Right-truncatable Neural Word Embeddings

Jun Suzuki and **Masaaki Nagata**
NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237 Japan
{suzuki.jun, nagata.masaaki}@lab.ntt.co.jp

Abstract

This paper proposes an incremental learning strategy for neural word embedding methods, such as SkipGrams and Global Vectors. Since our method iteratively generates embedding vectors one dimension at a time, obtained vectors equip a unique property. Namely, any right-truncated vector matches the solution of the corresponding lower-dimensional embedding. Therefore, a single embedding vector can manage a wide range of dimensional requirements imposed by many different uses and applications.

1 Introduction

Word embedding vectors obtained from '*neural word embedding methods*', such as SkipGram, continuous bag-of-words (CBoW) and the family of vector log-bilinear (vLBL) models (Mnih and Kavukcuoglu, 2013; Mikolov et al., 2013a; Mikolov et al., 2013c; Mikolov et al., 2013b) have now become an important fundamental resource for tackling many natural language processing (NLP) tasks. These NLP tasks include part-of-speech tagging (Tsuboi, 2014; Ling et al., 2015), dependency parsing (Chen and Manning, 2014; Dyer et al., 2015; Alberti et al., 2015), semantic role labeling (Zhou and Xu, 2015; Woodsend and Lapata, 2015), machine translation (Sutskever et al., 2014), sentiment analysis (Kim et al., 2015), and question answering (Wang and Nyberg, 2015).

The main purpose of this paper is to further enhance the 'usability' of obtained embedding vectors in actual use. To briefly explain our motivation, we first introduce the following concept:

Definition 1 (D'-right-truncated vector[1])**.** *Let* $\mathbf{w}'$ *and* $\mathbf{w}''$ *be vectors, whose dimensions are* D' *and* D''*, respectively. Namely,* $\mathbf{w}' = (w'_1, \ldots, w'_{D'})$ *and* $\mathbf{w}'' = (w''_1, \ldots, w''_{D''})$*. Suppose* $\mathbf{w}$ *matches the concatenation of* $\mathbf{w}'$ *and* $\mathbf{w}''$*, that is,* $\mathbf{w} = (w'_1, \ldots, w'_{D'}, w''_1, \ldots, w''_{D''})$*. Then, we define* $\mathbf{w}'$ *as a* D'*-right-truncated vector of* $\mathbf{w}$*.*

This paper focuses on the fact that the appropriate dimension of embedding vectors strongly depends on applications and uses, and is basically determined based on the performance and memory space (or calculation speed) trade-off. Indeed, the actual dimensions of the previous studies listed above are diverse; often around 50, and at most 1000. It is worth noting here that each dimension of embedding vectors obtained by conventional methods has no interpretable meaning. Thus, we basically need to retrain D'-dimensional embedding vectors even if we already have a well-trained D-dimensional vector. In addition, we cannot take full advantage of freely available high-quality pre-trained embedding vectors[2] since their dimensions are already given and fixed, *i.e.*, $D = 300$.

To reduce the additional computational cost of the retraining, and to improve the 'usability' of embedding vectors, we propose a framework for incrementally determining embeddings one dimension at a time from 1 to D. As a result, our method always offers the relation that '*any* D'*-right-truncated em-*

[1] The term 'right-truncated' is originally taken from 'right-truncatable prime'

[2] *i.e.*, 'GoogleNews-vectors-negative300' obtained from https://code.google.com/archive/p/word2vec/, and 'glove.840B.300d' obtained from http://nlp.stanford.edu/projects/glove/

Proceedings of NAACL-HLT 2016, pages 1145–1151,
San Diego, California, June 12-17, 2016.

bedding vector is the solution for D'-dimensional embeddings of our method'. Therefore, in actual use, we only need to construct a relatively higher-dimensional embedding vector 'just once', *i.e.*, $D = 1000$, and then truncate it to an appropriate dimension for the application.

2 Neural Word Embedding Methods

Let $\mathcal{U}$ and $\mathcal{V}$ be two sets of predefined vocabularies of possible inputs and outputs. Let $|\mathcal{U}|$ and $|\mathcal{V}|$ be the number of words in $\mathcal{U}$ and $\mathcal{V}$, respectively. Then, neural word embedding methods generally assign a D-dimensional vector to each word in $\mathcal{U}$ and $\mathcal{V}$. We denote $\mathbf{e}_i$ as representing the i-th input vector, and $\mathbf{o}_j$ for the j-th output vector. In the rest of this paper, for convenience the notation 'i' is always used as the index of input vectors, and 'j' as the index of output vectors, where $1 \leq i \leq |\mathcal{U}|$ and $1 \leq j \leq |\mathcal{V}|$.

We introduce $\mathbf{E}$ and $\mathbf{O}$ that represent lists of all input and output vectors, respectively. Namely, $\mathbf{E} = (\mathbf{e}_1, \cdots, \mathbf{e}_{|\mathcal{U}|})$ and $\mathbf{O} = (\mathbf{o}_1, \cdots, \mathbf{o}_{|\mathcal{V}|})$. $\mathcal{X}$ represents training data. Then, embedding vectors are obtained by solving the following form of a minimization problem defined in each neural word embedding method:

$$(\hat{\mathbf{E}}, \hat{\mathbf{O}}) = \arg\min_{\mathbf{E},\mathbf{O}} \big\{\Psi(\mathbf{E}, \mathbf{O} \mid \mathcal{X})\big\}, \qquad (1)$$

where Ψ represents the objective function, and $\hat{\mathbf{E}}$ and $\hat{\mathbf{O}}$ are lists of solution embedding vectors.

Hereafter, we use Ψ as an abbreviation of $\Psi(\mathbf{E}, \mathbf{O} \mid \mathcal{X})$. For example, the objective function Ψ of 'SkipGram with negative sampling (SGNS)' can be written in the following form[3] :

$$\Psi = \sum_{(i,j)} \Big(c_{i,j} L(x_{i,j}) + c'_{i,j} L(-x_{i,j})\Big), \qquad (2)$$

where $x_{i,j} = \mathbf{e}_i \cdot \mathbf{o}_j$, and $L(x)$ represents a logistic loss function, namely, $L(x) = \log(1 + \exp(-x))$. Moreover, $c_{i,j}$ and $c'_{i,j}$ represent co-occurrences of the i-th input and j-th output words in training data and negative sampling data, respectively.

Another example, the objective function Ψ of the 'Global Vector (GloVe)' can be written in the following form (Pennington et al., 2014):

$$\Psi = \frac{1}{2} \sum_{(i,j)} \beta_{i,j} (x_{i,j} - m_{i,j})^2, \qquad (3)$$

where $m_{i,j}$ and $\beta_{i,j}$ represent certain co-occurrence and weighting factors of the i-th input and the j-th output words, respectively. For example, $\beta_{i,j} = \min(1, (c_{i,j}/x_{\max})^\gamma)$, and $m_{i,j} = \log(c_{i,j})$ are used in (Pennington et al., 2014), where $x_{\max}$ and γ are tunable hyper-parameters.

[3] We can obtain this form by a simple reformulation from the original objective of SGNS (Mikolov et al., 2013b).

Input: $\mathcal{X}$: training data, D: maximum number of dimensions (iterations)
1: $\mathbf{E}^{(0)} \leftarrow \emptyset$, $\mathbf{O}^{(0)} \leftarrow \emptyset$, and $\mathbf{B}^{(0)} \leftarrow \mathbf{0}$, $d \leftarrow 0$
2: **repeat**
3: $d \leftarrow d + 1$
4: $(\bar{\mathbf{q}}_d, \bar{\mathbf{r}}_d) \leftarrow \texttt{updateParams1D}(\mathcal{X}, \mathbf{B}^{(d-1)})$ // Eq. 5
5: $\mathbf{E}^{(d)} \leftarrow \texttt{appendVec}(\mathbf{E}^{(d-1)}, \bar{\mathbf{q}}_d)$
6: $\mathbf{O}^{(d)} \leftarrow \texttt{appendVec}(\mathbf{O}^{(d-1)}, \bar{\mathbf{r}}_d)$
7: $\mathbf{B}^{(d)} \leftarrow \texttt{updateBias}(\mathbf{B}^{(d-1)}, \bar{\mathbf{q}}_d, \bar{\mathbf{r}}_d)$ // Eq. 4
8: **until** $d = D$
Output: $(\mathbf{E}^{(D)}, \mathbf{O}^{(D)})$

Figure 1: An algorithm for solving an iterative additional coordinate optimization formulation for obtaining embedding vectors.

3 Incremental Construction of Embedding

This section explains our proposed method. The basic idea is very simple and clear: we convert the minimization problem shown in Eq. 1 to a series of minimization problems, each of whose individual problem determines one additional dimension of each embedding vector. We refer to this formulation of embedding problems as '*ITerative Additional Coordinate Optimization (ITACO)*' formulation. Fig. 1 shows our entire optimization algorithm for this formulation.

3.1 Bias terms and optimization variables

Suppose d represents a discrete time step, where $d \in \{1, \ldots, D\}$. Let $\mathbf{B}^{(d)}$ be a matrix representation of bias terms at the d-th time step, and $b_{i,j}^{(d)}$ denote the (i,j)-factor of $\mathbf{B}^{(d)}$. Then, we define that $b_{i,j}^{(d)}$ for all (i,j) and d have the following recursive relation:

$$b_{i,j}^{(d)} = \sum_{k=1}^{d} e_{i,k} o_{j,k} = b_{i,j}^{(d-1)} + e_{i,d} o_{j,d}, \qquad (4)$$

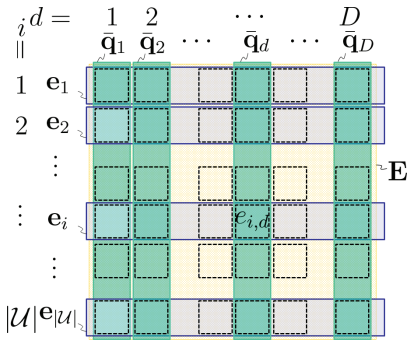

Figure 2: Relation of $\mathbf{e}_i$ and $\bar{\mathbf{q}}_d$ used to represent input vectors in this paper.

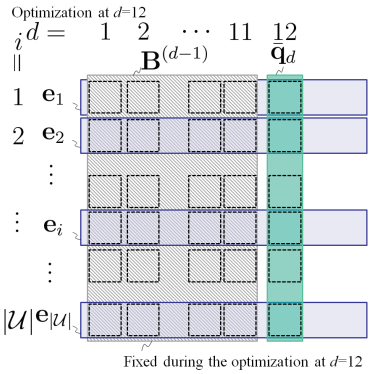

Figure 3: Relation of $\mathbf{B}^{(d-1)}$ and $\bar{\mathbf{q}}_d$, which are the constant and optimization parameters in the d-th optimization problems, respectively.

where we define $b_{i,j}^{(0)} = 0$ for all (i, j). This relation implies that the solutions of former optimizations are used as bias terms in latter optimizations.

Next, we define $\bar{\mathbf{q}}_d$ and $\bar{\mathbf{r}}_d$ as the vector representations of the concatenation of all the input and output parameters at the d-th step, respectively, that is, $\bar{\mathbf{q}}_d = (e_{1,d}, \ldots, e_{|\mathcal{U}|,d})$ and $\bar{\mathbf{r}}_d = (o_{1,d}, \ldots, o_{|\mathcal{V}|,d})$. Note that $\mathbf{e}_i$ used in the former part of this paper is a D-dimensional vector while $\bar{\mathbf{q}}_d$ and $\bar{\mathbf{r}}_d$ defined here are $|\mathcal{U}|$-dimensional and $|\mathcal{V}|$-dimensional vectors, respectively. Moreover, there are relations that $e_{i,d}$ is the d-th factor of $\mathbf{e}_i$, and, at the same time, the i-th factor of $\bar{\mathbf{q}}_d$.

Fig. 2 illustrates the relation of $\mathbf{e}_i$ and $\bar{\mathbf{q}}_d$ in this paper. We omit to explicitly show the relation of $\mathbf{o}_j$ and $\bar{\mathbf{r}}_d$, which are used to represent output vectors because of the space reason. However obviously, they also have the same relation as $\mathbf{e}_i$ and $\bar{\mathbf{q}}_d$.

3.2 Individual optimization problem

Then, we define the d-th optimization problem in our ITACO formulation as follows:

$$\begin{aligned} (\bar{\mathbf{q}}_d, \bar{\mathbf{r}}_d) &= \arg\min_{\bar{\mathbf{q}}, \bar{\mathbf{r}}} \left\{ \bar{\Psi}\left(\bar{\mathbf{q}}, \bar{\mathbf{r}} | \mathcal{X}, \mathbf{B}^{(d-1)}\right) \right\} \\ \text{subject to:} \quad & \frac{|\mathcal{V}|}{|\mathcal{U}|} ||\bar{\mathbf{q}}||_p = ||\bar{\mathbf{r}}||_p, \end{aligned} \tag{5}$$

where $||\cdot||_p$ represents the L_p-norm. We generally assume that $p = \{1, 2, \infty\}$, and often select $p = 2$. Note that $\bar{\mathbf{q}}_d$ is optimization parameters in the d-th optimization problem while $\mathbf{B}^{(d-1)}$ is the constant. Fig. 3 illustrates the relation of $\mathbf{B}^{(d-1)}$ and $\bar{\mathbf{q}}_d$.

We assume that the objective function $\bar{\Psi}$ takes an identical form as used in one of the conventional methods such as SGNS and GloVe as shown by Eqs. 2 and 3. The difference appears in the variables; our ITACO formulation uses $x_{i,j} = e_i o_j + b_{i,j}$ rather than $x_{i,j} = \mathbf{e}_i \cdot \mathbf{o}_j$ as described in Sec. 2.

3.3 Improving stability of embeddings

The additional norm constraint in Eq. 5 is introduced to improve stability. The optimization problems of neural word embedding methods including SGNS and GloVe can be categorized as a *bi-convex optimization problem* (Gorski et al., 2007); they are convex with respect to the parameters $\mathbf{E}$ if the parameters $\mathbf{O}$ are assumed to be constants, and vice versa. One well-known drawback of unconstrained bi-convex optimization is that the optimization parameters can possibly diverge to $\pm\infty$ (See Example 4.3 in (Gorski et al., 2007)). This is because the objective function only cares about the inner product value of two vectors. Therefore, each parameter can easily have a much larger value, *i.e.*, $o_1 = 10^9$, if e_1 is smaller and approaches a zero value *i.e.*, $e_1 = 10^{-10}$. This is mainly caused by inconsistent scale problem. Thus, our norm constraint in Eq. 5 can eliminate this problem by maintaining the scale of $\bar{\mathbf{q}}$ and $\bar{\mathbf{r}}$ at the same level.

3.4 Optimization algorithm

To solve Eq. 5, we employ the idea of the '*Alternating Convex Optimization (ACO)*' algorithm (Gorski et al., 2007). ACO and its variants have been widely developed in the context of (non-negative) matrix factorization, *i.e.*, (Kim et al., 2014), and are empirically known to be an efficient method in practice. The main idea of ACO is that it iteratively and al-

ternatively updates one parameter set, *i.e.*, $\bar{\mathbf{q}}$, while the other distinct parameter set is fixed, *i.e.*, $\bar{\mathbf{r}}$. In our case, ACO solves the following two optimization problems iteratively and alternately:

$$\bar{\mathbf{q}}_d = \arg\min_{\bar{\mathbf{q}}} \left\{ \bar{\Psi}(\bar{\mathbf{q}}, \bar{\mathbf{r}} \mid \mathcal{X}, \mathbf{B}^{(d-1)}) \right\} \quad (6)$$

$$\bar{\mathbf{r}}_d = \arg\min_{\bar{\mathbf{r}}} \left\{ \bar{\Psi}(\bar{\mathbf{q}}, \bar{\mathbf{r}} \mid \mathcal{X}, \mathbf{B}^{(d-1)}) \right\}. \quad (7)$$

There are at least two advantages of using ACO; (1) Eqs. 6 and 7 both become convex optimization problems. Therefore, the global optimum solution can be obtained when $\partial_{e_i} \bar{\Psi} = 0$ for all i and $\partial_{o_j} \bar{\Psi} = 0$ for all j, respectively. (2) ACO guarantees to converge to a stationary point (one of the local minima)[4].

For example, by a simple reformulation of $\partial_{e_i} \bar{\Psi} = 0$, we obtain the closed form solution of Eq. 6 with the GloVe objective, that is,

$$e_i = \frac{\sum_j \beta_{i,j}(m_{i,j} - b_{i,j}) o_j}{\sum_j \beta_{i,j} (o_j)^2} \quad \forall i. \quad (8)$$

Similarly, the closed form solution of Eq. 7 is:

$$o_j = \frac{\sum_i \beta_{i,j}(m_{i,j} - b_{i,j}) e_i}{\sum_i \beta_{i,j} (e_i)^2} \quad \forall j. \quad (9)$$

Thus, we can solve Eqs. 6 and 7 without performing iterative estimation. Next, we obtain the following equation by a simple reformulation of $\partial_{e_i} \bar{\Psi} = 0$ for the SGNS objective:

$$\sum_j c_{i,j} o_j = \sum_j (c_{i,j} + c'_{i,j}) \sigma(e_i o_j + b_{i,j}) o_j, \quad (10)$$

where $\sigma(x)$ represents a sigmoid function, that is, $\sigma(x) = \frac{1}{1+\exp(-x)}$. Similarly, we also obtain the following form of the equation for Eq. 7:

$$\sum_i c_{i,j} e_i = \sum_i (c_{i,j} + c'_{i,j}) \sigma(e_i o_j + b_{i,j}) e_i. \quad (11)$$

These equations are efficiently solvable by a simple binary search procedure since each equation only has a single parameter, that is, e_i or o_j,

During the optimization, there is no guarantee that the constraint $\frac{|\mathcal{V}|}{|\mathcal{U}|} ||\bar{\mathbf{q}}||_p = ||\bar{\mathbf{r}}||_p$ always holds. Fortunately, the following transformations always satisfy this norm constraint:

$$\tilde{e}_i = \frac{|\mathcal{U}|}{|\mathcal{V}|} \frac{e_i}{||\bar{\mathbf{q}}||_p} \left(\frac{|\mathcal{V}|}{|\mathcal{U}|} ||\bar{\mathbf{q}}||_p ||\bar{\mathbf{r}}||_p \right)^{\frac{1}{2}} \quad \forall i$$
$$\tilde{o}_j = \frac{o_j}{||\bar{\mathbf{r}}||_p} \left(\frac{|\mathcal{V}|}{|\mathcal{U}|} ||\bar{\mathbf{q}}||_p ||\bar{\mathbf{r}}||_p \right)^{\frac{1}{2}} \quad \forall j, \quad (12)$$

which also maintain $\tilde{e}_i \tilde{o}_j = e_i o_j$, and the objective value. Thus, we can safely apply them at any time during the optimization.

Finally, Fig. 4 shows the optimization procedure when using the ACO framework.

Input: $\mathcal{X}$: training data, $\mathbf{B}$: matrix form of bias terms, ϵ: constant for convergence check
$\bar{\mathbf{q}} \leftarrow \mathbf{1}$, and $\bar{\mathbf{r}} \leftarrow \mathbf{0}$
repeat
  $\bar{\mathbf{r}} \leftarrow$ `updateIVec1D`$(\bar{\mathbf{r}} \mid \bar{\mathbf{q}}, \mathbf{B})$ // Eq. 11 or 9
  $(\bar{\mathbf{q}}, \bar{\mathbf{r}}) \leftarrow$ `scaleVec`$(\bar{\mathbf{q}}, \bar{\mathbf{r}})$ // Eq. 12
  $\bar{\mathbf{q}} \leftarrow$ `updateOVec1D`$(\bar{\mathbf{q}} \mid \bar{\mathbf{r}}, \mathbf{B})$ // Eq. 10 or 8
  $(\bar{\mathbf{q}}, \bar{\mathbf{r}}) \leftarrow$ `scaleVec`$(\bar{\mathbf{q}}, \bar{\mathbf{r}})$ // Eq. 12
until `ConvergenceCheck`(ϵ)
Output: $(\bar{\mathbf{q}}, \bar{\mathbf{r}})$

Figure 4: Procedure of `updateParams1D` in Fig. 1 using the ACO-based algorithm.

4 Experiments

As in previously reported neural word embedding papers, our training data was taken from a Wikipedia dump (Aug. 2014). We used `hyperwords` tool[5] for our data preparation (Levy et al., 2015).

We compared our method, **ITACO**, with the widely used conventional methods, **SGNS** and **GloVe**. We used the `word2vec` implementation[6] to obtain word embeddings of SGNS, and `glove` implementation[7] for GloVe. Many tunable hyper-parameters were selected based on the recommended default values of each implementation, or suggestion explained in (Levy et al., 2015). For ITACO, we selected the Glove objective to solve Eqs. 6 and 7 since it requires a lower calculation cost than the SGNS objective.

We prepared three types of linguistic benchmark tasks, namely word similarity estimation (**Similarity**), word analogy estimation (**Analogy**), and sentence completion (**SentComp**) tasks. We gathered

[4]We somehow prevent the divergence of optimization parameters (Gorski et al., 2007)

[5]https://bitbucket.org/omerlevy/hyperwords

[6]https://code.google.com/p/word2vec/ (We made a modification to save the context vector as well as the word vector.)

[7]http://nlp.stanford.edu/projects/glove/

Methods	Analogy D=10	50	100	300	500	1000
ITACO (trunc)	*__2.6__	__38.0__	__51.5__	__63.5__	__65.4__	*__65.6__
SGNS (trunc)	0.0	11.8	34.6	57.4	61.9	63.2
GloVe (trunc)	0.1	20.1	42.3	60.7	63.2	*__65.6__
SGNS (retrain)	1.7	37.7	51.8	62.3	64.0	–
GloVe (retrain)	*2.6	*42.6	*57.0	*65.6	*66.4	–
Methods	Similarity D=10	50	100	300	500	1000
ITACO (trunc)	__41.6__	__55.2__	__58.5__	__61.4__	__62.2__	62.9
SGNS (trunc)	29.0	46.1	52.5	60.7	61.8	*__64.5__
GloVe (trunc)	29.3	46.2	50.6	56.8	58.5	59.9
SGNS (retrain)	*46.4	*58.2	*61.3	*63.9	*64.2	–
GloVe (retrain)	38.7	51.4	54.2	56.8	58.1	–
Methods	SentComp D=10	50	100	300	500	1000
ITACO (trunc)	*__29.2__	*__32.3__	__32.1__	*__34.3__	*__35.4__	*__36.6__
SGNS (trunc)	24.2	26.2	29.9	32.1	32.3	36.1
GloVe (trunc)	22.8	24.3	26.7	26.7	28.2	27.7
SGNS (retrain)	24.8	29.7	*33.0	32.7	33.9	–
GloVe (retrain)	26.3	27.3	27.1	28.1	28.1	–

Table 1: Results of right-truncated embedding vectors (trunc), and standard embedding vectors (retrain). '*' represents the best results in the corresponding column.

nine datasets for Similarity (Rubenstein and Goodenough, 1965; Miller and Charles, 1991; Agirre et al., 2009; Agirre et al., 2009; Bruni et al., 2014; Radinsky et al., 2011; Huang et al., 2012; Luong et al., 2013; Hill et al., 2014), three for Analogy (Mikolov et al., 2013a; Mikolov et al., 2013c) , and one for SentComp (Mikolov et al., 2013a).

Table 1 shows all the results of our experiments[8]. The rows labeled '(trunc)' show the performance of D-right-truncated embedding vectors, whose original vector of dimension is $D = 1000$. Thus, they were obtained from a single set of embedding vectors with $D = 1000$ for each corresponding method. Next, the rows labeled '(retrain)' show the performance provided by SGNS or GloVe that were independently constructed with using a standard setting and corresponding D. Note that the results of 'ITACO (retrain)' are identical to those of 'ITACO (trunc)'. Moreover, 'GloVe (trunc)' and 'GloVe (retrain)' in $D = 1000$ are equivalent, as are 'SGNS (trunc)' and 'SGNS (retrain)'. Thus, these results were omitted from the table.

First, comparing '(retrain)' and '(trunc)' in SGNS and GloVe, our experimental results first explicitly revealed that SGNS and GloVe with the simple truncation approach '(trunc)' cannot provide effective lower-dimensional embedding vectors. This observation strongly supports the significance of existence of our proposed method, ITACO.

Second, in most cases ITACO successfully provided almost the same performance level as the best SGNS and GloVe (retrain) results. We emphasize that ITACO constructed embedding vectors 'just once', while SGNS and GloVe required us to retrain embedding vectors in the corresponding times. In addition, single run of ITACO for $D = 1000$ took approximately 12,000 seconds in our machine environment, which was almost equivalent to run 4 iterations of SGNS and 8 iterations of GloVe. The results of SGNS and GloVe in Table 1 were obtained by 10 iterations and 20 iterations, respectively, which are one of the standard settings to run SGNS and GloVe[9]. This fact verified that ITACO can run efficiently as in the same level as SGNS and GloVe.

5 Conclusion

This paper proposed a method for generating interesting right-truncatable word embedding vectors. Our experiments revealed that the embedding vectors obtained with our method, ITACO, in any lower dimensions work as well as those obtained by SGNS and Glove. In addition, ITACO can also be a good alternative of SGNS and GloVe in terms of the execution speed of a single run. Now, we are free from retraining different dimensions of embedding vectors by using ITACO. Our method significantly reduces the total calculation cost and storage, which improves the 'usability' of embedding vectors[10].

Acknowledgment

We thank three anonymous reviewers for their helpful comments.

[8]Results for SGNS and GloVe are the average performance of ten runs as suggested in (Suzuki and Nagata, 2015)

[9]The performance of 4 iterations of SGNS and 8 iterations of GloVe were much lower than those of 10 iterations and 20 iterations of SGNS and GloVe shown in Table 1, respectively.

[10]The right-truncatable embedding vectors used in our experiments will be available in author's homepage

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 19–27, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chris Alberti, David Weiss, Greg Coppola, and Slav Petrov. 2015. Improved Transition-Based Parsing and Tagging with Neural Networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1354–1359, Lisbon, Portugal, September. Association for Computational Linguistics.

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal Distributional Semantics. *J. Artif. Int. Res.*, 49(1):1–47, January.

Danqi Chen and Christopher Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-Based Dependency Parsing with Stack Long Short-Term Memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China, July. Association for Computational Linguistics.

Yoav Goldberg and Omer Levy. 2014. word2vec Explained: Deriving Mikolov et al.'s Negative-sampling Word-embedding Method. *CoRR*, abs/1402.3722.

Jochen Gorski, Frank Pfeuffer, and Kathrin Klamroth. 2007. Biconvex Sets and Optimization with Biconvex Functions: a Survey and Extensions. *Math. Meth. of OR*, 66(3):373–407.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. SimLex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation. *ArXiv e-prints*, August.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, pages 873–882. Association for Computational Linguistics.

Jingu Kim, Yunlong He, and Haesun Park. 2014. Algorithms for Nonnegative Matrix and Tensor Factorizations: a Unified View Based on Block Coordinate Descent Framework. *Journal of Global Optimization*, 58(2):285–319.

Jonghoon Kim, Francois Rousseau, and Michalis Vazirgiannis. 2015. Convolutional Sentence Kernel from Word Embeddings for Short Text Categorization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 775–780, Lisbon, Portugal, September. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Neural Word Embedding as Implicit Matrix Factorization. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association for Computational Linguistics*, 3.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, Lisbon, Portugal, September. Association for Computational Linguistics.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better Word Representations with Recursive Neural Networks for Morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113, Sofia, Bulgaria, August. Association for Computational Linguistics.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

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

George A. Miller and Walter G. Charles. 1991. Contextual Correlates of Semantic Similarity. *Language & Cognitive Processes*, 6(1):1–28.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning Word Embeddings Efficiently With Noise-contrastive Estimation. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2265–2273. Curran Associates, Inc.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Represqentation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A Word at a Time: Computing Word Relatedness Using Temporal Semantic Analysis. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 337–346, New York, NY, USA. ACM.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual Correlates of Synonymy. *Commun. ACM*, 8(10):627–633, October.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Jun Suzuki and Masaaki Nagata. 2015. A Unified Learning Framework of Skip-Grams and Global Vectors. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 186–191, Beijing, China, July. Association for Computational Linguistics.

Yuta Tsuboi. 2014. Neural Networks Leverage Corpus-wide Information for Part-of-speech Tagging. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 938–950, Doha, Qatar, October. Association for Computational Linguistics.

Di Wang and Eric Nyberg. 2015. A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 707–712, Beijing, China, July. Association for Computational Linguistics.

Kristian Woodsend and Mirella Lapata. 2015. Distributed Representations for Unsupervised Semantic Role Labeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2482–2491, Lisbon, Portugal, September. Association for Computational Linguistics.

Jie Zhou and Wei Xu. 2015. End-to-end Learning of Semantic Role Labeling using Recurrent Neural Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1127–1137, Beijing, China, July. Association for Computational Linguistics.

Geoffrey Zweig and Christopher J.C. Burges. 2011. The microsoft research sentence completion challenge. Technical Report MSR-TR-2011-129, Microsoft Research, December.