Sparse Word Embeddings Using ℓ_1 Regularized Online Learning

Fei Sun, Jiafeng Guo, Yanyan Lan, Jun Xu, and **Xueqi Cheng**
CAS Key Lab of Network Data Science and Technology
Institute of Computing Technology, Chinese Academy of Sciences, China
ofey.sunfei@gmail.com, {guojiafeng, lanyanyan, junxu, cxq}@ict.ac.cn

Abstract

Recently, `Word2Vec` tool has attracted a lot of interest for its promising performances in a variety of natural language processing (NLP) tasks. However, a critical issue is that the dense word representations learned in `Word2Vec` are lacking of interpretability. It is natural to ask if one could improve their interpretability while keeping their performances. Inspired by the success of sparse models in enhancing interpretability, we propose to introduce sparse constraint into `Word2Vec`. Specifically, we take the Continuous Bag of Words (CBOW) model as an example in our study and add the ℓ_l regularizer into its learning objective. One challenge of optimization lies in that stochastic gradient descent (SGD) cannot directly produce sparse solutions with ℓ_1 regularizer in online training. To solve this problem, we employ the Regularized Dual Averaging (RDA) method, an online optimization algorithm for regularized stochastic learning. In this way, the learning process is very efficient and our model can scale up to very large corpus to derive sparse word representations. The proposed model is evaluated on both expressive power and interpretability. The results show that, compared with the original CBOW model, the proposed model can obtain state-of-the-art results with better interpretability using less than 10% non-zero elements.

1 Introduction

Word embedding aims to encode semantic meanings of words into low-dimensional dense vectors. Recently, neural word embeddings have attracted a lot of interest for their promising results in various natural language processing (NLP) tasks *e.g.*, language modeling [Bengio *et al.*, 2003], named entity recognition [Collobert *et al.*, 2011], and parsing [Socher *et al.*, 2013]. Among all the neural embedding approaches, CBOW and Skip Gram (SG) [Mikolov *et al.*, 2013a], implemented in the `Word2Vec` tool, are two state-of-the-art methods due to their simplicity, effectiveness and efficiency.

However, for `Word2Vec`, a critical issue is that the dense representations they derived are lacking of interpretability. We do not know which dimension in word vectors represent the gender of "*man*" and "*woman*", and also do not know what sort of value indicates "*male*" or "*female*". This makes dense representations as a black-box. Moreover, even if there exists some dimension corresponding to the gender information, such dimension would be active in all the word vectors including irrelevant words like "*parametric*", "*stochastic*", and "*bayesian*", which is very difficult in interpretation and uneconomic in storage.

Therefore, a natural question is that: can we improve the interpretability of `Word2Vec` while keeping their promising performances?

In this paper, we argue that sparsification is a possible answer for this question. In other domains (*e.g.*, image processing and computer vision), sparse representations have already been widely used as a way to increase interpretability [Olshausen and Field, 1997; Lewicki and Sejnowski, 2000]. For word representations, Murphy *et al.* [2012] improved dimension interpretability by introducing non-negative and sparse constraints into matrix factorization (NNSE). Recently, Faruqui *et al.* [2015] verified the effectiveness of sparsity under `Word2Vec` in a post-processing way. They converted the dense word vectors derived from `Word2Vec` using sparse coding (SC) and showed the resulting word vectors are more similar to the interpretable features used in NLP. However, SC usually suffers from heavy memory usage since they require a global matrix. This makes it quite difficult to train SC on large-scale text data. Since `Word2Vec` can easily scale up to large-scale raw text data, is it possible to directly derive sparse word representations under the `Word2Vec` framework?

In this paper, unlike SC introducing sparsity in a post-processing stage, we propose to directly applying the sparse constraint to `Word2Vec`. Specifically, we take CBOW as an example to conduct the study. A natural way to produce sparse representations is to add an ℓ_1 regularizer on the word vectors. This is non-trivial in CBOW since the online optimization with stochastic gradient descent (SGD) cannot directly produce the sparse solutions for ℓ_1 regularizer. To solve this issue, we employ the Regularized Dual Averaging (RDA) alogrithm [Xiao, 2009] to optimize ℓ_1 regularized loss function of CBOW in online learning. In this way, we can efficiently learn sparse word representations from large-scale raw text data on the fly.

We evaluate our model on both expressive power and interpretability. For expressive power, we evaluate the learned representations on two tasks, word similarity and word analogy. The results show that the proposed sparse model can achieve competitive performance with the state-of-the-art models under the same setting. Furthermore, our method also outperforms other sparse representation models significantly. For interpretability, we introduce a new evaluation metric for word intrusion task to get rid of human evaluation. Experimental results demonstrate the effectiveness of our sparse representations in comparison with the dense representations.

2 Related Work

Representing words as continuous vectors in a low-dimensional space dates back several decades [Hinton *et al.*, 1986]. Based on the distributional hypothesis [Harris, 1954; Firth, 1957], various methods have been developed in the NLP community, including matrix factorization [Deerwester *et al.*, 1990; Murphy *et al.*, 2012; Faruqui *et al.*, 2015; Pennington *et al.*, 2014] and neural networks [Bengio *et al.*, 2003; Collobert and Weston, 2008]. According to the constraint on the representations, we group the existing models into two categories, *i.e.*, dense word representation models and sparse word representation models.

2.1 Dense Word Representation Models

Inspired by the success in deep learning for NLP, there has been a flurry of subsequent work exploring various neural network structures and optimization methods to represent words as low-dimensional dense continuous vector [Bengio *et al.*, 2003; Collobert and Weston, 2008; Mikolov *et al.*, 2013a; Mnih and Kavukcuoglu, 2013; Mikolov *et al.*, 2013b]. Among all these neural embedding approaches, CBOW and SG are two state-of-the-art methods due to their simplicity, effectiveness and efficiency.

Besides, low-rank decomposition and spectral methods are also popular choices to learn dense word representations. LSA [Deerwester *et al.*, 1990] used Singular Value Decomposition (SVD) to factorize the word-document matrix to acquire continuous word representations. GloVe [Pennington *et al.*, 2014] factorized a log-transformed word-context co-occurrence matrix. Canonical Correlation Analysis (CCA) also provided a powerful tool to derive the word representations [Dhillon *et al.*, 2011; Stratos *et al.*, 2015]. Levy and Goldberg [2014b] showed the connection between matrix factorization and Skip Gram with negative sampling.

Because of its advantage over traditional one-hot (local) representation, the dense vectors learned by these models have been successfully used in various natural language processing tasks, *e.g.*, language modeling [Bengio *et al.*, 2003], named entity recognition [Collobert *et al.*, 2011], and parsing [Socher *et al.*, 2013].

2.2 Sparse Word Representation Models

Dense word representations have dominated the NLP community because of their effectiveness in a variety of NLP tasks. Nonetheless, they are usually criticized for lacking of interpretability and extravagance of storage [Griffiths *et al.*, 2007]. On the contrary, sparse representation is considered as a potential choice for interpretable word representations. It is believed that human brain represents the information in a sparse way. For example, in human vision, neurons in the primary visual cortex (V1) are believed to have a distributed and sparse representation [Olshausen and Field, 1997; Attwell and Laughlin, 2001]. In human language, Vinson and Vigliocco [2008] showed that the gathered descriptions for a given word are typically limited to approximately $20-30$ features in feature norming[1].

In practice, sparse overcomplete representations have been widely used as a way to improve separability and interpretability in image processing and computer vision [Olshausen and Field, 1997; Lewicki and Sejnowski, 2000]. There have been some work trying to explore sparse word representations. Murphy *et al.* [2012] improved the interpretability of word vectors by introducing sparse and non-negative constraints into matrix factorization. Lately, Faruqui *et al.* [2015] converted dense word vectors derived from any state-of-the-art word vector model (*e.g.*, CBOW or SG) into sparse vectors using sparse coding and showed the resulting word vectors are more similar to the interpretable features typically used in NLP tasks comparing with original dense word vectors.

3 Our Approach

In this section, we take CBOW as an example to conduct the study. We first briefly introduce CBOW and then elaborate the proposed sparse representations model. It is easy to apply the sparse constraint to SG model using the same strategy elaborated in this section.

3.1 Notation

First of all, We list the notations used in this paper. Let $\mathcal{C}=[w_1, \ldots, w_N]$[2] denotes a corpus of N word sequence over the word vocabulary W. The contexts for word $w_i \in W$ (*i.e.*, i-th word in corpus) are the words surrounding it in an l-sized window $(c_{i-l}, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{i+l})$, where $c_j \in C$, $j \in [i-l,\ i+l]$. Each word $w \in W$ and context $c \in C$ are associated with vectors $\vec{w} \in \mathbb{R}^d$ and $\vec{c} \in \mathbb{R}^d$ respectively, where d is the representation dimensionality. In this paper, $\vec{x}$ denotes the vector of the variable x unless otherwise specified. The entries in the vectors are treated as parameters to be learned.

3.2 CBOW

Continuous Bag-of-Words (CBOW) is a simple and effective state-of-the-art word representation model [Mikolov *et al.*, 2013a]. It aims to predict the target word using context words in a sliding window.

Formally, given a word sequence $\mathcal{C}$, the objective of CBOW is to maximize the following log-likelihood:

$$\mathcal{L}_{cbow} = \sum_{i=1}^{N} \Big(\log p(w_i | h_i) \Big)$$

[1]It is a task that participants are asked to list the properties of a word.

[2]It is worth noting that w_i and w_j in corpus $\mathcal{C}$ could be the same word w in the vocabulary W.

where h_i denotes the combination of w_i's contexts.

We use softmax function to define the probabilities $p(w_i|h_i)$ as follows:

$$p(w_i|h_i) = \frac{\exp(\vec{w}_i \cdot \vec{h}_i)}{\sum_{w \in W} \exp(\vec{w} \cdot \vec{h}_i)}$$

where $\vec{h}_i$ denotes the projected vectors of w_i's contexts. It is defined as the average of all context word vectors in CBOW[3]:

$$\vec{h}_i = \frac{1}{2l} \sum_{\substack{j=i-l \\ j \neq i}}^{i+l} \vec{c}_j$$

3.3 Sparse CBOW

In order to learn sparse word representations, a straightforward way is to introduce the sparse constraint, *e.g.*, the ℓ_1 regularizer, on word vectors. In this way, we obtain the new objective function as follows:

$$\mathcal{L}_{s-cbow} = \mathcal{L}_{cbow} - \lambda \sum_{w \in W} \|\vec{w}\|_1$$

where λ is the hyperparameter that controls the degree of regularization.

As we know, the optimization of `word2vec` is in an online fashion using stochastic gradient descent as in [Mikolov *et al.*, 2013b], which makes it very efficient in learning. However, a main drawback of directly applying stochastic subgradient descent to an ℓ_1 regularized objective in online training is that it will not produce a sparse solution. That is because the approximate gradient of SGD used at each update is very noisy and the value of each entry in the vector can be easily moved away from zero by those fluctuations. Fortunately, there have been several studies concerning the online optimization algorithms that target such ℓ_1 regularized objectives [Langford *et al.*, 2009; Duchi and Singer, 2009; Xiao, 2009; McMahan and Streeter, 2010]. In this paper, we propose to employing the Regularized Dual Averaging (RDA) algorithm [Xiao, 2009] to produce the sparse representations.

3.4 Optimization Details

The RDA method keeps track of the online average subgradients at time t : $\bar{g}^t = \sum_{t'=1}^{t} g^{t'}$. Here, the subgradient $g^{t'}$ at time t' does not include the regularization term ($\lambda = 0$). For Sparse CBOW, we use $g^t_{\vec{w}_i}$ to denote the subgradient with respect to $\vec{w}_i$ at time t.

However, the derivatives of $\mathcal{L}_{cbow}$ include high computational complexity normalization terms. For efficient learning, we employ the negative sampling technique [Mikolov *et al.*, 2013b] to approximate the original softmax function. It actually defines an alternative training objective function as follows:

$$\mathcal{L}^{ns}_{cbow} = \sum_{i=1}^{N} \Big(\log \sigma(\vec{w}_i \cdot \vec{h}_i) + k \cdot \mathbf{E}_{\tilde{w} \sim P_{\tilde{W}}} \log \sigma(-\vec{\tilde{w}} \cdot \vec{h}_i) \Big)$$

[3]It can also be sum, average, concatenate, max pooling, etc.

Algorithm 1 RDA algorithm for Sparse CBOW

```
 1: procedure SparseCBOW($\mathcal{C}$)
 2:     Initialize: $\vec{w}, \forall w \in W,\ \vec{c}, \forall c \in C,\ \bar{g}^0_{\vec{w}} = \vec{0}, \forall w \in W$
 3:     for $i = 1, 2, 3, \ldots$ do
 4:         $t \leftarrow$ update time of word $w_i$
 5:         $\vec{h}_i = \frac{1}{2l} \sum_{j=i-l, j \neq i}^{i+l} \vec{c}_j$
 6:         $g^t_{\vec{w}_i} = \left[ \mathbb{1}_h(w_i) - \sigma(\vec{w}^t_i \cdot \vec{h}_i) \right] \vec{h}_i$
 7:         $\bar{g}^t_{\vec{w}_i} = \frac{t-1}{t} \bar{g}^{t-1}_{\vec{w}_i} + \frac{1}{t} g^t_{\vec{w}_i}$
 8:         Update $\vec{w}_i$ element-wise according to
 9:         $\vec{w}^{t+1}_{ij} = \begin{cases} 0 & \text{if } |\bar{g}^t_{\vec{w}_{ij}}| \leq \frac{\lambda}{\#(w_i)}, \\ \eta t \left( \bar{g}^t_{\vec{w}_{ij}} - \frac{\lambda}{\#(w_i)} \operatorname{sgn}(\bar{g}^t_{\vec{w}_{ij}}) \right) & \text{otherwise,} \end{cases}$
            where, $j = 1, 2, \ldots, d$
10:         for $k = -l, \ldots, -1, 1, \ldots, l$ do
11:             update $\vec{c}_{i+k}$ according to
12:             $\vec{c}_{i+k} := \vec{c}_{i+k} + \frac{\alpha}{2l} \left[ \mathbb{1}_{h_i}(w_i) - \sigma(\vec{w}^t_i \cdot \vec{h}_i) \right] \vec{w}^t_i$
13:         end for
14:     end for
15: end procedure
```

where $\sigma(x) = 1/(1 + \exp(-x))$, $P_{\tilde{W}}$ denotes the distribution[4] of sampled *negative word* $\tilde{w}$ (*i.e.*, random sampled word which is not relevant with current contexts), and k is the number of negative samples. Negative sampling transforms the computationally expensive multi-class classification problem into a binary classification problem which can be regarded as to distinguish the correct word w_i from the random sampled words.

With the negative sampling, the subgradient of the positive/negative word w_i at time t given contexts h_i is:

$$g^t_{\vec{w}_i} = \left[\mathbb{1}_{h_i}(w_i) - \sigma(\vec{w}^t_i \cdot \vec{h}_i) \right] \vec{h}_i$$

where $\mathbb{1}_h(w)$ is an indicator function whether w is the right word in context h or not, and $\vec{w}^t_i$ denotes the vector for word w_i at time t.

Following the RDA algorithm, the update procedure for vectors $\vec{w}_i$ and $\vec{c}_i$ is shown in Algorithm 1. Specifically, we first initialize each word vector $\vec{w}$ and context vector $\vec{c}$ randomly using the same scheme as in `Word2Vec`. Then, the subgradient of word vector $\vec{w}_i$ at time t is computed as shown in line 6 and its online average subgradients $\bar{g}^t_{\vec{w}_i}$ is computed in line 7. We update each entry of $\vec{w}_i$ according to line 9, where η is the learning rate, $\operatorname{sgn}(\cdot)$ is a sign function, $\vec{w}_{ij}$ denotes the j-th entry of word vector $\vec{w}_i$, and $\bar{g}^t_{\vec{w}_{ij}}$ is the corresponding average subgradient at time t. Context vector $\vec{c}$ is updated according to line 12, where α is the learning rate. We adopt the same linear learning rate schedule described in [Mikolov *et al.*, 2013a], decreasing it linearly to zero at the end of the last training epoch.

[4]It is defined as $p_{\tilde{W}}(w) \propto \#(w)^{0.75}$, where $\#(w)$ means the number of word w appearing in corpus $\mathcal{C}$.

Table 1: Summary of results. We report precision (%) for word analogy task and spearman correlation coefficient for the word similarity task. Higher values are better. Bold scores are the best.

Model	Dim	Sparsity	Semantic	Syntactic	Total	WS-353	SL-999	RW	Average[‡]
GloVe	300	0%	79.31	61.48	69.57	59.18	32.35	34.13	48.81
CBOW	300	0%	**79.38**	**68.80**	**73.60**	67.21	38.82	45.19	56.21
SG	300	0%	77.79	67.32	72.09	**70.74**	36.07	**45.55**	56.11
PPMI(W-C)	40,000	86.55%	74.02	38.99	53.02	62.35	24.10	30.45	42.48
PPMI(W-C)	388,723	99.61%	58.55	31.19	43.60	58.99	23.01	27.98	38.40
NNSE (PPMI)[†]	300	89.15%	29.89	27.68	28.56	68.61	27.60	41.82	41.65
SC (CBOW)[⋆]	300	88.34%	28.99	28.43	28.68	59.85	30.44	38.75	39.43
SC (CBOW)[⋆]	3000	95.85%	74.71	61.24	67.35	68.22	39.12	44.75	54.61
Sparse CBOW	300	90.06%	73.24	67.48	70.10	68.29	**44.47**	42.30	**56.29**

[†] The input matirx of NNSE is the 40000-dimensional representations of PPMI in fourth row.

[⋆] The input matirx of SC is the 300-dimensional representations of CBOW in second row.

[‡] The average performance is calculated across the four different datasets/tasks (one for word analogy and three for word similarity), following the way used in [Faruqui *et al.*, 2015].

4 Experiments

In this section, we investigate the expressive power[5] and interpretability of our Sparse CBOW model by comparing with baselines including both dense and sparse models. Firstly, we describe our experimental settings including the corpus, hyper-parameter selections, and baseline methods. Then we evaluate expressive power of all models on two tasks, *i.e.*, word analogy and word similarity. After that, we test the interpretability using word intrusion task and case study.

4.1 Experimental Settings

We take the widely used Wikipedia April 2010 dump[6] [Shaoul and Westbury, 2010] as the corpus to train all the models. It contains 3,035,070 articles and about 1 billion words. We preprocess the corpus in a common way by lowercasing the corpus and removing pure digit words and non-English characters. During training, the words occurring less than 20 times are ignored, resulting in a vocabulary of 388,723 words. Following the practice in [Mikolov *et al.*, 2013b; Pennington *et al.*, 2014], we set the context window size as 10 and use 10 negative samples. Like CBOW, we set the initial learning rate of Sparse CBOW model as $\alpha = 0.05$ and decrease it linearly to zero at the end of the last training epoch. For the ℓ_1 regularization penalty λ, we perform a grid search on it and select the value that maximizing performance on one development testset (a small subset of WordSim-353[7]) while achieving at least 90% sparsity in word vectors.

We compare our model with two classes of baselines:

- Dense representation models: GloVe [Pennington *et al.*, 2014], CBOW, and SG [Mikolov *et al.*, 2013a].
- Sparse representation models: sparse coding (SC) [Faruqui *et al.*, 2015], positive pointwise mutual information (PPMI), and NNSE [Murphy *et al.*, 2012].

For GloVe[8], CBOW[9], and SG[9], we train them using the released tools on the same corpus with the same setting as our models for fair comparison. For SC[10], we use the result matrix of CBOW as its initial matrix. The PPMI matrix is built based on the word-context co-occurrence counts with window size as 10. Similar to [Murphy *et al.*, 2012], we implement NNSE based on word-context co-occurrence PPMI matrix using the `SPAMS` package[11]. Due to the memory issue, the PPMI matrix for NNSE is built over a vocabulary of 40,000 most frequent words just as the same setting in [Murphy *et al.*, 2012].

4.2 Expressive Power

To evaluate the expressive power of the representations of each model, we conduct experiments on two tasks, *i.e.*, word analogy and word similarity.

Word Analogy. The word analogy task is introduced by Mikolov *et al.* [2013c; 2013a] to quantitatively evaluate the models ability of encoding the linguistic regularities between word pairs. The dataset contains 5 types of semantic analogies and 9 types of syntactic analogies[12]. The semantic analogy contains 8869 questions, typically about places and people, like "*Athens* is to *Greece* as *Paris* is to *France*", while the syntactic analogy contains 10,675 questions, mostly focusing on the morphemes of adjective or verb tense, such as "*run* is to *running* as *walk* to *walking*".

This task is to assume the last word is missing (*e.g.*, "a is to b as a' is to __") and to correctly predict it. It is answered using 3CosMul for performance concern [Levy and Goldberg, 2014a]:

$$\arg \max_{x \in W \setminus \{a,b,a'\}} \frac{\text{sim}(x,b)\text{sim}(x,a')}{2\text{sim}(x,a)+\epsilon}$$

where ϵ is used to prevent division by zero and $\text{sim}(x,y)$ computes the similarity between word x and y. It is defined

[5]We focus on the intrinsic evaluation task since different extrinsic tasks favor different embeddings [Schnabel *et al.*, 2015]

[6]`http://www.psych.ualberta.ca/~westburylab/downloads/westburylab.wikicorp.download.html`

[7]Set1 in `http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/`

[8]`http://nlp.stanford.edu/projects/glove/`

[9]`https://code.google.com/p/word2vec/`

[10]`https://github.com/mfaruqui/sparse-coding`

[11]`http://spams-devel.gforge.inria.fr`

[12]`http://code.google.com/p/word2vec/source/browse/trunk/questions-words.txt`

as $\mathrm{sim}(x, y) = (\cos(x, y)+1)/2$ due to the non-negative constraint for similarity in 3CosMul. The prediction is judged as correct only if x is exactly the missing word in the evaluation set. The evaluation metric for this task is the percentage of questions answered correctly.

Word Similarity. The word similarity task is employed to measure how well the model captures the similarity between two words. We evaluate our model on three different testsets: (1) **WordSim-353 (WS-353)** [Finkelstein *et al.*, 2002], it is the most commonly used testset for semantic models, and consists of 353 pairs of English words; (2) **SimLex-999 (SL-999)** [Hill *et al.*, 2015], it is constructed to overcome the shortcomings of WS-353[13] and contains 999 pairs of nouns (666), verbs (222), and adjectives (111); (3) **Rare Word (RW)** [Luong *et al.*, 2013], it consists of 2034 word pairs, with more rare and morphological complex words than other word similarity testsets. These datasets all contain word pairs together with human assigned similarity scores.

The performance is evaluated using the spearman rank correlation between the similarity scores computed on learned word vectors and the human judgements[14].

Result Table 1 summarizes the results on word analogy and word similarity tasks.

The third block shows the results of word analogy task. It is easy to see that word analogy is more challenging for sparse models. All sparse models have not been able to outperform CBOW and SG. It seems like that dense representations are more effective on revealing linguistic regularities between word pairs. Nevertheless, we found Sparse CBOW can achieve the similar performance comparing with CBOW if we reduce its sparsity level less than 85%.

The fourth block shows the results of word similarity task on three different testsets. It is easy to observe that WS-353 is quite easy for all the models while SL-999 is more challenging as Hill *et al.* [2015] claimed. For SL-999, Sparse CBOW performs significantly better than all the other models including both sparse models and state-of-the-art dense models. This indicates that the salient semantic meanings learned from Sparse CBOW can be very helpful for modeling word similarities.

The last block of the table reports the average performance on all tasks. As we can see, NNSE and SC are much worse than the dense models (GloVe, SG, and CBOW) and their corresponding baseline, PPMI(W-C) and CBOW respectively. Moreover, SC still performs a little bit worse than CBOW even with 10 times dimensionality vectors. Even so, the proposed Sparse CBOW still achieves the best average performance. It is worth stressing that our sparse model performs as well as or even better than the state-of-the-art dense models with only about 10% non-zero entries of dense models.

Effects of Vector Length

The dimensionality is an important configuration in word representations. In Figure 1, we report the average performances of CBOW and Sparse CBOW across all tasks against the varying dimensionality. As can be seen in Figure 1, the peak performance of both CBOW and sparse CBOW are very close. CBOW achieves its best performance under the dimension 300, and then drops with the increasing dimensionality. On the contrary, Sparse CBOW is more stable than CBOW. Moreover, Sparse CBOW performs slightly better than CBOW on all dimensions except the smaller dimensions (50, 100). It suggests that sparsity can make the word representation learning process more stable.

[13]SL-999 focuses on measuring how well models capture similarity, rather than relatedness or association that WS-353 do. As a result, it is more challenging for word representation models.

[14]In all experiments, we removed the word pairs that cannot be found in the vocabulary

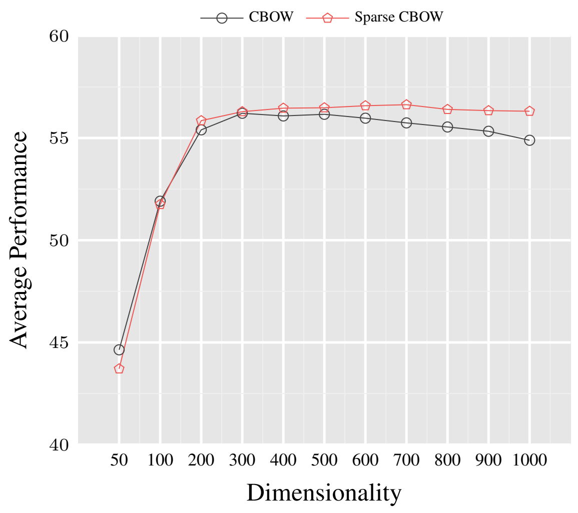

Figure 1: Average performance of CBOW and Sparse CBOW across all tasks against the varying dimensionality.

4.3 Interpretability

To evaluate the interpretability of our learned sparse word representations, we conduct experiments on word intrusion task and some case studies focusing on individual dimensions.

Word Intrusion

Following [Murphy *et al.*, 2012; Faruqui *et al.*, 2015], we also evaluate the interpretability of learned word representations through the word intrusion task. The task seeks to measure how coherent each dimension of these vectors are.

The data construction of word intrusion task proceeds as follows: for each dimension i of the learned word vector, it first sorts the words on that dimension alone in descending order. Next, it creates a set consisting of the top 5 words from the sorted list, and also one word from the bottom half of this list, which is also present in the top 10% of some other dimension i'. The last word added from the bottom half is called an *intruder*. An example of such a set constructed from a dimension of the Sparse CBOW is shown below:

{poisson, parametric, markov, bayesian, stochastic, *jodel*}

where *jodel* is the intruder word which means an aircraft company, while the rest of the words represent different concepts in statistical learning.

The goal of the traditional word intrusion task is to evaluate whether human judges can identify the intruder word. How-

Table 2: Results for word intrusion task. Higher values are better. Bold scores are the best.

Model	Sparsity	DistRatio
GloVe	0%	1.07
CBOW	0%	1.09
SG	0%	1.12
NNSE (PPMI)	89.15%	**1.55**
SC (CBOW)	88.34%	1.24
Sparse CBOW	90.06%	1.39

ever, such manual evaluation method is an arduous, costly, and subjective process. In this paper, we propose a new evaluation metric for the word intrusion task without human assessment. The intuition of word intrusion task is that if the learned representation is coherent and interpretable, then it should be easy to pick out the intruder word. To this end, the intruder word should be dissimilar to the top 5 words while those top words should be similar to each other. Therefore, we use the ratio of the distance between the intruder word and top words to the distance between the top words to quantify the interpretability of learned word representations. The higher ratio corresponds to better interpretability since it indicates the intrusion word is far away from the top words and can be easy picked out. Formally, the evaluation metric can be formalized as:

$$\mathrm{DistRatio} = \frac{1}{d}\sum_{i=1}^{d}\frac{\mathrm{InterDist}_i}{\mathrm{IntraDist}_i}$$

$$\mathrm{IntraDist}_i = \sum_{w_j \in \mathrm{top}_k(i)} \sum_{\substack{w_k \in \mathrm{top}_k(i)\\ w_k \neq w_j}} \frac{\mathrm{dist}(w_j, w_k)}{k(k-1)}$$

$$\mathrm{InterDist}_i = \sum_{w_j \in \mathrm{top}_k(i)} \frac{\mathrm{dist}(w_j, w_{b_i})}{k}$$

where $\mathrm{top}_k(i)$ denotes top k words on dimension i, w_{b_i} denotes the intrusion word for dimension i, $\mathrm{dist}(w_j, w_k)$ denotes the distance between word w_j and w_k, $\mathrm{IntraDist}_i$ denotes the average distance between top 5 words on dimension i, and $\mathrm{InterDist}_i$ denotes the average distance between the intruder word and top words on dimension i. In this paper, k is set as 5 and $\mathrm{dist}(w_j, w_k)$ is defined as euclidean distance.

Result We run the experiment ten times since there exists randomness in selection of intruder words. The average results for 300 dimensional vectors of each model are reported in Table 2. we can observe that all sparse models perform significantly better than dense models. This confirms that the sparse representations are more interpretable than the dense vectors. Besides, as the two methods based on CBOW, the results show that our model can gain more improvement than SC. The reason might be that our method directly learns the sparse word representations with respect to the original predictive task of CBOW, and thus may avoid the information loss caused by a separate sparse coding step in SC.

Moreover, NNSE obtains the highest score for this task. This suggests that, besides sparse constraint, non-negative constraint might also be a good choice for improving the interpretability of word representations. Luo *et al.* [2015] also verified that non-negativity is a beneficial factor for interpretability in skip gram model.

Table 3: Top 5 words of some dimensions in CBOW and Sparse CBOW.

Model	Top 5 Words
CBOW	beat, finish, wedding, prize, read
	rainfall, footballer, breakfast, weekdays, angeles
	landfall, interview, asked, apology, dinner
	becomes, died, feels, resigned, strained
	best, safest, iucn, capita, tallest
Sparse CBOW	poisson, parametric, markov, bayesian, stochastic
	ntfs, gzip, myfile, filenames, subdirectories
	hugely, enormously, immensely, wildly, tremendously
	earthquake, quake, uprooted, levees, spectacularly
	bosons, accretion, higgs, neutrinos, quarks

Case Study

Besides the quantitative evaluation, we also conduct some case studies to verify if a vector dimension is interpretable. For this purpose, we select top five words from word vectors' dimensions and check whether these words reveal some semantic or syntactic groupings.

Table 3 shows top 5 words from some dimensions in learned CBOW and Sparse CBOW, one dimension per row. For Sparse CBOW, It is clear to see that the first row lists the concepts in statistical learning, second row talks about the computer file system, the third row contains all adverbs describing "to a great degree", the fourth row lists different things about disasters like earthquake or flood, and the last row talks about particles in physics. All these show the dimensions of Sparse CBOW reveal some clear and consistent semantic meanings. In contrast, the dimensions of CBOW do not convey consistent meanings. These results also confirm that our proposed model has a better interpretability.

5 Conclusion

In this paper, we present a method to learn sparse word representations directly from raw text data. The proposed Sparse CBOW model applies the ℓ_1 regularization on CBOW model and uses regularized dual averaging algorithm to optimize it in online training. The experimental results on both word similarity tasks and word analogy tasks show that, compared with the original CBOW model, Sparse CBOW can obtain competitive results using less than 10% non-zero elements. Besides, we also test our model on word intrusion task and design a new evaluation metric for it to get rid of human evaluation. The results demonstrate the effectiveness of our proposed model in interpretability.

Acknowledgments

This work was funded by 973 Program of China under Grants No. 2014CB340401 and 2012CB316303, 863 Program of China under Grants No. 2014AA015204, the National Natural Science Foundation of China (NSFC) under Grants No. 61232010, 61472401, 61433014, 61425016, 61203298, and 61572473, Key Research Program of the Chinese Academy of Sciences under Grant No. KGZD-EW-T03-2, and Youth Innovation Promotion Association CAS under Grants No. 20144310 and 2016102.

References

[Attwell and Laughlin, 2001] D Attwell and S B Laughlin. An energy budget for signaling in the grey matter of the brain. *J Cereb Blood Flow Metab*, 21(10):1133–1145, Oct 2001.

[Bengio *et al.*, 2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, 2003.

[Collobert and Weston, 2008] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167, 2008.

[Collobert *et al.*, 2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November 2011.

[Deerwester *et al.*, 1990] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *J. Am. Soc. Inf. Sci. Technol.*, 41(6):391–407, 1990.

[Dhillon *et al.*, 2011] Paramveer Dhillon, Dean P Foster, and Lyle H. Ungar. Multi-view learning of word embeddings via cca. In *Proceedings of NIPS*, pages 199–207. 2011.

[Duchi and Singer, 2009] John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *J. Mach. Learn. Res.*, 10:2899–2934, December 2009.

[Faruqui *et al.*, 2015] Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. Sparse overcomplete word vector representations. In *Proceedings of ACL*, pages 1491–1500, July 2015.

[Finkelstein *et al.*, 2002] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan andGadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. *ACM Trans. Inf. Syst.*, 20(1):116–131, January 2002.

[Firth, 1957] J. R. Firth. A synopsis of linguistic theory 1930-55. *Studies in Linguistic Analysis (special volume of the Philological Society)*, 1952-59:1–32, 1957.

[Griffiths *et al.*, 2007] Thomas L Griffiths, Mark Steyvers, and Joshua B Tenenbaum. Topics in semantic representation. *Psychol Rev*, 114(2):211–244, Apr 2007.

[Harris, 1954] Zellig Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

[Hill *et al.*, 2015] Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, pages 665–695, September 2015.

[Hinton *et al.*, 1986] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. Distributed representations. In *Parallel Distributed Processing, Vol. 1*, pages 77–109. MIT Press, 1986.

[Langford *et al.*, 2009] John Langford, Lihong Li, and Tong Zhang. Sparse online learning via truncated gradient. *J. Mach. Learn. Res.*, 10:777–801, June 2009.

[Levy and Goldberg, 2014a] Omer Levy and Yoav Goldberg. Linguistic regularities in sparse and explicit word representations. In *Proceedings of CoNLL*, pages 171–180, June 2014.

[Levy and Goldberg, 2014b] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Proceedings of NIPS*, pages 2177–2185. 2014.

[Lewicki and Sejnowski, 2000] Michael S. Lewicki and Terrence J. Sejnowski. Learning overcomplete representations. *Neural Computation*, 12(2):337–365, Feb 2000.

[Luo *et al.*, 2015] Hongyin Luo, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Online learning of interpretable word embeddings. In *Proceedings of EMNLP*, pages 1687–1692, 2015.

[Luong *et al.*, 2013] Minh-Thang Luong, Richard Socher, and Christopher D. Manning. Better word representations with recursive neural networks for morphology. In *Proceedings of CoNLL*, pages 104–113, 2013.

[McMahan and Streeter, 2010] H. Brendan McMahan and Matthew J. Streeter. Adaptive bound optimization for online convex optimization. In *COLT*, pages 244–256, 2010.

[Mikolov *et al.*, 2013a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of ICLR*, 2013.

[Mikolov *et al.*, 2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119. 2013.

[Mikolov *et al.*, 2013c] Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL*, May 2013.

[Mnih and Kavukcuoglu, 2013] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Proceedings of NIPS*, pages 2265–2273. 2013.

[Murphy *et al.*, 2012] Brian Murphy, Partha Talukdar, and Tom Mitchell. Learning effective and interpretable semantic models using non-negative sparse embedding. In *Proceedings of COLING*, pages 1933–1950, 2012.

[Olshausen and Field, 1997] Bruno A. Olshausen and David J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311–3325, 1997.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543, 2014.

[Schnabel *et al.*, 2015] Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. Evaluation methods for unsupervised word embeddings. In *Proceedings of EMNLP*, pages 298–307, 2015.

[Shaoul and Westbury, 2010] Cyrus Shaoul and Chris Westbury. The westbury lab wikipedia corpus. *Edmonton, AB: University of Alberta*, 2010.

[Socher *et al.*, 2013] Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. Parsing with compositional vector grammars. In *Proceedings of ACL*, pages 455–465, 2013.

[Stratos *et al.*, 2015] Karl Stratos, Michael Collins, and Daniel Hsu. Model-based word embeddings from decompositions of count matrices. In *Proceedings of ACL*, pages 1282–1291, July 2015.

[Vinson and Vigliocco, 2008] DavidP. Vinson and Gabriella Vigliocco. Semantic feature production norms for a large set of objects and events. *Behavior Research Methods*, 40(1):183–190, 2008.

[Xiao, 2009] Lin Xiao. Dual averaging method for regularized stochastic learning and online optimization. In *Proceedings of NIPS*, pages 2116–2124. 2009.