How to Generate a Good Word Embedding

Siwei Lai, Kang Liu, and Shizhu He, *National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences*

Jun Zhao, *National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences and University of Chinese Academy of Sciences*

Three critical components in training word embeddings are the model, corpus, and training parameters. By systemizing existing neural-network-based word embedding methods, it performs experimental comparison among these components using the same corpus.

Word embedding,[1,2] also known as distributed word representation,[3] can capture both the semantic and syntactic information of words from a large unlabeled corpus[4] and has attracted considerable attention from many researchers. In recent years, several models have been proposed, many of which have yielded state-of-the-art results in various natural language processing (NLP) tasks. Although these studies always claim that their models are better than previous ones based on various evaluation criteria, very little research compares existing word-embedding methods. This article focuses on this issue and presents a detailed analysis of several important features for training word embeddings via an experimental review, including model construction, training corpus, and parameter design. To the best of our knowledge, no such study has been performed previously (see the "Related Work in Word Embedding Comparison" sidebar).

To design an effective word-embedding method, we must first clarify the model construction. Almost all methods for training word embeddings are based on the same distributional hypothesis: words that occur in similar contexts tend to have similar meanings.[5,6] Given this hypothesis, different methods model the relationship between a target word w and its context c in a corpus in different ways, where w and c are embedded into vectors. Generally, existing methods vary in two major aspects in model construction: the relationship between the target word and its context, and the representation of the context. Table 1 shows a brief comparison of commonly used models.

In terms of the relationship between the target word and its context, the first five models are identical. They use an object similar to conditional probability $P(w|c)$, which predicts the target word w based on its context c. Ronan Collobert and Jason Weston (C&W)[2] use an object similar to joint probability, where the (w, c) pairs in the corpus are trained to obtain higher scores. In terms

1541-1672/16/$33.00 © 2016 IEEE
Published by the IEEE Computer Society

Related Work in Word Embedding Comparison

Regarding model comparison, we find that the work most similar to ours is by Joseph Turian and colleagues.[1] Their work compares the hierarchical log-bilinear (HLBL) model and Collobert and Weston (C&W) model for the named entity recognition (NER) and chunking tasks. They trained word embeddings on a relatively small corpus containing 63 million tokens and concluded that the two embedding methods resulted in similar improvements to these tasks. Marco Baroni and colleagues[2] compared count models (models based on a word-context co-occurrence matrix) and a predict model (the neural-network-based word embedding model; in their experiments, they used the continuous bag of words (CBOW) model) on several tasks involving the evaluation of words' semantic properties. They claimed that the predict model achieved significantly better performance than the count model in almost all semantic tasks. Jeffrey Pennington and colleagues[3] proposed the matrix-based Global Vectors (GloVe) model and claimed that it outperformed the CBOW and skip-gram models. This result is contrary to our findings. We believe that the difference can be predominantly attributed to different iteration strategies. Pennington and colleagues used the CBOW and skip-gram models with a single iteration, whereas 25 iterations have been performed for the GloVe model. Our experiment shows that performing multiple iterations can improve performance by a large margin.

Several works have demonstrated the influence of corpus size and domain when comparing corpora. Tomas Mikolov and colleagues[4] observed that larger training corpora could lead to better performance of the CBOW model on analogy tasks (*syn* and *sem*). Pennington and colleagues found that a larger corpus size is beneficial only for the *syn* task and not for the *sem* task when training with the GloVe model.

References

1. J. Turian, L. Ratinov, and Y. Bengio, "Word Representations: A Simple and General Method for Semi-supervised Learning," *Proc. Assoc. Computational Linguistics*, 2010, pp. 384–394.
2. M. Baroni, G. Dinu, and G. Kruszewski, "Don't Count, Predict! A Systematic Comparison of Context-Counting vs. Context-Predicting Semantic Vectors," *Proc. Assoc. Computational Linguistics*, 2014, pp. 238–247.
3. J. Pennington, R. Socher, and C.D. Manning, "GloVe: Global Vectors for Word Representation," *Proc. Empirical Methods Natural Language Processing*, 2014, pp. 1532–1543.
4. T. Mikolov et al., "Efficient Estimation of Word Representations in Vector Space," *Proc. Int'l Conf. Learning Representation*, 2013; http://arxiv.org/pdf/1301.3781.pdf.

Table 1. How models handle the relationship between the target word *w* and its context *c*, and how they use embeddings to represent context.

Model	Relation of *w*, *c*	Representation of *c*
Skip-gram[4]	*c* predicts *w*	One of *c*
Continuous bag of words (CBOW)[4]	*c* predicts *w*	Average
Order	*c* predicts *w*	Concatenation
Log-bilinear language (LBL) model[5]	*c* predicts *w*	Compositionality
Neural network language model (NNLM)[1]	*c* predicts *w*	Compositionality
Collobert and Weston (C&W)[2]	Scores *w*, *c*	Compositionality

of context representation, the models use four different types of methods, listed in ascending order of complexity from top to bottom in Table 1. Skip-gram[4] uses the simplest strategy, which is to choose one of the words from the window of the target word and utilize its embedding as the representation of context. Continuous bag of words (CBOW)[4] uses the average embedding of context words as context representation. Both methods ignore word order to accelerate the training process. However, Thomas Landauer[7] estimates that 20 percent of meaning in text comes from word order, and the remainder comes from word choice. Therefore, these two models could lose some critical information. In contrast, the order model uses the concatenation of context words' embeddings, which maintains word-order information. Furthermore, the log-bilinear language (LBL) model,[5] neural network language model (NNLM),[1] and C&W model add a hidden layer to the order model. Thus, these models use the semantic compositionality of context words as context representation. Based on the above analysis, we wish to know two things: Which model performs best? Which selection should we make in terms of the relationship between the target word and its context as well as the different types of context representation?

In addition, the ability to train precise word embeddings is strongly related to the training corpus. Different training corpora with different sizes and on different domains can considerably influence the final results. Thus, we additionally wish to know how corpus size and domain affect word-embedding performance.

Finally, training precise word embedding strongly depends on certain parameters, such as the number of iterations and the dimensionality of the embedding. In this regard, we wish to know two more things: How many iterations should be applied to obtain a sufficient word embedding while avoiding overfitting? What dimension should we choose to obtain a sufficiently good embedding?

To answer these questions objectively, we evaluate various word embeddings on eight tasks of three major types: calculating the word embedding's semantic properties, employing word embeddings as features for existing NLP tasks, and employing word embedding to initialize other neural network models. We argue that all current applications of word embeddings are covered by the types mentioned here. Through different tasks with different types of word embeddings, we attempt to determine which type of model design is most suitable for each particular task (the first question). Moreover, we vary the corpus size and domain to attempt to answer the second question.

Models

Before we compare the word embedding models via experiments, let's first describe and analyze them. We denote the embedding of word w by $\boldsymbol{e}(w)$.

Model Overview

We describe six representative models. These models are widely used in training word embedding and NLP tasks such as language model.

Neural network language model. Yoshua Bengio and colleagues[1] first proposed a neural network language model (NNLM) that simultaneously learns a word embedding and a language model. For each sample in the corpus, we maximize the log-likelihood of the probability of the last word, given the previous words. For example, for a sequence $w_1, w_2, \ldots, w_n$ in a corpus, we need to maximize the log-likelihood of

$$P(w_n|w_1, w_2, \ldots, w_{n-1}),$$

where we refer to the word to be predicted (w_n) as the target word. This model uses a concatenation of the previous words' embeddings as the input:

$$x = [\boldsymbol{e}(w_1), \ldots, \boldsymbol{e}(w_{n-2}), \boldsymbol{e}(w_{n-1})].$$

The model structure is a feed-forward neural network with one hidden layer:

$$\boldsymbol{h} = \texttt{tanh}(\boldsymbol{d} + H\boldsymbol{x})$$

$$\boldsymbol{y} = \boldsymbol{b} + U\boldsymbol{h},$$

where U is a transformation matrix, and $\boldsymbol{b}$ and $\boldsymbol{d}$ are bias vectors. The final step is to apply to a softmax layer to obtain the probability of the target word.

Log-bilinear language model. The log-bilinear language (LBL) model proposed by Andriy Mnih and Hinton[5] is similar to the NNLM. The LBL model uses a log-bilinear energy function that's almost equal to that of the NNLM and removes the nonlinear activation function `tanh`.

Collobert and Weston. The C&W model[2] trains only the word embedding and doesn't predict the target word. Instead, it combines the target word and its context and then scores them. The scoring function is a one-hidden-layer neural network. The input is the concatenation of the target word's and the context words' embeddings. The training object is to maximize the score of a corpus sequence while minimizing the score of a noise sequence pairwisely, or formally, to minimize

$$\max(0, 1 - s(w,c) + s(w', c)).$$

In the noise sequence, the target word w is replaced with a random word w' also from the vocabulary.

CBOW and skip-gram. The CBOW and skip-gram models[4] attempt to minimize computational complexity. CBOW uses the average embedding of the context words as context representation,

IEEE Intelligent Systems

IEEE Computer Society Publications Office
10662 Los Vaqueros Circle, PO Box 3014
Los Alamitos, CA 90720-1314

Lead Editor
Bonnie Wylie
b.wylie@computer.org

Editorial Management
Jenny Stout

Publications Coordinator
isystems@computer.org

Director, Products & Services
Evan Butterfield

Senior Manager, Editorial Services
Robin Baldwin

Digital Library Marketing Manager
Georgann Carter

Senior Business Development Manager
Sandra Brown

Senior Advertising Coordinator
Marian Anderson
manderson@computer.org

Submissions: For detailed instructions and formatting, see the author guidelines at www.computer.org/intelligent/author.htm or log onto *IEEE Intelligent Systems*' author center at Manuscript Central (www.computer.org/mc/intelligent/author.htm). Visit www.computer.org/intelligent for editorial guidelines.

Editorial: Unless otherwise stated, bylined articles, as well as product and service descriptions, reflect the author's or firm's opinion. Inclusion in *IEEE Intelligent Systems* does not necessarily constitute endorsement by the IEEE or the IEEE Computer Society. All submissions are subject to editing for style, clarity, and length.

Table 2. Formalized context representation for models that predict target words, assuming the target word is w_n and its context words are $w_1, w_2, \ldots, w_{n-1}$.

Model	Context representation
Skip-gram	$\mathbf{e}(w_i), 1 \le i \le n-1$
CBOW	$(\mathbf{e}(w_1) + \ldots + \mathbf{e}(w_{n-2}), \mathbf{e}(w_{n-1}))/(\boldsymbol{n} - 1)$
Order	$[\mathbf{e}(w_1), \ldots, \mathbf{e}(w_{n-2}), \mathbf{e}(w_{n-1})]$
LBL	$H[\mathbf{e}(w_1), \ldots, \mathbf{e}(w_{n-2}), \mathbf{e}(w_{n-1})]$
NNLM	$\mathtt{tanh}(\boldsymbol{d} + H[\mathbf{e}(w_1), \ldots, \mathbf{e}(w_{n-2}), \mathbf{e}(w_{n-1})]$

Table 3. Training settings for word embeddings.

Type	Setting
Model	GloVe, skip-gram, CBOW, order, LBL, NNLM, C&W
Corpus	Wiki: 100M, 1.6B
	NYT: 100M, 1.2B
	W&N: 10M, 100M, 1B, 2.8B
	IMDB: 13M
Parameters	Dimensionality: 10, 20, 50, 100, 200
	Fixed window size: 5

and skip-gram uses one of the context words as the representation of the context. Both models neglect word-order information and only use logistic regression to predict the target word.

Order. To analyze the use of word-order information, we introduce a virtual model called "order," whose complexity is between that of the CBOW and LBL models. This model maintains word order like LBL while removing the hidden layer like CBOW.

GloVe. In addition to the neural network approaches, another approach trains word embeddings based on a word-context matrix in which each row corresponds to a word and each column corresponds to a context. The element in the matrix is relevant to the co-occurrence number of instances of the corresponding word and context. These models are called count-based models.[8] The latest research on such matrix approaches is the Global Vectors (GloVe) model.[9]

Model Analysis

The models in the previous section vary in two major aspects: the relationship between the target word and its context, and the representation of the context.

Relation between target word and its context. Existing neural word-embedding models design the relation between target word and context in two different ways. Most models use context to predict the target word. In NNLM, the hidden layer $\boldsymbol{h}$ is the representation of context $\boldsymbol{c}$. The dimension of the transformation matrix U is $|V| \times |\boldsymbol{h}|$, in which $|V|$ is the size of the vocabulary, and $|\boldsymbol{h}|$ is the dimension of the hidden layer. Each row in matrix U can be regarded as a supplementary embedding of the corresponding word. We denote the supplementary embedding for word w as $\boldsymbol{e}'(w)$. So, in these models, each word in vocabulary has two embeddings, $\boldsymbol{e}(w)$ when w is in context and $\boldsymbol{e}'(w)$ when w is the target word. The energy function for word w is $\boldsymbol{e}'(w)^{\mathrm{T}}\boldsymbol{c}$. In contrast to the language model-based approaches, the C&W model puts the target word in the input layer and only maintains one embedding for each word. The energy function for word w is $A\boldsymbol{e}(w) + B\boldsymbol{c}$, in which A and B are transformation matrices, and $\boldsymbol{c}$ is the representation of the context. Therefore, the two types of models are considerably different.

Representation of context. The models that predict target words use different strate-gies to represent the context. Table 2 formalizes context representation.

Tasks

We evaluate the various word-embedding models in eight tasks grouped into three major types as follows.

Semantic Properties

Word-embedding models are designed based on the distributional hypothesis (words with similar meanings tend to have similar word embeddings). We include several classic tasks following Baroni and colleagues[8]:

- *ws*. The WordSim353 set[10] contains 353 word pairs. It was constructed by asking human subjects to rate the degree of semantic similarity or relatedness between two words on a numerical scale. The performance is measured by the Pearson correlation of the two word embeddings' cosine distance and the average score given by the participants.
- *tfl*. The TOEFL set[11] contains 80 multiple-choice synonym questions, each with 4 candidates. We choose the nearest neighbor of the question word from the candidates based on the cosine distance and use the accuracy to measure the performance.
- *sem and syn*. The analogy task[4] has approximately 9,000 semantic and 10,500 syntactic analogy questions. The questions are similar to "*man* is to (*woman*) as *king* is to *queen*" or "*predict* is to (*predicting*) as *dance* is to *dancing*." Following the previous work, we use the nearest

neighbor of ($queen - king + man$) in the vocabulary as the answer. Accuracy helps measure performance.

Embedding as Features

Word-embedding models capture useful information from unlabeled corpora. Many existing works have directly used word embeddings as features to improve the performance of certain tasks.

- The *avg* task uses the weighted average of the word embeddings as the representation of the text and subsequently applies a logistic regression to perform text classification. The weight for each word is its term frequency. We use the IMDB dataset.[12]
- The *ner* task uses word embeddings as the additional feature of the near state-of-the-art named entity recognition (NER) system.[3] The performance is evaluated with the F1 score on the testset of the CoNLL03 shared task dataset.

Embedding as the Initialization of Neural Networks

Dumitru Erhan and colleagues[13] have demonstrated that a better initialization can cause a neural network model to converge to a better local optimum. In recent neural network approaches to NLP tasks, word embeddings have been used to initialize the first layer. We choose two state-of-the-art research pieces in this part:

- We use convolutional neural networks (CNNs)[14] to perform sentence-level sentiment classification on the Stanford Sentiment Treebank dataset.[15] We repeat our experiment five times and report the average accuracy of these experiments.
- We use the neural network proposed by Ronan Collobert and colleagues[16] to perform part-of-speech (POS) tagging on *Wall Street Journal* data.[17]

Table 4. Best results for 50-dimensional embeddings trained by each model on the W&N corpus (2.8 billion tokens).

Model	syn	sem	ws	tfl	avg	ner	cnn	pos
Random	0.00	0.00	0.00	25.00	64.38	84.39	36.60	95.41
GloVe	40.00	27.92	56.47	**77.50**	74.51	88.19	43.29	96.42
Skip-gram	51.78	**44.80**	**63.89**	76.25	**74.94**	**88.90**	43.84	96.57
CBOW	**55.83**	44.43	62.21	**77.50**	74.68	88.47	43.75	96.63
Order	55.57	36.38	62.44	**77.50**	**74.93**	88.41	**44.77**	**96.76**
LBL	45.74	29.12	57.86	75.00	74.32	88.69	43.98	**96.77**
NNLM	41.41	23.51	59.25	71.25	73.70	88.36	44.40	96.73
C&W	3.13	2.20	46.17	47.50	73.26	88.15	41.86	96.66

Table 5. Nearest neighbors of selected words when trained with the CBOW and C&W models on the W&N corpus.

Model	Monday	Commonly	Reddish
CBOW	Thursday	generically	greenish
	Friday	colloquially	reddish-brown
	Wednesday	popularly	yellowish
	Tuesday	variously	purplish
	Saturday	commonly	brownish
C&W	8:30	often	purplish
	12:50	generally	pendulous
	1PM	previously	brownish
	4:15	have	orange-brown
	mid-afternoon	are	grayish

Experiments and Results

Table 3 shows our detailed experimental settings.

Performance Gain Ratio

To compare the models across the eight tasks, we need a unified evaluation metric, because existing evaluation metrics for each task vary in mean and variance (see Table 4). To address this problem, we propose a new indicator, the *performance gain ratio* (*PGR*), as the normalization of the original evaluation metrics.

The *PGR* of embedding a with respect to embedding b is defined as

$$PGR(a,b) = \frac{p_a - p_{rand}}{p_b - p_{rand}} \times 100\%,$$

where p_x is the performance of embedding x for a given task and p_{rand} is the performance achieved by a random embedding. Here, a random embedding is a word embedding of the same dimensionality as embedding x, where each dimension is a uniformly distributed random variable ranging from –1 to 1. Embedding b is chosen to be the best embedding for a given setting; thus, we simply call this measure the *PGR* of embedding a. With this definition, the *PGR* is a percentage. If $PGR = 100$ percent, then the embedding achieved the best result among all embeddings for the same setting. If $PGR = 0$, then the embedding might contain no more useful information than a random embedding. If $PGR <$ 0, then the embedding is detrimental to the task.

Table 6. Tasks for which a model "wins" on a certain corpus. In each cell, $a + b$ indicates that the model wins a tasks of the first four tasks (semantic properties) and b tasks of the last four tasks.

Model	10 million	100 million	1 billion	2.8 billion
Skip-gram	4 +2	4 +2	2 +2	3 +2
CBOW	1 +1	3 +3	4 +1	4 +1
Order	0 +2	1 +2	2 +3	3 +3
LBL	0 +2	0 +2	0 +2	1 +2
NNLM	0 +2	0 +3	0 +3	0 +2

Table 7. *PGR* values for the CBOW model trained on different corpora.

Corpus		Syn	sem	ws	tfl	avg	ner	cnn	pos
NYT	1.2B	93	52	90	98	50	76	85	96
	100M	76	30	88	93	46	77	83	86
Wiki	1.6B	92	**100**	**100**	93	51	**100**	86	94
	100M	74	65	98	93	47	88	90	83
W&N	2.8B	**100**	89	95	93	50	97	91	**100**
	1B	98	87	95	**100**	48	98	90	98
	100M	79	63	97	96	51	85	92	86
	10M	29	27	76	60	42	49	77	42
IMDB	13M	32	21	55	82	**100**	26	**100**	–13

Model Comparison

To fairly compare the different models, we use the same implementation for all of them. Our GloVe implementation is based on the GloVe toolkit (http://nlp.stanford.edu/projects/glove), and the CBOW and skip-gram implementations are based on the word2vec toolkit (https://code.google.com/p/word2vec). The other models are implemented by modifying the CBOW implementation in word2vec. For all models, we use the center word in the window as the target word. For the neural-network-based models (all models except GloVe), we use subsampling with $t = 10^{-4}$ and negative sampling with five negative samples.[18]

For each model, we iterate until the model converges to or overfits all the tasks. Table 4 shows the best performance for each model trained on the W&N corpus (a collection of articles from Wikipedia and *The New York Times*). Compared with random embedding, all investigated word embeddings demonstrate better performance on the tasks.

Relation between target word and context. To investigate the influence of the relationship between a target word and its context, we should compare the C&W model, which scores the combination of the target word and its context, with the other models, which predict the target word.

The C&W model exhibits a lower performance compared with the other models for the semantic property tasks (*syn*, *sem*, *ws*, and *tfl*). Particularly in the analogy tasks (*syn* and *sem*), the results show that the C&W model almost entirely lacks the feature of linear semantic subtraction.

To intuitively understand the difference between the two types of models, we show the nearest neighbors of selected words in Table 5. From these cases, we find that the nearest neighbors of "Monday" trained by the CBOW model are the other days of the week. In contrast, when trained by the C&W model, the nearest neighbors are times of day. The nearest neighbors of "commonly" also reveal similar results: the CBOW model finds words that can replace the word "commonly," whereas the C&W model will find words that are used together with "commonly." Most of the nearest neighbors of "reddish" in both models are colors, except for the word "pendulous" in the C&W model, which can be used with "reddish" to describe flowers.

Representation of context. To investigate the influence of context representation, we further compare embeddings trained on corpora of different scales.

Table 6 reports the number of tasks for which each model achieves 95 percent *PGR* (for convenience, we say the model "wins" in that case). A model that wins a task when trained on a 10 million-token corpus should be compared with the best embedding trained on the 10 million-token corpus. We only focus on the listed five models because the only difference between these models are their representations of context.

For a smaller corpus, a simpler model such as skip-gram can achieve better results, whereas for a larger corpus, more complex models, such as CBOW and order, are typically superior. For tasks in which word embeddings are used as features or to initialize neural networks, model choice doesn't significantly affect results. In addition, based on the results in Table 4, the margins between the simple and complex models are relatively small. Therefore, simpler models are typically sufficient for real tasks.

The Effect of the Training Corpus

We used two large-scale corpora and one small corpus with different domains in our experiments. These corpora are the Wikipedia dump (Wiki; https://dumps.wikimedia.org/enwiki), *The New York Times* (NYT) corpus (https://catalog.ldc.upenn.edu/LDC2008T19), and the IMDB corpus. We combine the Wiki and NYT corpora to obtain the W&N corpus. The vocabulary is set as the most frequent 200,000 words in both the Wiki and NYT corpora. For each corpus, the words not in the vocabulary are ignored. We shuffle the corpus at the document level to decrease the bias caused by certain online learning models. Small corpora are sampled uniformly from the larger corpus at the document level; thus, a 10 million-token corpus is a subset of the corresponding 100 million-token corpus.

We chose a representative CBOW model to analyze the influence of the corpus; other models yield similar results. Table 7 shows the *PGR* values for the embeddings trained on various corpora compared with the best results across all corpora. The best *PGR* for each task is 100 percent.

Corpus size. From the results in Table 7, we can conclude that using a larger corpus can yield a better embedding when the corpora are in the same domain. We compare the full-sized NYT corpus with its 100 million-token subset, the Wiki corpus with its subset, and the W&N corpus with its three subsets. In almost all cases, the larger corpus is superior to the smaller corpus. The observed exceptions could be attributable to instability in the evaluation metrics.

Specifically, in the *syn* task (analogy task with syntax test cases, such as "year:years law:—"), the corpus size is the main driver of performance. Clearly, different corpora use English in a similar manner, thereby producing similar syntax information.

Table 8. Nearest neighbors of certain words when trained on the IMDB and W&N corpora.

Corpus	Movie	Sci-Fi	Season
IMDB	film	SciFi	episode
	this	sci-fi	seasons
	it	fi	installment
	thing	sci	episodes
	miniseries	SF	series
W&N	film	Nickelodeon	half-season
	big-budget	cartoon	seasons
	movies	PBS	homestand
	live-action	SciFi	playoffs
	low-budget	TV	game

Table 9. *PGRs* of *avg* task when trained on mixed corpora.

W&N \ IMDB	0%	20%	40%	60%	80%	100%
+0%		91	94	100	100	100
+20%	32	79	87	91	96	99
+40%	41	68	86	88	92	98
+60%	41	65	79	85	88	93
+80%	41	64	75	84	87	92
+100%	42	64	70	83	86	88

Corpus domain. In most tasks, the influence of the corpus domain is dominant. In different tasks, it affects performances in different ways:

- In tasks involving the evaluation of semantic features, such as the analogy task with semantic test cases (*sem*) and the semantic similarity task (*ws*), the Wiki corpus is superior to the NYT corpus. Even the 100 million-token subset of the Wiki corpus can achieve a better performance than the 1.2 billion-token NYT corpus. We believe that the Wikipedia corpus contains more comprehensive knowledge, which could be beneficial for semantic tasks.
- The small IMDB corpus is beneficial for the *avg* and *cnn* tasks, but it performs very poorly in the *ner* and *pos* tasks. The IMDB corpus consists of movie reviews from the IMDB website, which is the same source as that used for the training and testsets of the *avg* and *cnn* tasks. An in-domain corpus is helpful for the tasks. Especially in the *avg* task, the in-domain IMDB corpus achieves a PGR almost twice that of the second-best one. On the other hand, these movie reviews are much more informal than the Wiki and NYT corpora, which is detrimental to the POS tagging (*pos*) task.

To intuitively illustrate how an in-domain corpus helps a given task, Table 8 shows several selected words and their nearest neighbors. The neighbors of "movie" in the IMDB corpus consist of words such as "this," "it," and "thing," implying that the word "movie" can be treated as a stop word in the IMDB corpus. The neighbors of "Sci-Fi" in the IMDB corpus consist of other abbreviations for "science fiction," whereas the neighbors in

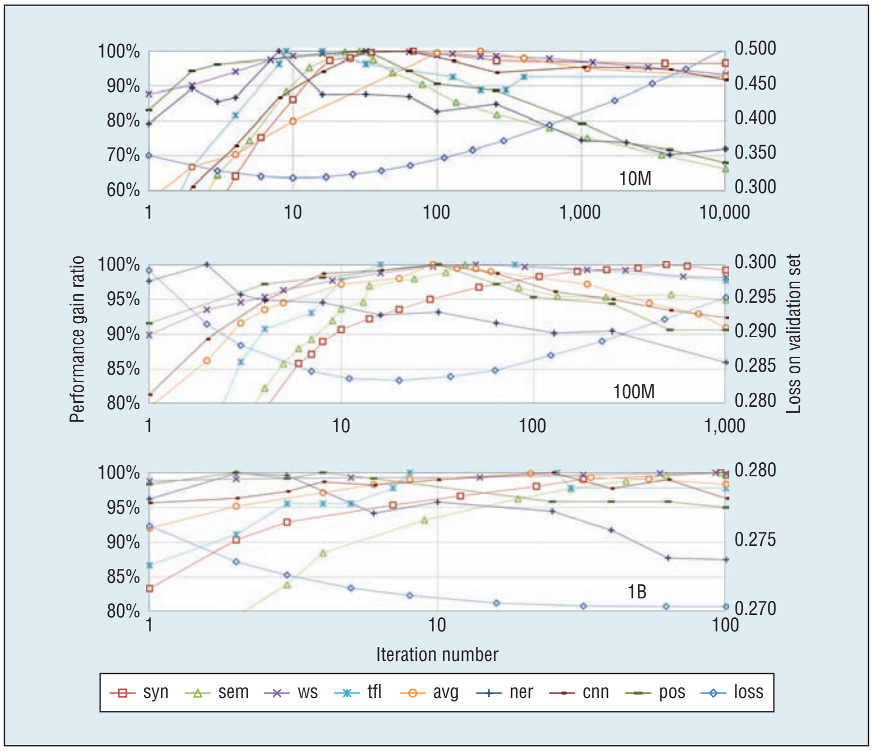

Figure 1. *Task PGRs* for embeddings trained with different numbers of iterations using the CBOW model on 10 million-, 100 million-, and 1 billion-token subsets of the W&N corpus.

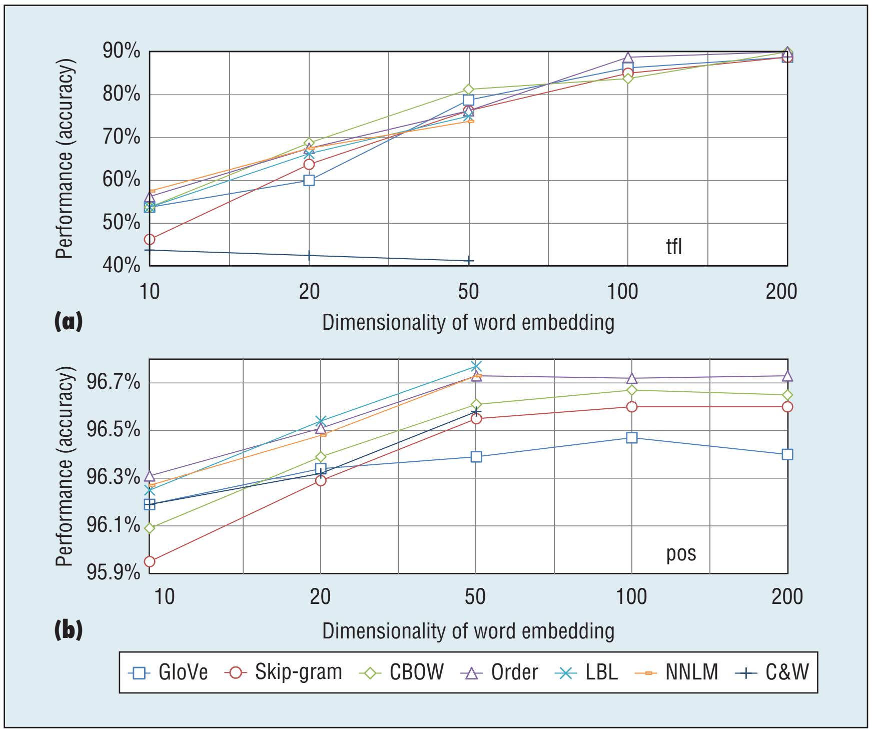

Figure 2. Task performances for embeddings of different dimensions trained using the various models on the 1 billion-token subset of the W&N corpus: the (a) tfl and (b) pos tasks.

the W&N corpus are predominantly other genres. The word “season” in the IMDB corpus is predominantly associated with the episodes of TV shows, whereas it refers to the sports competition season in the W&N corpus. From these cases, we observe that an in-domain corpus can improve task performance because it gives more suitable word embeddings.

Size or Domain?

So which aspect is more important for obtaining better word embeddings, corpus size or domain? Should we keep the corpus pure or add the out-domain corpus? To find out, we used the 13 million-token IMDB corpus and a subset of W&N corpus with 13 million tokens. The subsets of both corpora are mixed to train word embedding. Table 9 shows the *PGR* of the *avg* task for each embedding. For example, the entry with value 68 in the third row and the second column means we combine 20 percent of the IMDB corpus with 40 percent of the 13 million-token subset of the W&N corpus to train a word embedding and obtain a *PGR* of 68 on the *avg* task. For each column in the table, the elements are listed in ascending order of corpus size, as well as in descending order of the in-domain purity. The experiment results demonstrate that no matter which scale the IMDB corpus is, adding the W&N corpus consistently decreases the performance. The pure IMDB corpus is better than the larger mixed corpus.

From the results, we can see that corpus domain is more important than corpus size. Using an in-domain corpus significantly improves performance for a given task, whereas using a corpus in an unsuitable domain can decrease performance. For specific tasks, a pure in-domain corpus yields better performance than a mixed-domain corpus. For corpora in the same domain, a larger corpus will achieve better performance.

Training Parameters

The number of training iterations and the dimensionality of word embedding are two important hyper-parameters in training word embedding.

Number of iterations. In machine learning, early stopping is a form of regularization used to tackle with the over-fitting problem. The most widely used early stopping method is to stop the iterator when the loss on the validation set peaks. In word-embedding training, the loss measures how well the model predicts the target word. However, real tasks typically don't consist of predicting target words. Therefore, the loss on the validation set is only a proxy for these real tasks, and in some cases, can be inconsistent with task performance. Therefore, it's worth investigating whether the traditional early stopping method is a good metric for stopping the training of word embeddings.

In our experiment, we used 95 percent of the corpus as the training set and the remaining 5 percent as the validation set. Figure 1 shows three examples of the training procedure.

From these cases, we find that the loss on the validation set is inconsistent with the performance in NLP tasks. It's also worth noting that most tasks always exhibit similar peaks. The results demonstrate that we can use a simple task to verify whether word embedding has peaked on other tasks. We consider the various combinations of the eight tasks, the seven models, and the three subsets of the W&N corpus, thereby obtaining 168 tuples. If we use a strategy that stops at the peak of the loss on the validation set, we win (achieve 95 percent performance with respect to the peak performance on the task) in 89 cases. If we use the strategy that stops at the peak of the *tfl* task (the simplest task in our experiments), we win in 117 cases.

When training a word embedding for a specific task, using the development set for the task to determine the stopping iteration is the optimal choice because it yields the results that are most consistent with the task's final performance. However, in certain situations, testing the development set performance is time-consuming, and our strategy can be used as a fast approximation of peak performance.

Therefore, we could conclude that iterating the model until it peaks on some simple task will yield sufficient embeddings for most tasks. To train a more suitable word embedding for specific tasks, we can use a development set for that task to decide when to stop iterating.

Dimensionality. To investigate the influence of dimensionality on word-embedding training, we compared models of different dimensions in the eight evaluation tasks. Interestingly, the results show that all tasks that involve analyzing an embedding's semantic properties behave in a similar manner. Figure 2a shows the performance for the *tfl* task as an example. Moreover, the tasks in which the embedding is used as a feature or for initialization also behave in a similar manner. Figure 2b shows the performance for the *pos* task as an example.

We find that for the semantic property tasks, larger dimensions will lead to better performance (except for with the C&W model). However, for NLP tasks, a dimensionality of 50 is typically sufficient.

Although we didn't find any specific settings for training a word embedding that can yield the best performance on all tasks (and indeed, such settings might not exist), we provide several guidelines for training a good word embedding:

- For model construction, more complex models require a larger training corpus to outperform simpler models. Faster (simpler) models are sufficient in most cases, but in semantic tasks, models that predict the target word (the first five models

THE AUTHORS

Siwei Lai is a PhD student in the National Laboratory of Pattern Recognition at the Institute of Automation, Chinese Academy of Sciences. His research interests include deep learning and text representation. Contact him at swlai@nlpr.ia.ac.cn.

Kang Liu is an associate professor in the National Laboratory of Pattern Recognition at the Institute of Automation, Chinese Academy of Sciences. His research interests include opinion mining, information extraction, and machine learning. Liu has a PhD in pattern recognition and intelligent systems from the Chinese Academy of Sciences. Contact him at kliu@nlpr.ia.ac.cn.

Shizhu He is a PhD student in the National Laboratory of Pattern Recognition at the Institute of Automation, Chinese Academy of Sciences. His research interests include question answering and knowledge base embedding. He has an MS in computer application technology from Jiangxi Normal University. Contact him at shizhu.he@nlpr.ia.ac.cn.

Jun Zhao is a professor in the National Laboratory of Pattern Recognition at the Institute of Automation, Chinese Academy of Sciences. His research interests include information extraction and question answering. Zhao has a PhD in computer science and technology from Tsinghua University. Contact him at jzhao@nlpr.ia.ac.cn.

in Table 1) demonstrate higher performance than one that scores the combination of the target word and its context (the C&W model).

- When choosing a corpus in a suitable domain, using a larger corpus is better. Training on a large corpus generally improves the quality of word embeddings, and training on an in-domain corpus can significantly improve the quality of word embeddings for a specific task. More importantly, we can see that corpus domain is more important than corpus size.
- Validation loss isn't a good early stopping metric for training word embeddings; rather, the best approach is to check the performance of the development set for that task. If evaluating the task is time-consuming, the performance for other tasks can be used as an alternative.
- For tasks that analyze the semantic properties of a word embedding, larger dimensions can provide better performance. For NLP tasks that utilize an embedding as a feature or for initialization, a dimensionality of 50 is sufficient.

We determined that the domain of a training corpus is very important in achieving good performance, but we believe that the data source (such as monolingual corpus, multilingual corpus and knowledge bases) used in the word embedding might also be important to this end. ■

Acknowledgments

This work was supported by the National Natural Science Foundation of China (no. 61533018), the National Basic Research Program of China (no. 2014CB340503), and the National Natural Science Foundation of China (no. 61272332).

References

1. Y. Bengio et al., "A Neural Probabilistic Language Model," *J. Machine Learning Research*, vol. 3, 2003, pp. 1137–1155.
2. R. Collobert and J. Weston, "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning," *Proc. Int'l Conf. Machine Learning*, 2008, pp. 160–167.
3. J. Turian, L. Ratinov, and Y. Bengio, "Word Representations: A Simple and General Method for Semi-supervised Learning," *Proc. Assoc. Computational Linguistics*, 2010, pp. 384–394.
4. T. Mikolov et al., "Efficient Estimation of Word Representations in Vector Space," *Proc. Int'l Conf. Learning Representation*, 2013, arXiv:1301.3781.
5. A. Mnih and G. Hinton, "Three New Graphical Models for Statistical Language Modelling," *Proc. Int'l Conf. Machine Learning*, 2007, pp. 641–648.
6. Z.S. Harris, "Distributional Structure," *Word*, vol. 10, no. 2, 1954, pp. 146–162.
7. T.K. Landauer, "On the Computational Basis of Learning and Cognition: Arguments from LSA," *Psychology of Learning and Motivation*, vol. 41, 2002, pp. 43–84.
8. M. Baroni, G. Dinu, and G. Kruszewski, "Don't Count, Predict! A Systematic Comparison of Context-Counting vs. Context-Predicting Semantic Vectors," *Proc. Assoc. Computational Linguistics*, 2014, pp. 238–247.
9. J. Pennington, R. Socher, and C.D. Manning, "GloVe: Global Vectors for Word Representation," *Proc. Empirical Methods in Natural Language Processing*, 2014, pp. 1532–1543.
10. L. Finkelstein et al., "Placing Search in Context: The Concept Revisited," *ACM Trans. Information Systems*, vol. 20, no. 1, 2002, pp. 116–131.
11. T.K. Landauer and S.T. Dumais, "A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge," *Psychological Rev.*, vol. 104, no. 2, 1997, p. 211.
12. A.L. Maas et al., "Learning Word Vectors for Sentiment Analysis," *Proc. Assoc. Computational Linguistics*, 2011, pp. 142–150.
13. D. Erhan et al., "Why Does Unsupervised Pre-training Help Deep Learning?," *J. Machine Learning Research*, vol. 11, 2010, pp. 625–660.
14. Y. Kim, "Convolutional Neural Networks for Sentence Classification," *Proc. Empirical Methods Natural Language Processing*, 2014, pp. 1746–1751.
15. R. Socher et al., "Recursive Deep Models for Semantic Compositionality over a Sentiment Treebank," *Proc. Empirical Methods in Natural Language Processing*, 2013, pp. 1631–1642.
16. R. Collobert et al., "Natural Language Processing (almost) from Scratch," *J. Machine Learning Research*, vol. 12, no. 8, 2011, pp. 2493–2537.
17. K. Toutanova et al., "Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network," *Proc. Annual Conf. North Am. Chapter Assoc. Computational Linguistics: Human Language Technologies*, 2003, pp. 173–180.
18. T. Mikolov et al., "Distributed Representations of Words and Phrases and Their Compositionality," *Proc. Neural Information Processing Systems*, 2013, pp. 3111–3119.