Extending and Evaluating a Multiplicative Model for Semantic Composition in a Distributional Semantic Model

Akira Utsumi (utsumi@inf.uec.ac.jp)
Department of Informatics, The University of Electro-Communications
1-5-1, Chofugaoka, Chofushi, Tokyo 182-8585, Japan

Abstract

This paper addresses a multiplicative model for semantic composition in a distributional semantic model. This paper proposes new composition algorithms using a multiplicative model by two approaches: to extend a multiplicative model by averaging or weighting and to modify context-sensitive additive models such as Kintsch's (2001) predication by replacing vector addition with vector multiplication. In addition, this paper examines conditions for the superiority of the multiplicative models found in previous research by comparing two semantic spaces constructed by latent semantic analysis (LSA) and positive pointwise mutual information (PPMI) in terms of the representational ability of composition algorithms. The experiment using noun compounds demonstrated that the multiplicative model performed better than the additive model only in the PPMI-based space, suggesting that component-wise multiplication works effectively in a semantic space whose dimensions represent distinctive features. Some multiplicative modifications of additive algorithms also improved the performance in the PPMI-based space, but did not in the LSA-based space. Interestingly, however, the extension of the multiplicative model by weighting was effective in improvement of the performance in the LSA-based spaces, although the extensions of the multiplicative model were not effective in the PPMI-based space.

Keywords: Distributional semantic models; Semantic spaces; Vector composition; Component-wise multiplication; Latent semantic analysis; Pointwise mutual information

Introduction

Recently there is a growing interest in vector-based semantic space models or distributional semantic models in the field of cognitive science (e.g., Landauer, McNamara, Dennis, & Kintsch, 2007; McNamara, 2011) as well as in natural language processing (e.g., Padó & Lapata, 2007; Turney & Pantel, 2010). The primary reason for the growing interest is that despite its simplicity the semantic space model has demonstrated high performance of cognitive modeling for a number of cognitive tasks such as similarity judgment (Landauer & Dumais, 1997), semantic priming (Jones, Kintsch, & Mewhort, 2006), and metaphor comprehension (Utsumi, 2011). In particular, the ability of the semantic space model to represent the meanings of single words has been extensively examined in a number of studies. However, less attention has been paid to the problem of how to construct the semantic representations of larger linguistic units such as phrases and sentences. Given the creativity of human language that an infinite number of phrases or sentences can be constructed from a finite number of words, distributional semantic models should provide appropriate methods for constructing (or composing) vectors for phrases or sentences from the vectors of their constituent words. This paper addresses this problem.

In the previous studies on vector composition (e.g., Baroni & Zamparelli, 2010; Mitchell & Lapata, 2008, 2010; Zanzotto et al., 2010), two classes of composition methods, namely *an additive model* and *a multiplicative model*, have been proposed and evaluated in a number of tasks. The widely-used algorithm in the additive class is to compute the centroid of constituent word vectors. Although the centroid method simply combines the contents of all the constituent words involved in the target phrase or sentence, more sophisticated algorithms in the additive class such as predication (Kintsch, 2001) and comparison (Utsumi, 2011) use additional information (i.e., words that are semantically similar or related to the constituent words) to compute contextually dependent meanings. These additive algorithms have been shown to be effective in a number of applications of a distributional semantic model (e.g., Kintsch, 2001; Landauer et al., 2007). However, recent studies (Mitchell & Lapata, 2008, 2010) have demonstrated that component-wise multiplication of multiple vectors, one representative method of the multiplicative class model, performs better than the centroid algorithm and other additive methods. This finding is particularly interesting because until recently the centroid algorithm and other additive models have been widely accepted as an effective method for vector composition. Hence, this paper is concerned with the multiplicative model and tackles two issues concerning the multiplicative model.

One issue addressed in this paper is why and when a multiplicative model performs better than an additive model. As mentioned above, some studies have found the superiority of the multiplicative model in general and component-wise multiplication in particular, but no studies have revealed the rationale behind, and conditions for, the superiority of the multiplicative model. Our working hypothesis is that, unless dimensions of a semantic space represent distinctive features or meanings, component-wise multiplication does not work effectively, because its function is to highlight dimensions relevant to both vectors and downplay irrelevant dimensions. This paper examines the validity of this hypothesis using (at least) two different types of semantic spaces whose dimensions represent or do not represent distinctive features.

Another important issue of interest is to propose new composition algorithms on the basis of the superiority of component-wise multiplication. Two approaches are considered to devise a method for semantic composition. One approach is that we can extend the component-wise multiplication algorithm by averaging or weighting. The purpose of this modification is to adjust the degree of highlighting relevant dimensions to a more appropriate level. Weighting is also aimed at involving the information of word order in a composition vector. Another approach is to modify the additive model using a multiplicative function. For example, the predication algorithm (Kintsch, 2001) generates the composition vector of a two-word phrase by computing the centroid of the both words and some semantic neighbors of a predicate word that are also related to an argument word. If the multiplicative model generally performs better than the additive model, it naturally follows that we can obtain new and possibly efficient algorithms by replacing vector addition in the predication algorithm with component-wise multiplication. In this paper, we propose new algorithms for vector composition by adopting these approaches, and compare the representational ability of these algorithms with the original ones.

In order to evaluate the ability of new composition algorithms, we conducted an experiment using two-noun compounds. In the experiment, the vector for noun compounds was computed by the composition algorithms, and its representational ability was evaluated using the plausibility of the semantic relatedness or similarity computed between vectors for noun compounds and semantically related words. The plausibility of the similarity was evaluated by two measures: correlation between the computed cosine similarity and human similarity judgment for compound-word pairs, and ranking of the similarity of compound-word pairs. Furthermore, we also examined the hypothesis that multiplicative models perform differently depending on the types of semantic spaces using two different semantic spaces. The experiment was conducted in two languages, i.e., English and Japanese, to test whether different languages show consistent results.

Composition Algorithm

In this section, we review existing methods for vector composition and propose new methods by extending or modifying the existing methods. Throughout this section, we explain the methods by assuming that we are generating the vector $\boldsymbol{v}$ for a two-word phrase or sentence $P(A)$ that comprises the argument (or the head) A (whose vector is denoted by $\boldsymbol{a}$) and the predicate (or the modifier) P (whose vector is denoted by $\boldsymbol{p}$). For example, a vector for the sentence “A horse runs” is computed from the vector for the argument “horse” and the vector for the predicate “run.”

Reviewing the existing composition algorithms

The simplest but widely used additive method for vector composition is to compute the *centroid* of constituent word vectors i.e., $\boldsymbol{v} = (\boldsymbol{p}+\boldsymbol{a})/2$. Note that, when the cosine is used as a similarity measure, additive models yield the same result of similarity computation, whether vectors are averaged or simply summed. Hence, the simple addition (i.e., $\boldsymbol{v} = \boldsymbol{a} + \boldsymbol{p}$) suffices as the centroid algorithm.

A more sophisticated method, namely *dilation*, in the additive class is proposed by Mitchell and Lapata (2010); this method stretches or “dilates” the argument (or head) vector along the direction of the predicate (or modifier) vector. This can be done by decomposing the argument vector $\boldsymbol{a}$ into two orthogonal vectors, i.e., a vector $\boldsymbol{x}$ parallel to $\boldsymbol{p}$ and a vector $\boldsymbol{y}$ orthogonal to $\boldsymbol{p}$, and then stretching the parallel component $\boldsymbol{x}$ so that a modified vector $\boldsymbol{v}$ of $\boldsymbol{a}$ is more like $\boldsymbol{p}$.

$$\begin{aligned} \boldsymbol{v} &= \lambda \boldsymbol{x} + \boldsymbol{y} = \lambda \frac{\boldsymbol{p}\cdot\boldsymbol{a}}{\boldsymbol{p}\cdot\boldsymbol{p}}\boldsymbol{p} + \left(\boldsymbol{a} - \frac{\boldsymbol{p}\cdot\boldsymbol{a}}{\boldsymbol{p}\cdot\boldsymbol{p}}\boldsymbol{p}\right) \\ &= (\lambda - 1)\frac{\boldsymbol{p}\cdot\boldsymbol{a}}{\boldsymbol{p}\cdot\boldsymbol{p}}\boldsymbol{p} + \boldsymbol{a} \qquad (1) \end{aligned}$$

In Equation 1, λ is a parameter that represents the degree of dilation. When $\lambda = 1$, the resulting vector $\boldsymbol{v}$ is identical to the argument vector $\boldsymbol{a}$. Mitchell and Lapata (2010) demonstrated that the dilation algorithm performs consistently well on a phrase similarity task.

The *predication* algorithm proposed by Kintsch (2001) does not simply combine the constituent vectors, but embodies the semantic interaction between constituent words on the basis of his construction-integration model. The predication algorithm makes use of semantic neighbors (i.e., words similar to the constituent words) to embody the interaction; it first chooses m nearest neighbors of a predicate P (i.e., m words with the highest cosine to P) and then picks up k neighbors of P that are also related to A. Finally the algorithm computes the centroid vector of P, A, and the k neighbors of P as a vector representation of $P(A)$. Formally, the predication algorithm computes a composition vector $\boldsymbol{v}$ by

$$\boldsymbol{v} = \boldsymbol{p} + \boldsymbol{a} + \sum_{\boldsymbol{q_i} \in Q} \boldsymbol{q_i}, \qquad (2)$$

where Q denotes a set of the k neighbors of P that are also related to A.

Utsumi (2011) proposes a different algorithm for vector composition, namely, *comparison*, to compute the meanings of metaphors. This algorithm uses a set of common neighbors of two constituent words to capture the relevance between them. It is motivated by the fact that the interaction between constituent words embodied in the predication algorithm is rather predicate-directed. Formally, the comparison algorithm computes a compound vector $\boldsymbol{p}$ using the

$$\boldsymbol{v} = \boldsymbol{a} + \sum_{\boldsymbol{c_i} \in C} \boldsymbol{c_i}, \qquad (3)$$

where C denotes a set of k common neighbors of A and P. The set C of common neighbors can be obtained by finding the smallest i that satisfies $|N_i(A) \cap N_i(P)| \geq k$, where $N_i(w_j)$ denotes a set of the top i neighbors of the word w_j.

Turning now to the multiplicative model, the simplest method is to use a function of *component-wise multiplication* for vector composition:

$$\begin{aligned} \boldsymbol{v} &= \boldsymbol{p} \odot \boldsymbol{a} \qquad (4) \\ v_i &= p_i \cdot a_i \qquad (5) \end{aligned}$$

where the symbol $\odot$ expresses that two vectors are multiplied component-wise.

Circular convolution is also a member of the multiplicative class, and defined as

$$\begin{aligned} \boldsymbol{v} &= \boldsymbol{p} \circledast \boldsymbol{a} \qquad (6) \\ v_i &= \sum_{j=0}^{n-1} p_{j \bmod n} \cdot a_{(i-j) \bmod n}. \qquad (7) \end{aligned}$$

Proposing new composition algorithms

Extension of component-wise multiplication This approach extends the component-wise multiplication algorithm in Equation 4 in two ways: averaging and weighting.

$$\begin{aligned} v_i &= \sqrt{p_i \cdot a_i} \qquad (8) \\ v_i &= p_i^{\alpha} \cdot a_i \qquad (9) \end{aligned}$$

In the averaging approach expressed in Equation 8, each component of a composed vector is calculated as a geometric mean, rather than a simple multiplication. This extended algorithm can be regarded as corresponding to the centroid algorithm in the additive class. On the other hand, in the weighting approach in Equation 9, the predicate vector is weighted by the factor α. When $0 < \alpha < 1$, the argument vector has a stronger influence on the resulting composition vector, while the predicate vector has when $\alpha > 1$. This modification can be regarded as a multiplicative version of the dilation algorithm.

Modification of context-sensitive additive models using a multiplicative model This approach modifies the context-sensitive additive models, namely the predication and comparison algorithms, to benefit from the multiplicative model.

One way of modifying these additive models is to replace the vector addition ($+$) with the vector multiplication ($\odot$). For example, the predication algorithm can be modified as

$$\boldsymbol{v} = \boldsymbol{p} \odot \boldsymbol{a} \odot \prod_{\boldsymbol{q_i} \in Q} \boldsymbol{q_i} \tag{10}$$

$$\boldsymbol{v} = \sqrt[(k+2)]{\boldsymbol{p} \odot \boldsymbol{a} \odot \prod_{\boldsymbol{q_i} \in Q} \boldsymbol{q_i}}. \tag{11}$$

Equation 10 shows a simple multiplicative version of the predication algorithm, in which all the argument, predicate, and neighbor vectors are combined by component-wise multiplication. On the other hand, Equation 11 is an averaged version of the multiplicative predication algorithm, in which the argument, predicate, and neighbor vectors are geometrically averaged. (Note that in these equations the product symbol denotes the product by component-wise multiplication, and the radical symbol denotes the component-wise root.)

Another way of modifying the additive models is to keep the sum of neighbor vectors unchanged, and multiply the argument, predicate, and the sum of neighbor vectors.

$$\boldsymbol{v} = \boldsymbol{p} \odot \boldsymbol{a} \odot \sum_{\boldsymbol{q_i} \in Q} \boldsymbol{q_i} \tag{12}$$

$$\boldsymbol{v} = \sqrt[3]{\boldsymbol{p} \odot \boldsymbol{a} \odot \sum_{\boldsymbol{q_i} \in Q} \boldsymbol{q_i}} \tag{13}$$

This modification is motivated by the assumption that the contextually dependent meaning of a predicate should be computed by the disjunction of neighbors rather than by the conjunction of neighbors. For example, according to this assumption, the meaning of the predicate *run* in "A horse runs" should be represented as "move, flee, **or** walk," rather than "move, flee, **and** walk"

The comparison algorithm can be modified by the same approaches embodied in Equations 10–13. (These modifications are listed in Table 1.)

Method

Materials

Compound-word pairs Noun compounds we used in the experiment comprised two nouns (i.e., head and modifier) such as "apple pie" or "ruling class." These compounds included two types: *familiar noun compounds* that occur in the corpus from which semantic spaces were constructed and *novel noun compounds* that do not occur in the corpus. The selection criterion for compounds was that familiar compounds should occur at least 20 times in the corpus, and both types of compounds should be included in the thesaurus. From the compounds that satisfied this criterion, we randomly selected 50 familiar and 50 novel compounds in both languages for evaluation. For each of these noun compounds, a semantically related word was selected randomly from synonyms, hypernyms and coordinate words of that compound.

In this experiment, the written and non-fiction parts of the British National Corpus of the size of 54.7 million words and Japanese newspaper corpora (i.e., four years' worth of Mainichi newspaper articles and two year's worth of Nikkei newspaper articles) of the size of 26.2 million words were used as a corpus. They contained 73,422 English and 63,875 Japanese different words. The thesauri used in this experiment were English WordNet 3.0 and a Japanese thesaurus "Nihongo Dai-Thesaurus."

Semantic space In order to test our working hypothesis that component-wise multiplication works effectively in the semantic space whose dimensions represent distinctive features, we used two different semantic spaces.

One semantic space is based on the word-word matrix whose elements are word cooccurrence frequencies within a context window spanning some number of words. This model is a popular semantic space (e.g., Bullinaria & Levy, 2007; Recchia & Jones, 2009) and it was also used by Mitchell and Lapata's (2008, 2010) study demonstrating the superiority of the component-wise multiplication. Formally, the element a_{ij} of the word-word matrix is initially the number of times the word w_j occurs within n words around the word w_i, and weighted by positive pointwise mutual information (PPMI; Bullinaria & Levy, 2007; Turney & Pantel, 2010). In this study, we used a context window of five words, i.e., $n = 5$.

Another semantic space was constructed using latent semantic analysis (LSA; Landauer & Dumais, 1997; Landauer et al., 2007). LSA is based on the word-document matrix whose element a_{ij} is the number of times the word w_i occurs in the j-th document. The elements of this initial matrix are weighted, and the matrix is smoothed by singular value decomposition (SVD). Among a number of weighting schemes that have been proposed so far, we used Quesada's (2007) scheme in which the initial word frequency is weighted by the product of its logarithm and the entropy. The number of the reduced dimensions was determined to be 300.

One essential difference between these two semantic spaces lies in the meaningfulness of vector dimensions. In the PPMI-based semantic space, each vector component represents a distinctive feature, namely, a context word, while the dimensions of the LSA-based semantic space do not have such the clear meaning. Hence, by comparing these two spaces in terms of the performance of the multiplicative models, we tested the working hypothesis that component-wise multiplication works well in a semantic space with semantically meaningful dimensions. In addition, to test whether the smoothing by SVD is a main cause of the loss of dimension semantics, we also examined the performance of the PPMI-

Table 1: Composition algorithms compared in the experiment

Algorithm (abbr.)	Function
Centroid (CENT)	$v_i = p_i + a_i$
Dilation (DILA)	$v_i = (\lambda - 1) p_i \sum_j p_j a_j + a_i \sum_j p_j p_j$
Predication (PRED)	$v_i = p_i + a_i + \sum_j q_{ji}$
Multiplicative (+MUL)	$v_i = p_i \cdot a_i \cdot \prod_j q_{ji}$
Geometrically averaged (+AVE)	$v_i = \sqrt[(k+2)]{p_i \cdot a_i \cdot \prod_j q_{ji}}$
Partially multiplicative (+PARTMUL)	$v_i = p_i \cdot a_i \cdot \sum_j q_{ji}$
Partially averaged (+PARTAVE)	$v_i = \sqrt[3]{p_i \cdot a_i \cdot \sum_j q_{ji}}$
Comparison (COMP)	$v_i = a_i + \sum_j c_{ji}$
Multiplicative (+MUL)	$v_i = a_i \cdot \prod_j c_{ji}$
Geometrically averaged (+AVE)	$v_i = \sqrt[(k+1)]{a_i \cdot \prod_j c_{ji}}$
Partially multiplicative (+PARTMUL)	$v_i = a_i \cdot \sum_j c_{ji}$
Partially averaged (+PARTAVE)	$v_i = \sqrt{a_i \cdot \sum_j c_{ji}}$
Multiplication (MULT)	$v_i = p_i \cdot a_i$
Averaged (+AVE)	$v_i = \sqrt{p_i \cdot a_i}$
Weighted (+WEI)	$v_i = p_i^{\alpha} \cdot a_i$
Convolution (CONV)	$v_i = \sum_{j=0}^{n-1} p_{j \bmod n} \cdot a_{(i-j) \bmod n}$
Head only (HEAD)	$v_i = a_i$
Vector (VECT)	$\boldsymbol{v}$ is computed directly from the corpus by treating compounds as single words

Table 2: Correlation coefficients between human similarity ratings and the cosine similarity computed by the composition algorithms

	PPMI		LSA		PPMI+SVD	
Algorithm	Fam	Nov	Fam	Nov	Fam	Nov
CENT	.376	.381	.367	.319	.446	.466
DILA	.359	.348	.443	.405	.445	.422
PRED	.410	.343	.378	.264	.433	.422
+MUL	.388	.446	.446	.310	.322	.094
+AVE	.331	.412	.474	.213	.408	.321
+PARTMUL	.340	.432	.446	.308	.346	.108
+PARTAVE	.336	.407	.441	.218	.403	.345
COMP	.360	.331	.346	.403	.432	.410
+MUL	.219	.397	.296	.413	.442	.127
+AVE	.247	.333	.325	.239	.461	.348
+PARTMUL	.245	.355	-.238	-.079	.057	.050
+PARTAVE	.262	.314	-.249	-.218	.051	.122
MULT	.393	.414	-.149	.005	.037	.002
+AVE	.369	.413	-.194	-.086	.000	.056
+WEI	.392	.409	.281	.437	.483	.310
CONV	.294	.123	.216	.347	.418	.386
HEAD	.287	.260	.361	.411	.383	.365
VECT	.331	—	.416	—	.404	—

▨ $p < .05$, ▨ $p < .01$, 🟥 $p < .001$
Fam=Familiar compounds, Nov=Novel compounds.

based semantic space smoothed by SVD.

Human similarity judgment

In order to collect the data on human similarity judgment, we conducted an experiment using Japanese compound-word pairs. (English compound-word pairs were not used for the experiment, because a sufficient number of native English speakers could not be recruited.) Fourteen participants, who were all native speakers of Japanese, were assigned all the 100 Japanese compound-word pairs and asked to rate the semantic relatedness between the compound and the word of each pair. These pairs were rated on a 7-point scale ranging from 1 (*unrelated*) to 7 (*related*). The presentation order of those pairs was randomized for each participant.

Procedure

Given a semantic space and a set P of compound-word pairs, the compound vector of each compound was computed by the composition algorithms. Afterward, the similarity between the compound and the paired word was calculated as the cosine between the computed compound vector and the vector for the paired word. Using these cosine values for the set of compound-word pairs, we evaluated the composition algorithms in the following two ways.

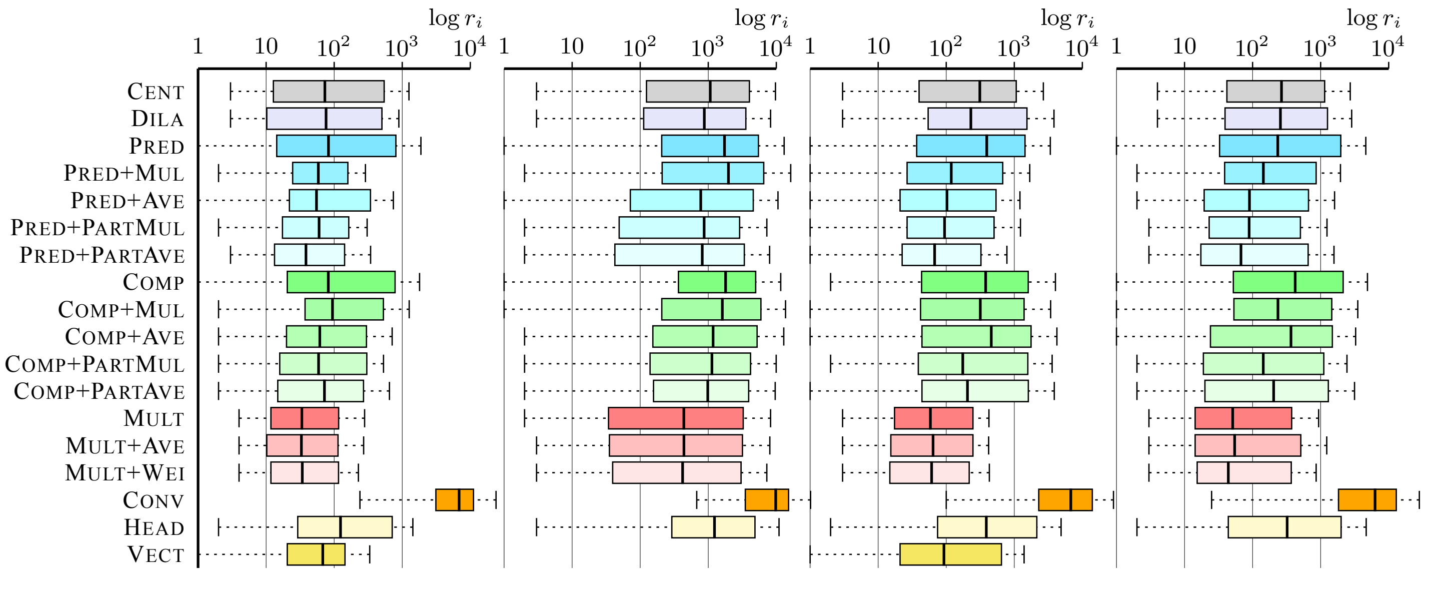
Correlation analysis Spearman's correlation coefficient was calculated between the computed cosine values and the mean human ratings for compound-word pairs.

Word ranking For each compound-word pair (c_i, w_i), the rank r_i of the paired word w_i was assessed by computing the cosine similarity between c_i and all words (including w_i) in the space, and sorting all words in descending order of the cosine. A higher rank implies that the word w_i is semantically more related to the compound c_i. Next, all compound-word pairs in the set P were sorted in ascending order of the rank r_i. The sorted list of the rank r_i is denoted as $r'_1, \cdots, r'_{|P|}$. Finally, the overall performance of each algorithm was measured by the median rank $\simeq r'_{0.5|P|}$ and first quartile rank (i.e., 25th percentile) $\simeq r'_{0.25|P|}$ of the sorted list.

Result

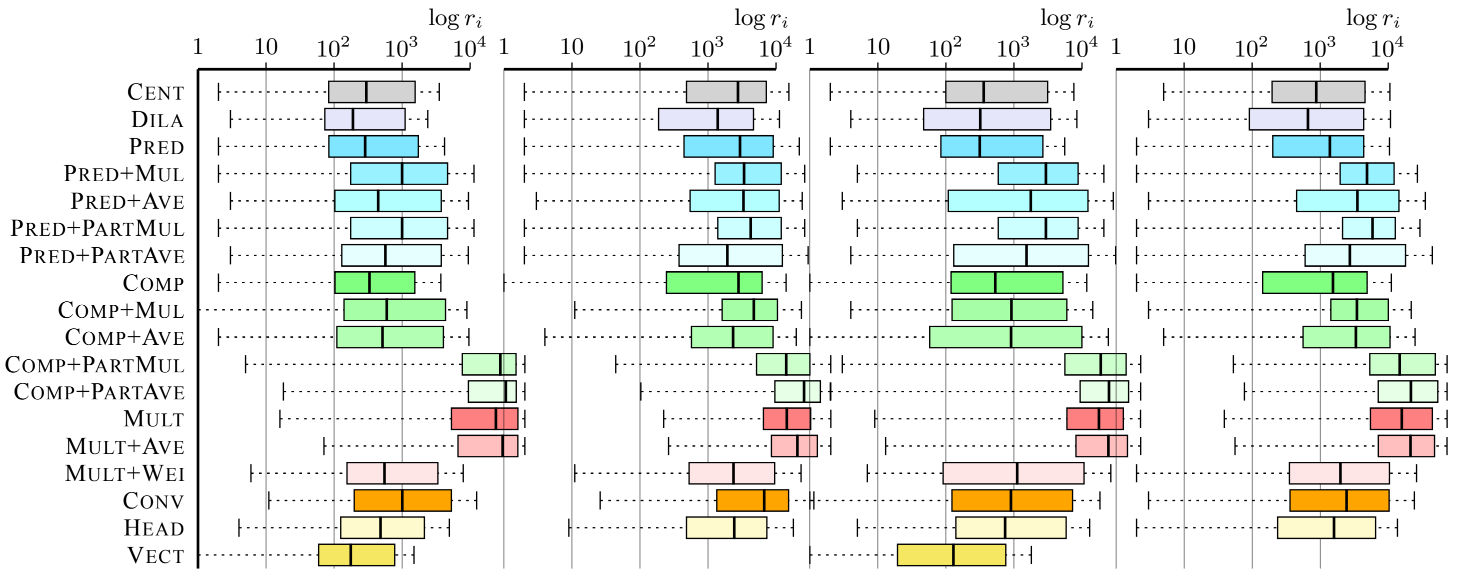
In order to compute the performance of the composition algorithms with free parameters, we estimated the optimal parameter values using a leave-one-out cross-validation procedure. The cosine similarity and the rank of each compound-word pair was calculated, with the optimal parameters estimated using all the remaining pairs as training data. The parameter m (for PRED) was optimized over integers ranging between 1 and 50 and between 100 and 500 in steps of 50, the parameter k (for PRED and COMP) over integers ranging between 1 and 10, the parameter λ (for DILA) over real numbers ranging between 1.1 and 10.0 in steps of 0.1, and the parameter α (for MULT+WEI) over real numbers ranging between 0.01 and 1.00 in steps of 0.01. The composition algorithms we compared in the experiment are listed in Table 1 for reference. Note that two non-compositional methods (i.e., HEAD and VECT) are considered for purpose of comparison.

Table 2 shows correlation coefficients between human similarity rating and cosine similarity computed in the three semantic spaces. Concerning the superiority between the additive model and the multiplicative model, the result is almost consistent with our hypothesis. In the PPMI-based space, the multiplication algorithm (MULT) performs better than the additive models (i.e., CENT, DILA, PRED, and



(a) Japanese familiar (b) Japanese novel (c) English familiar (d) English novel

Figure 1: Boxplots of the rank of cosine similarity for compound-word pairs computed in the PPMI-based semantic space



(a) Japanese familiar (b) Japanese novel (c) English familiar (d) English novel

Figure 2: Boxplots of the rank of cosine similarity for compound-word pairs computed in the LSA-based semantic space

COMP), although for familiar compounds it performs worse than the predication algorithm. On the other hand, in the LSA-based semantic space, the multiplication algorithm does not achieve a significant correlation with human judgment, while the additive algorithms are significantly correlated with human judgment. In addition, the smoothed PPMI space (i.e., PPMI+SVD) shows the same tendency, thus suggesting that smoothing by SVD disables the function expected by the component-wise multiplication regardless of the initial matrix and weighting function.

Concerning the proposed algorithms, the extended multiplication algorithms (i.e., MULT+AVE, MULT+WEI) cannot improve the performance in the PPMI-based space, but the weighted multiplication (MULT+WEI) considerably improves the performance for the semantic space without dimension semantics (i.e., LSA and PPMI+SVD). This surprising finding suggests that moderate weighting enables the multiplicative model to work in these spaces.

The predication and comparison algorithms are improved by some modification methods, especially by multiplying all the vectors (i.e., +MUL), when novel compounds are considered. Furthermore, some modifications also improve the performance for the semantic spaces whose dimensions do not represent distinctive features; the predication algorithm performs best for familiar compounds in the LSA-based space by geometrically averaging all the vectors (i.e., +AVE), and the averaged comparison algorithm (COMP+AVE) performs nearly best for familiar compounds in the smoothed PPMI space. Simple multiplication (+MUL) also improves the performance, but the partial extensions (i.e., +PARTMUL, +PARTAVE) seem not to improve the performance.

Figures 1 and 2 respectively show the results of the similarity ranking of compound-word pairs for the PPMI-based and LSA-based spaces. The result is almost consistent with the result of the correlation analysis. In the PPMI-based space, the multiplication algorithm (MULT) performs better than the additive models, while in the LSA-based spaces it performs much worse than the additive models. The extensions of the multiplication algorithm do not improve the performance in the PPMI-based space, but the weighted multiplication considerably improves the performance in the LSA-based space. The result of the multiplicative modifications (i.e., +MUL, +AVE, +PARTMUL, +PARTAVE) for the predication and comparison algorithms slightly differs between the word ranking test and the correlation analysis. In the word ranking, the multiplicative modifications consistently improve the performance in the PPMI-based space, although the correlation analysis shows that only some modifications do. Most of these improvements observed in the PPMI-based space are not obtained in the LSA-based space. Note that, although not shown in this paper due to lack of space, the results of the smoothed PPMI space do not significantly differ from the results of the LSA-based space.

Discussion

The obtained finding confirms our working hypothesis that the component-wise multiplication works effectively in the semantic spaces whose dimensions represent distinctive features; when reduced by SVD, the dimension of semantic spaces itself loses a semantics, and thus, the component-wise multiplication does not work in those spaces. It follows from this hypothesis that the modification of the additive models by replacing the vector addition with the component-wise multiplication also does not work in those spaces. The experiment demonstrates that in most cases it is true. Although in some cases the multiplicative modification appears effective in the smoothed space, we have no idea whether it is a robust finding or an artifact of a particular experimental setting. Furthermore, the multiplicative modification improves the additive models in the space with dimension semantics, but its degree is not so high. It is interesting for further research to explore in detail the possibility of improving the additive models using the multiplicative model.

A more interesting finding is that the multiplicative model itself can greatly improve by the simple weighting scheme, although the improved performance does not always exceed the performance of the additive models. We are developing a more sophisticated method for extending the component-wise multiplication so that the multiplicative model works in any semantic space.

Acknowledgment

This study was supported by the Grant-in-Aid for Scientific Research B (No.23300098) from the Japan Society for the Promotion of Science.

References

Baroni, M., & Zamparelli, R. (2010). Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 conference on empirical methods in natural language processing (emnlp2010)* (pp. 1183–1193).

Bullinaria, J., & Levy, J. (2007). Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, *39*(3), 510–526.

Jones, M. N., Kintsch, W., & Mewhort, D. J. (2006). High-dimensional semantic space accounts of priming. *Journal of Memory and Language*, *55*, 534–552.

Kintsch, W. (2001). Predication. *Cognitive Science*, *25*(2), 173–202.

Landauer, T., McNamara, D., Dennis, S., & Kintsch, W. (2007). *Handbook of latent semantic analysis*. Mahwah, NJ: Lawrence Erlbaum Associates.

Landauer, T. K., & Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, *104*, 211–240.

McNamara, D. S. (2011). Computational methods to extract meaning from text and advance theories of human cognition. *Topics in Cognitive Science*, *3*, 3–17.

Mitchell, J., & Lapata, M. (2008). Vector-based models of semantic composition. In *Proceedings of acl-08: Hlt* (pp. 236–244).

Mitchell, J., & Lapata, M. (2010). Composition in distributional models of semantics. *Cognitive Science*, *34*, 1388–1429.

Padó, S., & Lapata, M. (2007). Dependency-based construction of semantic space models. *Computational Linguistics*, *33*(2), 161–199.

Quesada, J. (2007). Creating your own LSA spaces. In T. Landauer, D. McNamara, S. Dennis, & W. Kintsch (Eds.), *Handbook of latent semantic analysis* (pp. 71–85). Mahwah, NJ: Lawrence Erlbaum Associates.

Recchia, G., & Jones, M. N. (2009). More data trumps smarter algorithms: Comparing pointwise mutual information with latent semantic analysis. *Behavior Research Methods*, *41*, 647–656.

Turney, P. D., & Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, *37*, 141–188.

Utsumi, A. (2011). Computational exploration of metaphor comprehension processes using a semantic space model. *Cognitive Science*, *35*(2), 251–296.

Zanzotto, F. M., Korkontzelos, I., Fallucchi, F., & Manandhar, S. (2010). Estimating linear models for compositional distributional semantics. In *Proceedings of the 23rd international conference on computational linguistics (coling-2010)* (pp. 1263–1271).