 | 99 |
| 11 | 1 | 86 |
| 12 | 1 | 86 |
| 13 | 1 | 84 |

**Table 5.** Accuracy and most frequent errors of each dependency relation.

| Epochs | Whole | Dependency 1' |
|---|---|---|
| 200 | 0.402 | 0.327 |
| 400 | 0.489 | 0.415 |
| 600 | 0.541 | 0.480 |
| 800 | 0.571 | 0.521 |
| 1,000 | 0.590 | 0.547 |
| 1,200 | 0.603 | 0.567 |
| 1,400 | 0.613 | 0.583 |
| 1,600 | 0.620 | 0.594 |
| 1,800 | 0.624 | 0.602 |
| 2,000 | 0.629 | 0.609 |

**Table 6.** Experiment with Dependency 1'

of the experiment. **Whole** and **Dependency 1'** in the table indicate the cosine similarities of the whole dataset and the class of *Dependency Relation 1'*, respectively.

When **+Dependency** in Table 3 and **Whole** in Table 6 are compared, the results of original setting, **+Dependency** in Table 3, was slightly better. We think this is because the cosine similarities for *Dependency Relation 3* and *Dependency Relation 4* were higher than *Dependency Relation 1*. The examples of *Dependency Relation 3* and *Dependency Relation 4* did not contribute the cosine similarities of *Dependency Relation 1* because their examples were much fewer than *Dependency Relation 1*. In addition, the classification accuracies of *Dependency Relation 3* and *Dependency Relation 4* were low because their examples were few and they misclassified to *Dependency Relation 1* because it was the class with the most examples.

| Epochs | Whole | Dependency 13' |
|---|---|---|
| 200 | 0.415 | 0.497 |
| 400 | 0.500 | 0.570 |
| 600 | 0.548 | 0.599 |
| 800 | 0.575 | 0.616 |
| 1,000 | 0.596 | 0.622 |
| 1,200 | 0.608 | 0.627 |
| 1,400 | 0.616 | 0.630 |
| 1,600 | 0.622 | 0.622 |
| 1,800 | 0.627 | 0.632 |
| 2,000 | 0.630 | 0.630 |

**Table 7.** Experiment with Dependency 13'

### 7.2 Error Analysis

The class of dependency relation with the lowest performance was *Dependency Relation 12*, the combination where each short-term unit has no meaning. The properties of *Dependency Relation 12* are as follows.

1. It has fewer examples.
2. It contains particles and verbal auxiliaries, which are not usually contained in other classes, as short-term units that constitutes long-term units.

To address a problem caused from the first property, we combined *Dependency Relation 12* to *Dependency Relation 13*, refer to it as *Dependency Relation 13'*, and learned the models again. Table 7 summarizes the results of the experiment. **Whole** and **Dependency 13'** in the table indicate the cosine similarities of the whole dataset and the class of *Dependency Relation 13'*, respectively.

When **+Dependency** in Table 3 and **Whole** in Table 7 are compared, the results of new setting, **Whole** in Table 7, was slightly better. However, the difference was very small because the number of examples of *Dependency Relation 12* was quite few.

We believe that the second property is the main reason of the low performance of the class of *Dependency Relation 12*. We conduct the composition assuming that the meanings of compound words could be composed from those of constituent words. Therefore, we believe that the composition mechanism does not work if the constituent words had no meanings.

### 7.3 Fine-tuning of Models

Fine-tuning is a method to improve the performance of one task by re-learning using the weights of a model learned for another task as initial values. It is useful to speed up the learning of the target task and is effective when the data size of the target task is small. In the current study, we investigated two fine-tuning models, fine-tuning using the weights of the model of *Dependency Relation 1* and

| Epochs | Cos-Sim | Difference from Random Initial Values |
|---|---|---|
| 200 | 0.491 | +0.080 |
| 400 | 0.541 | +0.041 |
| 600 | 0.573 | +0.025 |
| 800 | 0.591 | +0.017 |
| 1,000 | 0.606 | +0.009 |
| 1,200 | 0.612 | +0.005 |
| 1,400 | 0.620 | +0.005 |
| 1,600 | 0.624 | +0.002 |
| 1,800 | 0.628 | +0.001 |
| 2,000 | 0.630 | +0.001 |

**Table 8.** Fine-tuning using weights of the model of *Dependency Relation 1*.

| Epochs | Cos-Sim | Difference from Random Initial Values |
|---|---|---|
| 200 | 0.609 | +0.193 |
| 400 | 0.623 | +0.123 |
| 600 | 0.630 | +0.082 |
| 800 | 0.635 | +0.060 |
| 1,000 | 0.637 | +0.041 |
| 1,200 | 0.639 | +0.032 |
| 1,400 | 0.641 | +0.025 |
| 1,600 | 0.642 | +0.020 |
| 1,800 | 0.643 | +0.016 |
| 2,000 | 0.643 | +0.014 |

**Table 9.** Fine-tuning using weights of the model of *Dependency Relation 13*.

that using the weights of the model of *Dependency Relation 13*. We selected them because *Dependency Relation 1* had the most examples in all the relations and we believe that *Dependency Relation 13* is independent from the other classes because it contains "other" dependency relations.

Tables 8 and 9 shows that cosine similarities improved at all epochs when the models were fine-tuned. In addition, the cosine similarities are relatively high with fewer epochs. The performances of the models with fine-tuning converged at approximately 2,000 epochs, which is almost identical to the model without fine-tuning.

When Tables 8 and 9 are compared, fine-tuning with weights of the model of *Dependency Relation 13* is better than *Dependency Relation 1*. When Tables 10 and 11 are compared, fine-tuning with weights of the model of *Dependency Relation 13* tends to obtain better results for the class of word vectors for compound words with dependency relations with more examples. We think that fine-tuning with weights of the model of *Dependency Relation 13* is better because it could improve the result for the class of word vectors for compound words with *Dependency Relation 1*, which is the class with the most examples.

| Dependency | Cos-Sim | Epochs | Difference from Table 4 |
|---|---|---|---|
| 1 | 0.608 | 2000 | ±0 |
| 2 | 0.659 | 600 | +0.003 |
| 3 | 0.626 | 200 | +0.061 |
| 4 | 0.674 | 400 | +0.014 |
| 5 | 0.619 | 1,400 | +0.002 |
| 6 | 0.637 | 2,000 | +0.007 |
| 7 | 0.621 | 1,000 | +0.002 |
| 8 | 0.646 | 200 | +0.026 |
| 9 | 0.624 | 2,000 | +0.005 |
| 10 | 0.727 | 2,000 | +0.005 |
| 11 | 0.680 | 200 | +0.014 |
| 12 | 0.506 | 200 | +0.034 |
| 13 | 0.634 | 1,200 | +0.002 |

**Table 10.** Cosine similarities of each dependency relation with fine-tuning using weights of the model of *Dependency Relation 1*.

| Dependency | Cos-Sim | Epochs | Difference from Table 4 |
|---|---|---|---|
| 1 | 0.635 | 2,000 | +0.026 |
| 2 | 0.661 | 600 | +0.005 |
| 3 | 0.630 | 200 | +0.019 |
| 4 | 0.678 | 400 | +0.009 |
| 5 | 0.620 | 1,400 | +0.003 |
| 6 | 0.638 | 2,000 | +0.008 |
| 7 | 0.622 | 800 | +0.003 |
| 8 | 0.650 | 200 | +0.034 |
| 9 | 0.728 | 2,000 | +0.007 |
| 10 | 0.682 | 2,000 | +0.005 |
| 11 | 0.524 | 200 | +0.017 |
| 12 | 0.632 | 200 | +0.051 |

**Table 11.** Cosine similarities of each dependency relation with fine-tuning using weights of the model of *Dependency Relation 13*.

## 8 Conclusions

In this paper, we composed compound word vectors from those of constituent word vectors. We took into consideration the dependency relations of compound words and separately learn models for each dependency relation to compose word vectors of compound words. The experiments revealed that, when we took into consideration dependency relations, we could obtain better representations for compound words with fewer epochs, and the learned weights for a model of compound words with one dependency relation could be used for fine-tuning for models for compound words of other dependency relations. Error analysis of the class of the dependency relation with the lowest performance indicated

that the composition mechanism does not work if the constituent words had no meanings because we conducted the composition assuming that the meanings of compound words could be composed from those of constituent words.

## Acknowledgments

## References


1. Baroni, M., Zamparelli, R.: Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In: Proceedings of EMNLP 2010. pp. 1183–1193 (2010)
2. Hashimoto, K., Tsuruoka, Y.: Learning embeddings for transitive verb disambiguation by implicit tensor factorization. In: Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality. pp. 1–11 (2015)
3. Hashimoto, K., Tsuruoka, Y.: Adaptive joint learning of compositional and non-compositional phrase embeddings. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 205–215. Association for Computational Linguistics (2016), http://arxiv.org/abs/1603.06067
4. Maekawa, K., Yamazaki, M., Maruyama, T., Yamaguchi, M., Ogura, H., Kashino, W., Ogiso, T., Koiso, H., Den, Y.: Design, Compilation, and Preliminary Analyses of Balanced Corpus of Contemporary Written Japanese. In: Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC 2010). pp. 1483–1486 (2010)
5. Maekawa, K., Yamazaki, M., Ogiso, T., Maruyama, T., Ogura, H., Kashino, W., Koiso, H., Yamaguchi, M., Tanaka, M., Den, Y.: Balanced Corpus of Contemporary Written Japanese. In: Language Resources and Evaluation. vol. 48, pp. 345–371 (2014)
6. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Proceedings of ICLR Workshop 2013. pp. 1–12 (2013)
7. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of NIPS 2013. pp. 1–9 (2013)
8. Mikolov, T., tau Yih, W., Zweig, G.: Linguistic regularities in continuous space word representations. In: Proceedings of NAACL 2013. pp. 746–751 (2013)
9. Muraoka, M., Shimaoka, S., Yamamoto, K., Watanabe, Y., Okazaki, N., Inui, K.: Finding The Best Model Among Representative Compositional Models. In: Proceedings of the 28th Pacific Asia Conference on Language, Information and Computation (PACLIC 2014). pp. 65–74 (2014)
10. Socher, R., Bauer, J., Manning, C.D., Ng, A.Y.: Parsing with compositional vector grammars. In: Proceedings of ACL 2013. pp. 455–465 (2013)
11. Socher, R., Karpathy, A., Le, Q.V., Manning, C.D., Ng, A.Y.: Grounded compositional semantics for finding and describing images with sentences. Transactions of the Association for Computational Linguistics 2, 207–218 (2014)