Neural-based Noise Filtering from Word Embeddings

Kim Anh Nguyen and **Sabine Schulte im Walde** and **Ngoc Thang Vu**
Institut für Maschinelle Sprachverarbeitung
Universität Stuttgart
Pfaffenwaldring 5B, 70569 Stuttgart, Germany
{nguyenkh,schulte,thangvu}@ims.uni-stuttgart.de

Abstract

Word embeddings have been demonstrated to benefit NLP tasks impressively. Yet, there is room for improvement in the vector representations, because current word embeddings typically contain unnecessary information, i.e., *noise*. We propose two novel models to improve word embeddings by unsupervised learning, in order to yield word denoising embeddings. The word denoising embeddings are obtained by strengthening salient information and weakening noise in the original word embeddings, based on a deep feed-forward neural network filter. Results from benchmark tasks show that the filtered word denoising embeddings outperform the original word embeddings.

1 Introduction

Word embeddings aim to represent words as low-dimensional dense vectors. In comparison to distributional count vectors, word embeddings address the problematic sparsity of word vectors and achieved impressive results in many NLP tasks such as sentiment analysis (e.g., Kim (2014)), word similarity (e.g., Pennington et al. (2014)), and parsing (e.g., Lazaridou et al. (2013)). Moreover, word embeddings are attractive because they can be learned in an unsupervised fashion from unlabeled raw corpora. There are two main approaches to create word embeddings. The first approach makes use of neural-based techniques to learn word embeddings, such as the Skip-gram model (Mikolov et al., 2013). The second approach is based on matrix factorization (Pennington et al., 2014), building word embeddings by factorizing word-context co-occurrence matrices.

In recent years, a number of approaches have focused on improving word embeddings, often by integrating lexical resources. For example, Adel and Schütze (2014) applied coreference chains to Skip-gram models in order to create word embeddings for antonym identification. Pham et al. (2015) proposed an extension of a Skip-gram model by integrating synonyms and antonyms from WordNet. Their extended Skip-gram model outperformed a standard Skip-gram model on both general semantic tasks and distinguishing antonyms from synonyms. In a similar spirit, Nguyen et al. (2016) integrated distributional lexical contrast into every single context of a target word in a Skip-gram model for training word embeddings. The resulting word embeddings were used in similarity tasks, and to distinguish between antonyms and synonyms. Faruqui et al. (2015) improved word embeddings without relying on lexical resources, by applying ideas from sparse coding to transform dense word embeddings into sparse word embeddings. The dense vectors in their models can be transformed into sparse overcomplete vectors or sparse binary overcomplete vectors. They showed that the resulting vector representations were more similar to interpretable features in NLP and outperformed the original vector representations on several benchmark tasks.

In this paper, we aim to improve word embeddings by reducing their noise. The hypothesis behind our approaches is that word embeddings contain unnecessary information, i.e. *noise*. We start out with the idea of learning word embeddings as suggested by Mikolov et al. (2013), relying on the distributional hypothesis (Harris, 1954) that words with similar distributions have related meanings. We address those

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2699–2707, Osaka, Japan, December 11-17 2016.

distributions in embedded vectors of words that decrease the value of such vector representations. For instance, consider the sentence *the quick brown fox gazing at the cloud jumped over the lazy dog*. The context *jumped* can be used to predict the words *fox*, *cloud* and *dog* in a window size of 5 words; however, a *cloud* cannot *jump*. The context `jumped` is therefore considered as noise in the embedded vector of `cloud`. We propose two novel models to smooth word embeddings by filtering noise: We strengthen salient contexts and weaken unnecessary contexts.

The first proposed model is referred to as *complete word denoising embeddings model (CompEmb)*. Given a set of original word embeddings, we use a filter to learn a denoising matrix, and then project the set of original word embeddings into this denoising matrix to produce a set of complete word denoising embeddings. The second proposed model is referred to as *overcomplete word denoising embeddings model (OverCompEmb)*. We make use of a sparse coding method to transform an input set of original word embeddings into a set of overcomplete word embeddings, which is considered as the "overcomplete process". We then apply a filter to train a denoising matrix, and thereafter project the set of original word embeddings into the denoising matrix to generate a set of overcomplete word denoising embeddings. The key idea in our models is to use a filter for learning the denoising matrix. The architecture of the filter is a feed-forward, non-linear and parameterized neural network with a fixed depth that can be used to learn the denoising matrices and reduce noise in word embeddings. Using state-of-the-art word embeddings as input vectors, we show that the resulting word denoising embeddings outperform the original word embeddings on several benchmark tasks such as word similarity and word relatedness tasks, synonymy detection and noun phrase classification. Furthermore, the implementation of our models is made publicly available[1].

The remainder of this paper is organized as follows: Section 2 presents the two proposed models, the loss function, and the sparse coding technique for overcomplete vectors. In Section 3, we demonstrate the experiments on evaluating the effects of our word denoising embeddings, tuning hyperparameters, and we analyze the effects of filter depth. Finally, Section 4 concludes the paper.

2 Learning Word Denoising Embeddings

In this section, we present the two contributions of this paper. Figure 1 illustrates our two models to learn denoising for word embeddings. The first model on the top, the complete word denoising embeddings model "CompEmb" (Section 2.1), filters noise from word embeddings $\mathbf{X}$ to produce complete word denoising embeddings $\mathbf{X}^*$, in which the vector length of $\mathbf{X}^*$ in comparison to $\mathbf{X}$ is unchanged after denoising (called *complete*). The second model at the bottom of the figure, the overcomplete word denoising embeddings model "OverCompEmb" (Section 2.2), filters noise from word embeddings $\mathbf{X}$ to yield overcomplete word denoising embeddings $\mathbf{Z}^*$, in which the vector length of $\mathbf{Z}^*$ tends to be greater than the vector length of $\mathbf{X}$ (called *overcomplete*).

For the notations, let $\mathbf{X} \in \mathbb{R}^{V \times L}$ is an input set of word embeddings in which V is the vocabulary size, and L is the vector length of $\mathbf{X}$. Furthermore, $\mathbf{Z} \in \mathbb{R}^{V \times K}$ is the overcomplete word embeddings in which K is the vector length of $\mathbf{Z}$ ($K > L$); finally, $\mathbf{D} \in \mathbb{R}^{L \times L}$ is the pre-trained dictionary (Section 2.4).

2.1 Complete Word Denoising Embeddings

In this subsection, we aim to reduce noise in the given input word embeddings $\mathbf{X}$ by learning a denoising matrix $\mathbf{Q_c}$. The complete word denoising embeddings $\mathbf{X}^*$ are then generated by projecting $\mathbf{X}$ into $\mathbf{Q_c}$. More specifically, given an input $\mathbf{X} \in \mathbb{R}^{V \times L}$, we seek to optimize the following objective function:

$$\underset{\mathbf{X},\mathbf{Q_c},\mathbf{S}}{\operatorname{argmin}} \sum_{i=1}^{V} \|\mathbf{x_i} - f(\mathbf{x_i}, \mathbf{Q_c}, \mathbf{S})\| + \alpha \|\mathbf{S}\|_1 \tag{1}$$

where f is a filter; $\mathbf{S}$ is a lateral inhibition matrix; and α is a regularization hyperparameter. Inspired by studies on sparse modeling, the matrix $\mathbf{S}$ is chosen to be symmetric and has zero on the diagonal.

[1] `https://github.com/nguyenkh/NeuralDenoising`

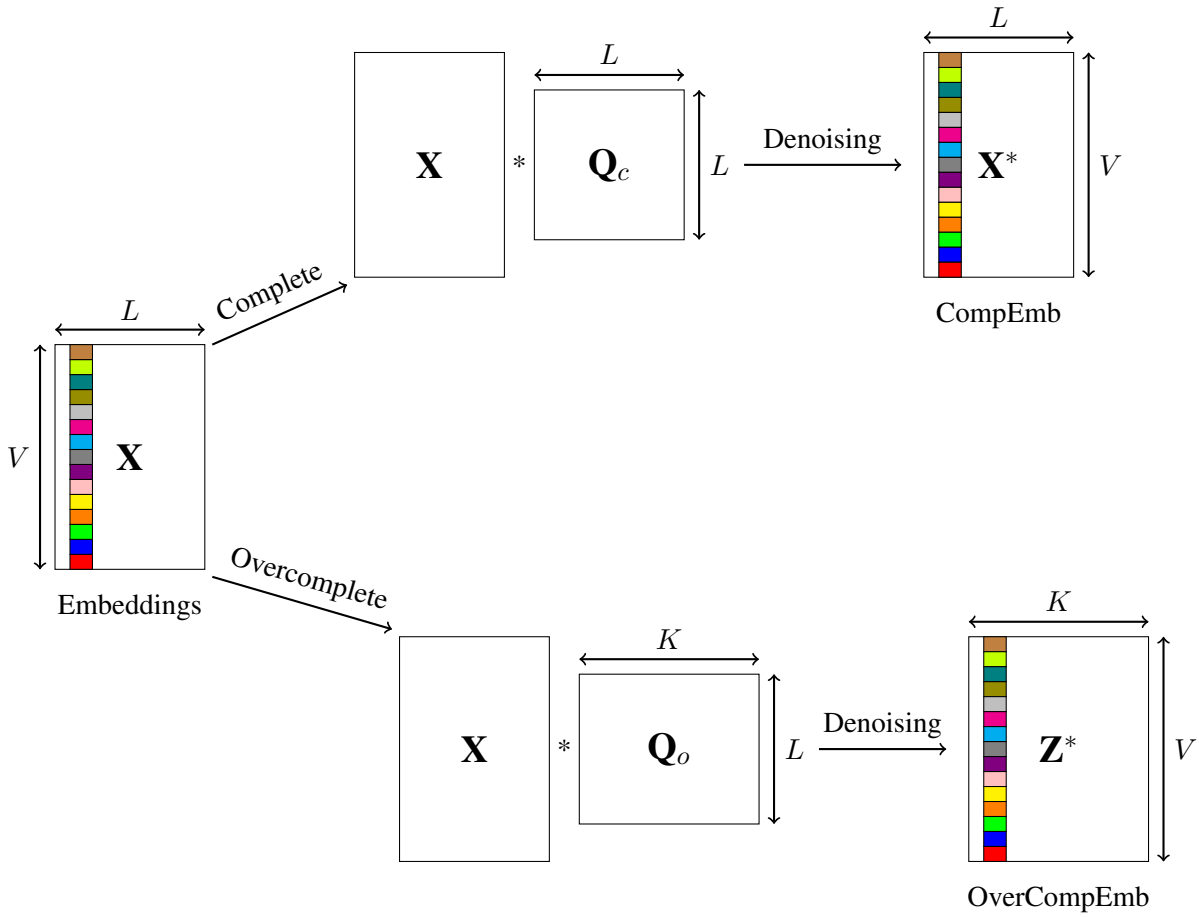


Figure 1: Illustration of word denoising embeddings methods, with complete word denoising embeddings at the top, and overcomplete word denoising embeddings at the bottom.

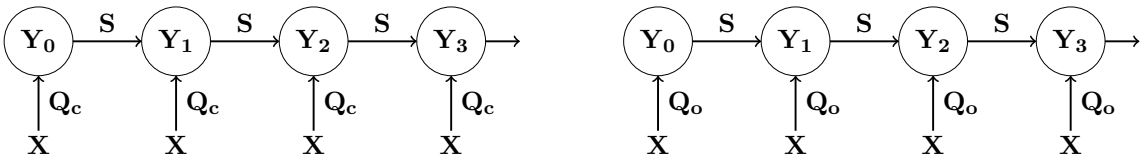
The goal of this matrix is to implement excitatory interaction between neurons, and to increase the convergence speed of the neural network (Szlam et al., 2011). More concretely, the matrices $\mathbf{Q_c}$ and $\mathbf{S}$ are initialized with $\mathbf{I}$ and E, which are identity matrices, and the Lipschitz constant:

$$\begin{aligned}&\mathbf{Q_c} = \tfrac{1}{E}\mathbf{D}; \mathbf{S} = \mathbf{I} - \tfrac{1}{E}\mathbf{D^T D}\\ &E > \text{ the largest eigenvalue of } \mathbf{D^T D}\\ &\mathbf{D} \in \mathbb{R}^{L\times L} \text{ be pre-trained dictionary}\end{aligned}$$

The underlying idea for reducing noise is to make use of a filter f to learn a denoising matrix $\mathbf{Q_c}$; hence, we design the filter f as a non-linear, parameterized, feed-forward architecture with a fixed depth that can be trained to approximate $f(\mathbf{X}, \mathbf{Q_c}, \mathbf{S})$ to $\mathbf{X}$ as in Figure 2a. As a result, noise from word embeddings will be filtered by layers of the filter f. The filter f is encoded as a recursive function by iterating over the number of fixed depth T, as the following recursive Equation 2 shows:

$$\begin{aligned}&\mathbf{Y} = f(\mathbf{X}, \mathbf{Q_c}, \mathbf{S})\\ &\mathbf{Y}(0) = \mathcal{G}(\mathbf{X}\mathbf{Q_c})\\ &\mathbf{Y}(k+1) = \mathcal{G}(\mathbf{X}\mathbf{Q_c} + \mathbf{Y}(k)\mathbf{S})\\ &0 \le k < T\end{aligned} \tag{2}$$

$\mathcal{G}$ is a non-linear activation function. The matrices $\mathbf{Q_c}$ and $\mathbf{S}$ are learned to produce the lowest possible error in a given number of iterations. Matrix $\mathbf{S}$, in the architecture of filter f, acts as a controllable matrix to filter unnecessary information on embedded vectors, and to impose restrictions on further reducing the



(a) CompEmb (b) OverCompEmb

Figure 2: Architecture of the filters with the fixed depth $T = 3$.

computational burden (e.g., solving low-rank approximation problem or keeping the number of terms at zero (Gregor and LeCun, 2010)). Moreover, the initialization of the matrices $\mathbf{Q_c}$, $\mathbf{S}$ and E enhances a highly efficient minimization of the objective function in Equation 1, due to the pre-trained dictionary $\mathbf{D}$ that carries the information of reconstructing $\mathbf{X}$.

The architecture of the filter f is a recursive feed-forward neural network with the fixed depth T, so the number of T plays a significant role in controlling the approximation of $\mathbf{X}^*$. The effects of T will be discussed later in Section 3.4. When $\mathbf{Q_c}$ is trained, the complete word denoising embeddings $\mathbf{X}^*$ are yielded by projecting $\mathbf{X}$ into $\mathbf{Q_c}$, as shown by the following Equation 3:

$$\mathbf{X}^* = \mathcal{G}(\mathbf{X}\mathbf{Q_c}) \tag{3}$$

2.2 Overcomplete Word Denoising Embeddings

Now we introduce our method to reduce noise and overcomplete vectors in the given input word embeddings. To obtain overcomplete word embeddings, we first use a sparse coding method to transform the given input word embeddings $\mathbf{X}$ into overcomplete word embeddings $\mathbf{Z}$. Secondly, we use overcomplete word embeddings $\mathbf{Z}$ as the intermediate word embeddings to optimize the objective function: A set of input word embeddings $\mathbf{X} \in \mathbb{R}^{V \times L}$ is transformed to overcomplete word embeddings $\mathbf{Z} \in \mathbb{R}^{V \times K}$ by applying sparse coding method in Section 2.4. We then make use of the pre-trained dictionary $\mathbf{D} \in \mathbb{R}^{L \times K}$ and $\mathbf{Z} \in \mathbb{R}^{V \times K}$ to learn the denoising matrix $\mathbf{Q_o}$ by minimizing the following Equation 4:

$$\underset{\mathbf{X},\mathbf{Q_o},\mathbf{S}}{\operatorname{argmin}} \sum_{i=1}^{V} \|\mathbf{z_i} - f(\mathbf{x_i}, \mathbf{Q_o}, \mathbf{S})\| + \alpha\|\mathbf{S}\|_1 \tag{4}$$

The initialization of the parameters $\mathbf{Q_o}$, $\mathbf{S}$, E and α follows the same procedure as described in Section 2.1, and with the same interpretation of the filter architecture in Figure 2b. The overcomplete word denoising embeddings $\mathbf{Z}^*$ are then generated by projecting $\mathbf{X}$ into the denoising matrix $\mathbf{Q_o}$ and using the non-linear activation function $\mathcal{G}$ in the following Equation 5:

$$\mathbf{Z}^* = \mathcal{G}(\mathbf{X}\mathbf{Q_o}) \tag{5}$$

2.3 Loss Function

For each pair of term vectors $\mathbf{x_i} \in \mathbf{X}$ and $\mathbf{y_i} \in \mathbf{Y} = f(\mathbf{X}, \mathbf{Q_c}, \mathbf{S})$, we make use of the cosine similarity to measure the similarity between $\mathbf{x_i}$ and $\mathbf{y_i}$ as follows:

$$sim(\mathbf{x_i}, \mathbf{y_i}) = \frac{\mathbf{x_i} \cdot \mathbf{y_i}}{\|\mathbf{x_i}\|\|\mathbf{y_i}\|} \tag{6}$$

Let Δ be the difference between $sim(\mathbf{x_i}, \mathbf{x_i})$ and $sim(\mathbf{x_i}, \mathbf{y_i})$, equivalently $\Delta = 1 - sim(\mathbf{x_i}, \mathbf{y_i})$. We then optimize the objective function in Equation 1 by minimizing Δ; and the same loss function is also applied to optimize the objective function in Equation 4. Training is done through Stochastic Gradient Descent with the Adadelta update rule (Zeiler, 2012).

2.4 Sparse Coding

Sparse coding is a method to represent vector representations as a sparse linear combination of elementary atoms of a given dictionary. The underlying assumption of sparse coding is that the input vectors can be reconstructed accurately as a linear combination of some basis vectors and a few number of non-zero coefficients (Olshausen and Field, 1996).

The goal is to approximate a dense vector in $\mathbb{R}^L$ by a sparse linear combination of a few columns of a matrix $\mathbf{D} \in \mathbb{R}^{L \times K}$ in which K is a new vector length and the matrix $\mathbf{D}$ be called a *dictionary*. Concretely, given V input vectors of L dimensions $\mathbf{X} = [x_1, x_2, ..., x_V]$, the dictionary and sparse vectors can be formulated as the following minimization problem:

$$\min_{\mathbf{D} \in \mathcal{C}, \mathbf{Z} \in \mathbb{R}^{K \times V}} \sum_{i=1}^{V} \|\mathbf{x_i} - \mathbf{D}\mathbf{z}_i\|_2^2 + \lambda \|\mathbf{z}_i\|_1 \tag{7}$$

$\mathbf{Z} = [z_1, ..., z_V]$ carries the decomposition coefficients of $\mathbf{X} = [x_1, x_2, ..., x_V]$; and λ represents a scalar to control the sparsity level of $\mathbf{Z}$. The dictionary $\mathbf{D}$ is typically learned by minimizing Equation 7 over input vectors $\mathbf{X}$. In the case of overcomplete representations $\mathbf{Z}$, the vector length K is typically implied as $K = \gamma L$ ($\gamma > 0$).

In the method of overcomplete word denoising embeddings (Section 2.2), our approach makes use of overcomplete word embeddings $\mathbf{Z}$ as the intermediate word embeddings reconstructed by applying a sparse coding method to word embeddings $\mathbf{X}$. The overcomplete word embeddings $\mathbf{Z}$ are then utilized to optimize Equation 4. To obtain overcomplete word embeddings $\mathbf{Z}$ and dictionaries, we use the `SPAMS` package[2] to implement sparse coding for word embeddings $\mathbf{X}$ and to train the dictionaries $\mathbf{D}$.

3 Experiments

3.1 Experimental Settings

As input word embeddings, we rely on two state-of-the-art word embeddings methods: word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). We use the `word2vec` tool[3] and the web corpus *ENCOW14A* (Schäfer and Bildhauer, 2012; Schäfer, 2015) which contains approximately 14.5 billion tokens, in order to train Skip-gram models with 100 and 300 dimensions. For the GloVe method, we use pre-trained vectors of 100 and 300 dimensions[4] that were trained on 6 billion words from Wikipedia and English Gigaword. The $\tanh$ function is used as the non-linear activation function in both approaches. The fixed depth of filter T is set to 3; further hyperparameters are chosen as discussed in Section 3.2. To train the networks, we use the `Theano` framework (Theano Development Team, 2016) to implement our models with a mini-batch size of 100. Regularization is applied by dropouts of 0.5 and 0.2 for input and output layers (without tuning), respectively.

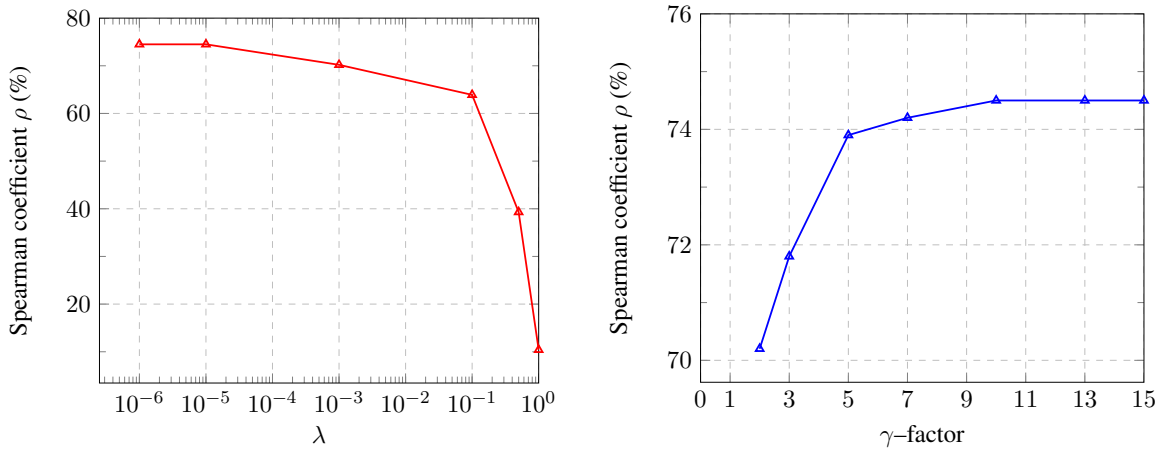
3.2 Hyperparameter Tuning

In both methods of denoising word embeddings, the ℓ_1 regularization penalty α is set to 0.5 without tuning in Equation 1 and 4. The method of learning overcomplete word denoising embeddings relies on the mediate word embeddings $\mathbf{Z}$ to minimize the objective function in Equation 4. The sparsity of $\mathbf{Z}$ depends on the ℓ_1 regularization λ in Equation 7; and the length vector K of $\mathbf{Z}$ is implied as $K = \gamma L$. Therefore, we aim to tune λ and γ such that $\mathbf{Z}$ represents the nearest approximation of the original vector representation $\mathbf{X}$. We perform a grid search on $\lambda \in \{1.0, 0.5, 0.1, 10^{-3}, 10^{-6}\}$ and $\gamma \in \{2, 3, 5, 7, 10, 13, 15\}$, developing on the word similarity task WordSim353 (to be discussed on Section 3.3). The hyperparameter tunings are illustrated in Figures 3a and 3b for sparsity and overcomplete vector length tuning, respectively. In both approaches, we set λ to 10^{-6} and γ to 10 for the sparsity and length of overcomplete word embeddings.

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

[3] https://code.google.com/p/word2vec/

[4] http://www-nlp.stanford.edu/projects/glove/



(a) Sparsity of sparse coding.

(b) Length of overcomplete vectors.

Figure 3: Illustration of hyperparameter tuning.

3.3 Effects of Word Denoising Embeddings

In this section, we quantify the effects of word denoising embeddings on three kinds of tasks: similarity and relatedness tasks, detecting synonymy, and bracketed noun phrase classification task. In comparison to the performance of word denoising embeddings, we take into account state-of-the-art word embeddings (Skip-gram and GloVe word embeddings) as baselines. Besides, we also use the public source code[5] to re-implement the two methods suggested by Faruqui et al. (2015) which are vectors $\mathbf{A}$ (sparse overcomplete vectors) and $\mathbf{B}$ (sparse binary overcomplete vectors).

The effects of the word denoising embeddings on the tasks are shown in Table 1. The results show that the vectors $\mathbf{X}^*$ and $\mathbf{Z}^*$ outperform the original vectors $\mathbf{X}$, $\mathbf{A}$ and $\mathbf{B}$, except for the NP task, in which the vectors $\mathbf{B}$ based on the 300-dimensional GloVe vectors are best. The effect of the vectors $\mathbf{Z}^*$ is slightly less impressive, when compared to the overcomplete vectors $\mathbf{X}^*$. The overcomplete word embeddings $\mathbf{Z}$ strongly differ from the word embeddings $\mathbf{X}$; hence, the denoising is affected. However, the performance of the vectors $\mathbf{Z}^*$ still outperforms the original vectors $\mathbf{X}$, $\mathbf{A}$ and $\mathbf{B}$ after the denoising process.

3.3.1 Relatedness and Similarity Tasks

For the relatedness task, we use two kinds of datasets: MEN (Bruni et al., 2014) consists of 3000 word pairs comprising 656 nouns, 57 adjectives and 38 verbs. The WordSim-353 relatedness dataset (Finkelstein et al., 2001) contains 252 word pairs. Concerning the similarity tasks, we evaluate the denoising vectors again on two kinds of datasets: *SimLex-999* (Hill et al., 2015) contains 999 word pairs including 666 noun, 222 verb and 111 adjective pairs. The WordSim-353 similarity dataset consists of 203 word pairs. In addition, we evaluate our denoising vectors on the WordSim-353 dataset which contains 353 pairs for both similarity and relatedness relations. We calculate cosine similarity between the vectors of two words forming a test pair, and report the Spearman rank-order correlation coefficient ρ (Siegel and Castellan, 1988) against the respective gold standards of human ratings.

3.3.2 Synonymy

We evaluate on 80 TOEFL (Test of English as a Foreign Language) synonym questions (Landauer and Dumais, 1997) and 50 ESL (English as a Second Language) questions (Turney, 2001). The first dataset represents a subset of 80 multiple-choice synonym questions from the TOEFL test: a word is paired with four options, one of which is a valid synonym. The second dataset contains 50 multiple-choice synonym questions, and the goal is to choose a valid synonym from four options. For each question, we compute the cosine similarity between the target word and the four candidates. The suggested answer is the candidate with the highest cosine score. We use accuracy to evaluate the performance.

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

Vectors		Simlex-999 Corr.	MEN Corr.	WS353 Corr.	WS353-SIM Corr.	WS353-REL Corr.	ESL Acc.	TOEFL Acc.	NP Acc.
SG-100	**X**	33.7	72.9	69.7	74.5	65.5	48.9	62.0	72.8
	X*	33.2	72.8	70.6	74.8	66.0	53.0	**64.5**	78.5
	Z*	**35.9**	**74.4**	**71.2**	**75.2**	**68.1**	53.0	62.0	**79.1**
	A	32.5	69.8	65.5	69.5	60.2	**55.1**	51.8	78.8
	B	31.9	70.4	65.8	72.6	62.2	53.0	58.2	74.1
SG-300	**X**	36.1	74.7	71.0	75.9	66.1	**59.1**	72.1	77.9
	X*	**37.1**	**75.8**	**71.8**	**76.4**	**66.9**	**59.1**	74.6	**79.3**
	Z*	36.5	75.0	70.6	**76.4**	64.4	57.1	**77.2**	78.6
	A	32.9	72.4	67.5	71.9	63.4	53.0	65.8	78.3
	B	32.7	71.2	63.3	68.7	56.2	51.0	70.8	78.6
GloVe-100	**X**	29.7	69.3	52.9	60.3	49.5	46.9	82.2	76.4
	X*	**31.7**	**70.9**	**58.0**	**63.8**	**57.3**	53.0	**88.6**	**77.4**
	Z*	30.0	**70.9**	56.0	62.8	53.8	**57.0**	81.0	77.3
	A	30.7	70.7	54.9	62.2	51.2	55.1	78.4	77.1
	B	31.0	69.2	57.3	62.3	53.7	46.9	73.4	76.4
GloVe-300	**X**	37.0	74.8	60.5	66.3	57.2	**61.2**	89.8	74.3
	X*	**40.2**	**76.8**	**64.9**	**69.8**	**62.0**	**61.2**	**92.4**	76.3
	Z*	39.0	75.2	63.0	67.9	59.7	57.1	86.0	75.7
	A	36.7	74.1	61.5	67.7	57.8	55.1	87.3	79.9
	B	33.1	70.2	57.0	62.2	53.0	51.0	91.4	**80.0**

Table 1: Effects of word denoising embeddings. Vectors **X** represent the baselines; vectors **A** and **B** were suggested by Faruqui et al. (2015); the vector length **Z*** is equal to 10 times of vector length **X**.

3.3.3 Phrase parsing as Classification

Lazaridou et al. (2013) introduced a dataset of noun phrases (NP) in which each NP consists of three elements: the first element is either an adjective or a noun, and the other elements are all nouns. For a given NP (such as *blood pressure medicine*), the task is to predict whether it is a left-bracketed NP, e.g., *(blood pressure) medicine*, or a right-bracketed NP, e.g., *blood (pressure medicine)*.

The dataset contains 2227 noun phrases split into 10 folds. For each NP, we use the average of word vectors as features to feed into the classifier by tuning the hyperparameters (w_1, w_2 and w_3) for each element (e_1, e_2 and e_3) within the NP: $\vec{e}_{NP} = \frac{1}{3}(w_1\vec{e}_1 + w_2\vec{e}_2 + w_3\vec{e}_3)$. We then employ the classification of the NPs by using a Support Vector Machine (SVM) with Radial Basis Function kernel. The classifier is tuned on the first fold, and cross-validation accuracy is reported on the nine remaining folds.

3.4 Effects of Filter Depth

As mentioned above, the architecture of the filter f is a feed-forward network with a fixed depth T. For each stage T, the filter f attempts to reduce the noise within input vectors by approximating these vectors based on vectors of a previous stage $T-1$. In order to investigate the effects of each stage T, we use pre-trained GloVe vectors with 100 dimensions to evaluate the denoising performance of the vectors on detecting synonymy in the TOEFL dataset across several stages of T.

The results are presented in Figure 4. The accuracy of synonymy detection increases sharply from 63.2% to 88.6% according to the number of stages T from 0 to 3. However, the denoising performance of vectors falls with the number of stages $T > 3$. This evaluation shows that the filter f with a consistently fixed depth T can be trained to efficiently filter noise for word embeddings. In other words, the number of stages T exceeds a consistent number T (with $T > 3$ in our case), leading to the loss of salient information in the vectors.

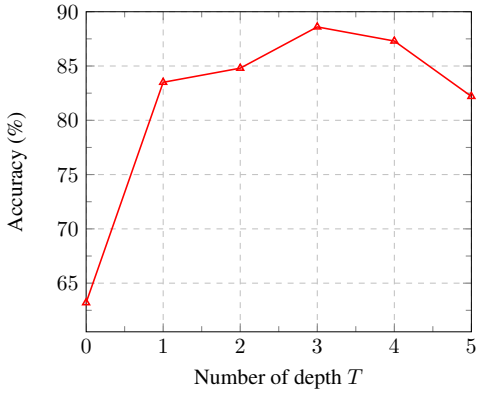


Figure 4: Effects of the filter with depth T on filtering noise.

4 Conclusion

To the best of our knowledge, we are the first to work on filtering noise in word embeddings. In this paper, we have presented two novel models to improve word embeddings by reducing noise in state-of-the-art word embeddings models. The underlying idea in our models was to make use of a deep feed-forward neural network filter to reduce noise. The first model generated complete word denoising embeddings; the second model yielded overcomplete word denoising embeddings. We demonstrated that the word denoising embeddings outperform the originally state-of-the-art word embeddings on several benchmark tasks.

Acknowledgements

The research was supported by the Ministry of Education and Training of the Socialist Republic of Vietnam (Scholarship 977/QD-BGDDT; Kim-Anh Nguyen), the DFG Collaborative Research Centre SFB 732 (Kim-Anh Nguyen, Ngoc Thang Vu), and the DFG Heisenberg Fellowship SCHU-2580/1 (Sabine Schulte im Walde).

References

Heike Adel and Hinrich Schütze. 2014. Using mined coreference chains as a resource for a semantic task. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1447–1452, Doha, Qatar.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artifical Intelligence Research (JAIR)*, 49:1–47.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. 2015. Sparse overcomplete word vector representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1491–1500, Beijing, China.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on the World Wide Web*, pages 406–414.

Karol Gregor and Yann LeCun. 2010. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning (ICML), Haifa, Israel*, pages 399–406.

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

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar.

Thomas K. Landauer and Susan T. Dumais. 1997. A solution to Platos problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.

Angeliki Lazaridou, Eva Maria Vecchi, and Marco Baroni. 2013. Fish transporters and miracle homes: How compositional distributional semantics can help NP parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1908–1913, Doha, Qatar.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. 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 (NAACL)*, pages 746–751, Atlanta, Georgia.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2016. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 454–459, Berlin, Germany.

Bruno A. Olshausen and David J. Field. 1996. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar.

Nghia The Pham, Angeliki Lazaridou, and Marco Baroni. 2015. A multitask objective to inject lexical contrast into distributional semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 21–26, Beijing, China.

Roland Schäfer and Felix Bildhauer. 2012. Building large corpora from the web using a new efficient tool chain. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 486–493, Istanbul, Turkey.

Roland Schäfer. 2015. Processing and querying large web corpora with the COW14 architecture. In *Proceedings of the 3rd Workshop on Challenges in the Management of Large Corpora*, pages 28–34, Mannheim, Germany.

Sidney Siegel and N. John Castellan. 1988. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, Boston, MA.

Arthur D. Szlam, Karol Gregor, and Yann L. Cun. 2011. Structured sparse coding via lateral inhibition. *Advances in Neural Information Processing Systems (NIPS)*, 24:1116–1124.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Peter D. Turney. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the 12th European Conference on Machine Learning (ECML)*, pages 491–502.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.