Memory-efficient Word Embedding Vectors

Jun Suzuki and Masaaki Nagata

Abstract

Word embedding is a technique for identifying the semantic relationships between words by computer. Word embedding vectors enable computers to provide a *guess* similar to the intuition or common sense of human beings. This article introduces a method for reducing the required memory consumption of this important fundamental operation of word embedding vectors while maintaining the ability to calculate semantic relationships, which is an important property when this technique is applied to real world systems.

Keywords: natural language processing, deep learning, word embeddings

1. Calculating the meaning of words

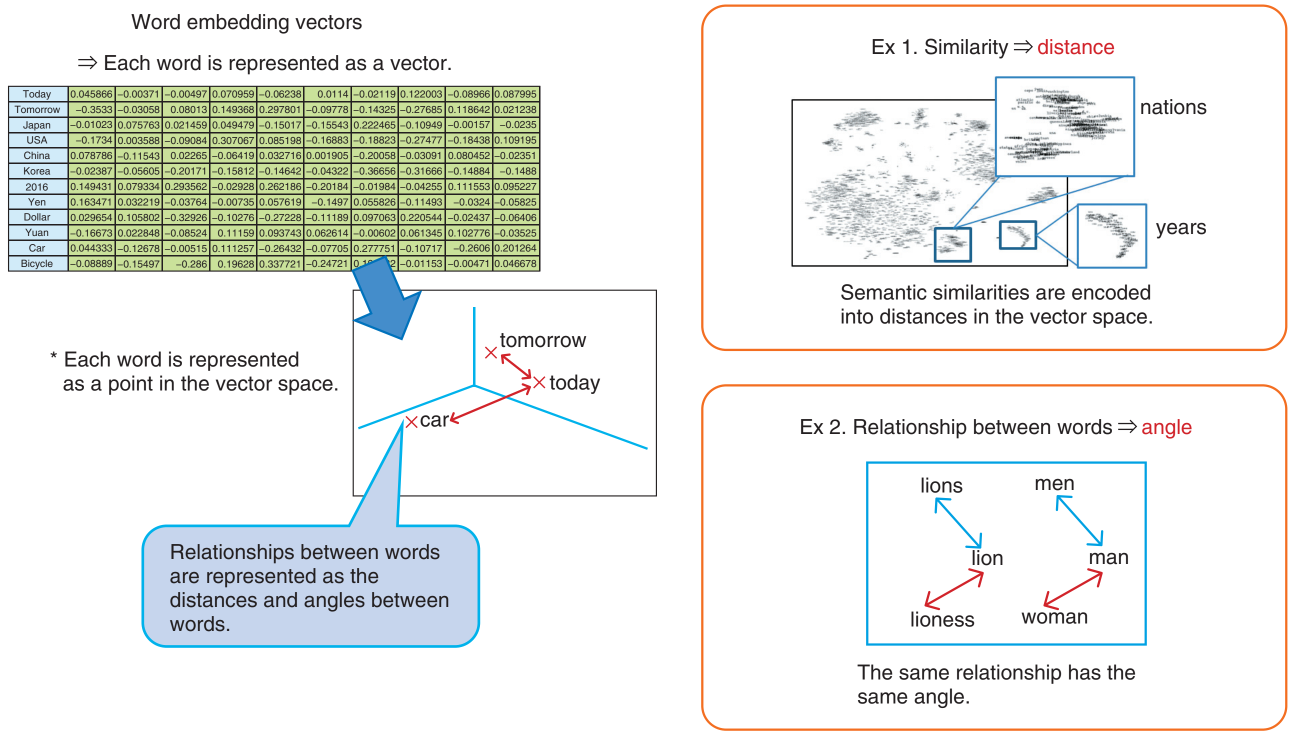
Many challenges still need to be overcome to reach the stage where computers can accurately understand and manipulate natural language at the same level as human beings. In particular, it is a difficult problem for computers to correctly understand semantic relationships between words. However, a method called *word embedding*—a technique enabling computers to understand the semantic relationships between words—is attracting a lot of attention from researchers and engineers in the natural language processing field (**Fig. 1**).

This technique was originally developed in the 1980s [1], but there has been a revival with the recent development of deep learning methods. More precisely, the method proposed by Mikolov et al. [2] has empirically proven that word embedding vectors can be trained within a feasible run-time even if the size of training data is very large, for example, web-scale data. Thus, the method can capture very accurate semantic relationships between words with the help of large-scale text data since such large-scale data should implicitly contain information equivalent to the *common sense* of human beings.

As a simple example for explaining the usefulness of word embedding vectors, computers can estimate the meanings of words by the vector calculations among the word embedding vectors. Suppose we ask someone "Which word has the most appropriate relation to the word 'Germany,' if the word is based on the same relation existing between 'France' and 'wine'?" Many people would answer "beer," for instance. Of course there is no unique correct answer for this question, and some people might say that "beer" is not a correct answer in his/her view. However, many people feel that "beer" is an acceptable answer.

These days, computers are becoming capable of developing such common sense or knowledge of human beings with the help of word embedding vectors. The most important point here is that this type of intuitive guess—similar to that done by human beings—is now manageable for computers.

Traditionally, the approach used by computers to identify the semantics of words involved the use of hand-made semantic dictionaries. The essential difference from such traditional dictionary-based methods with the word embedding approach is coverage and whether the method involves an automatic or hand-made construction. It is easily imaginable that the traditional dictionary-based methods can solve semantic problems with high accuracy if the dictionary has information on the given problems; if not, dictionary-based methods are not effective for solving such problems. Moreover, a large cost may be required to keep updating the dictionary to improve


Word embedding vectors
⇒ Each word is represented as a vector.
* Each word is represented as a point in the vector space.
tomorrow
today
car
Relationships between words are represented as the distances and angles between words.
Ex 1. Similarity ⇒ distance
nations
years
Semantic similarities are encoded into distances in the vector space.
Ex 2. Relationship between words ⇒ angle
lions
men
lion
man
lioness
woman
The same relationship has the same angle.

Today	0.045866	–0.00371	–0.00497	0.070959	–0.06238	0.0114	–0.02119	0.122003	–0.08966	0.087995
Tomorrow	–0.3533	–0.03058	0.08013	0.149368	0.297801	–0.09778	–0.14325	–0.27685	0.118642	0.021238
Japan	–0.01023	0.075763	0.021459	0.049479	–0.15017	–0.15543	0.222465	–0.10949	–0.00157	–0.0235
USA	–0.1734	0.003588	–0.09084	0.307067	0.085198	–0.16883	–0.18823	–0.27477	–0.18438	0.109195
China	0.078786	–0.11543	0.02265	–0.06419	0.032716	0.001905	–0.20058	–0.03091	0.080452	–0.02351
Korea	–0.02387	–0.05605	–0.20171	–0.15812	–0.14642	–0.04322	–0.36656	–0.31666	–0.14884	–0.1488
2016	0.149431	0.079334	0.293562	–0.02928	0.262186	–0.20184	–0.01984	–0.04255	0.111553	0.095227
Yen	0.163471	0.032219	–0.03764	–0.00735	0.057619	–0.1497	0.055826	–0.11493	–0.0324	–0.05825
Dollar	0.029654	0.105802	–0.32926	–0.10276	–0.27228	–0.11189	0.097063	0.220544	–0.02437	–0.06406
Yuan	–0.16673	0.022848	–0.08524	0.11159	0.093743	0.062614	–0.00602	0.061345	0.102776	–0.03525
Car	0.044333	–0.12678	–0.00515	0.111257	–0.26432	–0.07705	0.277751	–0.10717	–0.2606	0.201264
Bicycle	–0.08889	–0.15497	–0.286	0.19628	0.337721	–0.24721	[illegible]	–0.01153	–0.00471	0.046678

Fig. 1. Word embedding vectors.

the coverage.

For example, a computer might not be able to solve the above example of *beer* using a dictionary-based method since the importance of such common sense is relatively low, and thus, it might not be included in the dictionary. In contrast, word embedding vectors can be automatically generated from a large amount of text data, and no human cost is required for maintenance. Moreover, conceptually, information on all the words appearing in the training data can be stored in the embedding vectors. In fact, word embedding can easily handle millions of words. This fact implies that the word embedding method can handle a much larger number of words than dictionary-based methods (**Fig. 2**).

2. Usefulness of word embedding vectors for computers

Word embedding vectors can be utilized in many natural language processing applications such as machine translation, question answering, information retrieval, and document summarization. However, we sometimes encounter several inconvenient points when trying to apply word embedding vectors to real systems. For example, there are a lot of random factors when word embedding vectors are constructed using conventional methods. Therefore, the resultant word embedding vectors lack reproducibility, meaning they always differ from each other when many trials are conducted.

In another example, we have to completely retrain embedding vectors in situations where we need embedding vectors with distinct numbers of dimensions, since the dimensions of word embedding vectors can be pre-defined before starting the training, and different applications often prefer their own numbers of dimensions. This is an example indicating that an advanced technology cannot always be easily applied to real world systems. To overcome this inconvenience of low usability, we have developed several methods that have high usability [3–5]. In the remainder of this article, we explain one of our methods for significantly reducing the memory requirements of word embedding vectors [5].

3. Method for reducing memory requirements

We first explain the usefulness of reducing memory requirements. Suppose we are building a dialogue

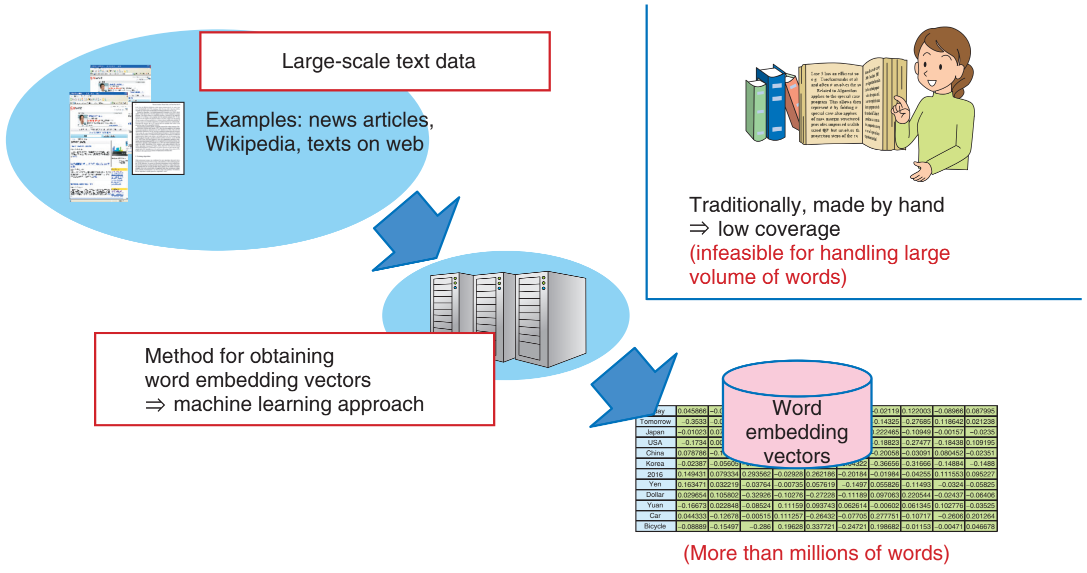

Large-scale text data
Examples: news articles, Wikipedia, texts on web
Traditionally, made by hand
⇒ low coverage
(infeasible for handling large volume of words)
Method for obtaining word embedding vectors
⇒ machine learning approach
Word embedding vectors
(More than millions of words)

Fig. 2. Method for obtaining word embedding vectors.

system in robots used mainly for communicating with users. In this case, we aim to put as many words into the system as possible since it is nearly impossible for the system to appropriately process unknown words. However, there is a trade-off between performance and memory requirements in general. Namely, the amount of required memory storage to store all of the word embedding vectors becomes a large problem when we add a large number of words into the systems.

For example, let us consider the case when utilizing three million words and a 300-dimensional vector is assigned to each word. Here, we assume that we need a 4-byte memory to represent a single precision floating point number. Then the memory requirement for representing overall word embedding vectors becomes 3,000,000 (words) x 300 (dimensions) x 4 (bytes) = 3,600,000,000 (bytes). This means that it requires 3.4 GB of memory. We emphasize that we need 3.4 GB of memory only for a *single* module, not for an entire system. This is unacceptably large in general.

Here, we assume that memory requirements are one-hundredth, that is, 34 MB, of the above amount. Then the total cost of memory storage integrated into robots can be significantly reduced. This actual cost reduction is essentially the most important factor in a real world product. In addition, we can easily integrate word embedding vectors into applications on mobile devices. Less memory usage also leads to lower power consumption even though the memory storage in mobile devices may increase rapidly in the near future. Consequently, we can expect various positive effects for real world applications by merely developing a means of reducing the memory requirements.

4. Method

Our method for reducing the memory requirements consists of a combination of several machine learning techniques such as group regularization, dual decomposition, augmented Lagrangian methods, and clustering. We do not describe these techniques in detail here but rather briefly explain the essence of our method. As described previously, word embedding vectors are generated from large training data. More precisely, what the method is actually trying to do during the learning process is to find appropriate values in the embedding vectors assigned to each word by minimizing the given objective function. The basic idea of our method is as follows: Suppose we observe that a certain value sequence pattern, for example, (0.3, –0.2, 0.1, 0.5), appears many times in the obtained embedding vectors. In this situation, we can discard these patterns with no information loss by preserving a single value sequence pattern among them and adding information consisting of *appearance of the same value sequence pattern* to the locations where all the same value sequence patterns

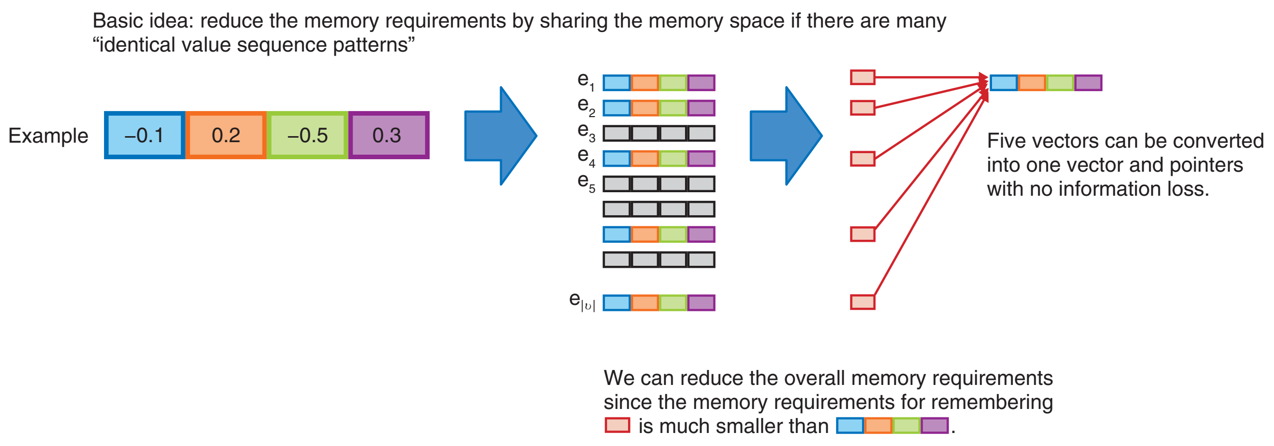

Basic idea: reduce the memory requirements by sharing the memory space if there are many "identical value sequence patterns"
Example
−0.1
0.2
−0.5
0.3
e1
e2
e3
e4
e5
e|υ|
Five vectors can be converted into one vector and pointers with no information loss.
We can reduce the overall memory requirements since the memory requirements for remembering ▭ is much smaller than ▭.

Fig. 3. Idea for reducing the memory requirements.

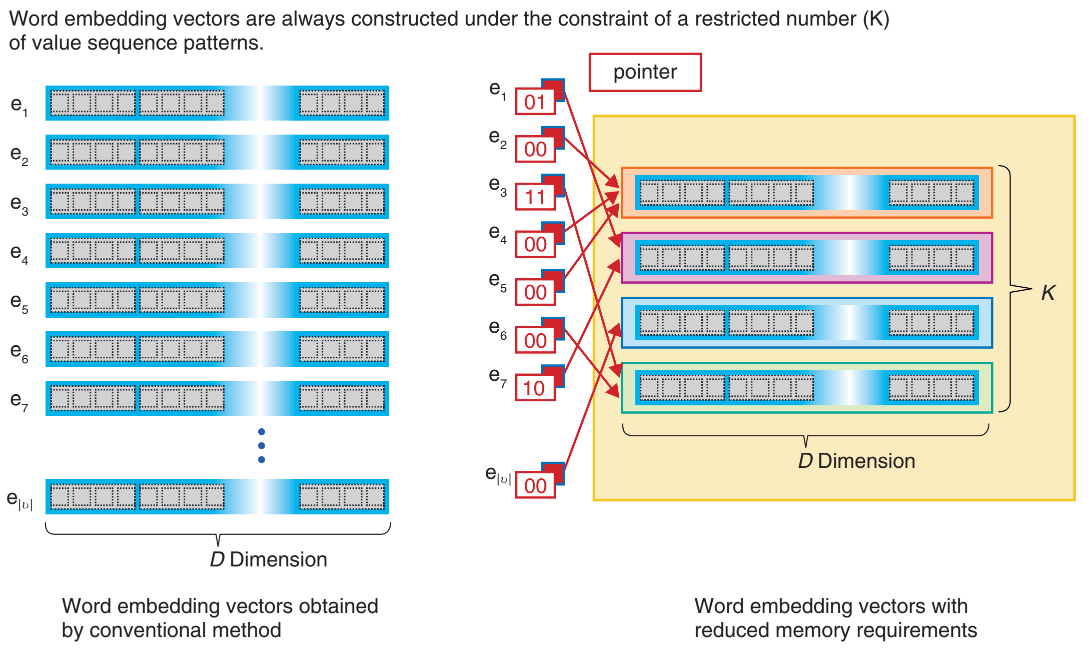

Word embedding vectors are always constructed under the constraint of a restricted number (K) of value sequence patterns.
e1
e2
e3
e4
e5
e6
e7
e|υ|
D Dimension
Word embedding vectors obtained by conventional method
pointer
01
00
11
00
00
00
10
00
K
D Dimension
Word embedding vectors with reduced memory requirements

Fig. 4. Proposed method.

appear. Then, we can reduce the overall memory requirements if the memory requirement of remembering where the value sequence patterns appear is smaller than remembering the original value sequence patterns (**Fig. 3**).

By implementing this idea, we can continue to reduce the overall memory requirements if the number of distinct value sequence patterns in the word embedding vectors gets smaller and smaller. Unfortunately, however, none of the conventional methods automatically generate such convenient value sequence patterns. Therefore, we have built a method that can force the system to produce the word embedding vectors under the condition of *constructing word embedding vectors with pre-defined K distinct value sequence patterns* while maintaining the performance. Using this method, we can produce word embedding vectors within the memory desired by the users (**Fig. 4**).

5. Future direction

We are conducting research with the objective of having the most advanced research results become basic technologies that are used in real world systems including artificial intelligence related systems. It is possible to directly and indirectly support improvements of actual systems being used by further developing the basic technologies used in the systems. Thus, our final goal is to develop many basic technologies that offer high usability for computers and system developers, which we believe to be one of the most important characteristics of basic technologies.

References

[1] G. E. Hinton, "Learning Distributed Representations of Concepts," Proc. of the Eighth Annual Conference of the Cognitive Science Society, pp. 1–12, Amherst, MA, USA, Aug. 1986.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," Proc. of the First International Conference on Learning Representations (ICLR 2013), Scottsdale, AZ, USA, May 2013.

[3] J. Suzuki and M. Nagata, "A Unified Learning Framework of Skip-grams and Global Vectors," Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference of the Asian Federation of Natural Language Processing (ACL-ICNLP2015), Beijing, China, July 2015.

[4] J. Suzuki and M. Nagata, "Right-truncatable Neural Word Embeddings," Proc. of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT2016), San Diego, CA, USA, June 2016.

[5] J. Suzuki and M. Nagata, "Learning Compact Neural Word Embeddings by Parameter Space Sharing," Proc. of the 25th International Joint Conference on Artificial Intelligence (IJCAI-16), New York, USA, July 2016.

Jun Suzuki

Senior Research Scientist, Linguistic Intelligence Research Group, Innovative Communication Laboratory, NTT Communication Science Laboratories.

He received a Ph.D. in engineering from the Graduate School of Information Science, Nara Institute of Science and Technology in 2005. He joined NTT Communication Science Laboratories in 2001, where he is researching machine learning, natural language processing, and artificial intelligence areas.

Masaaki Nagata

Senior Distinguished Researcher, Group Leader, NTT Communication Science Laboratories.

He received a B.E., M.E., and Ph.D. in information science from Kyoto University in 1985, 1987, and 1999. He joined NTT in 1987. His research interests include morphological analysis, named entity recognition, parsing, and machine translation. He is a member of the Institute of Electronics, Information and Communication Engineers, the Information Processing Society of Japan, the Japanese Society for Artificial Intelligence, the Association for Natural Language Processing, and the Association for Computational Linguistics.