Improving Word Embedding Using Variational Dropout

Zainab Albujasim[1,4], **Diana Inkpen**[3], **Xuejun Han**[1], **Yuhong Guo**[1,2]
[1] Carleton University 1125 Colonel By Dr, Ottawa, ON, Canada
[2] CIFAR AI Chair, Amii, Canada
[3] University of Ottawa, 800 King Edward Ave., Ottawa, ON, Canada
[4] Valparaiso University, 1700 Chapel Dr, Valparaiso, IN, USA
{ZainabAlbujasim, XuejunHan,}@cmail.carleton.ca, Diana.Inkpen@uottawa.ca, Yuhong.Guo@carleton.ca

Abstract

Pre-trained word embeddings are essential in natural language processing (NLP). In recent years, many post-processing algorithms have been proposed to improve the pre-trained word embeddings. We present a novel method - **O**rthogonal **A**uto**E**ncoder with **V**ariational **D**ropout (**OAEVD**) for improving word embeddings based on orthogonal autoencoders and variational dropout. Specifically, the orthogonality constraint encourages more diversity in the latent space and increases semantic similarities between similar words, and variational dropout makes it more robust to overfitting. Empirical evaluation on a range of downstream NLP tasks, including semantic similarity, text classification, and concept categorization shows that our proposed method effectively improves the quality of pre-trained word embeddings. Moreover, the proposed method successfully reduces the dimensionality of pre-trained word embeddings while maintaining high performance.

Introduction

Pre-trained word embeddings are a key component in natural language processing (NLP) downstream applications. Several algorithms have been for building word embeddings, such as Word2Vec (Mikolov et al. 2013), GloVe (Pennington, Socher, and Manning 2014), and fastText (Bojanowski et al. 2017). Recently, there has been increasing interest in improving the quality of pre-trained word embeddings for better performance in downstream applications. The post-processing operation was proposed as an effective technique to improve the pre-trained word embeddings without requiring additional training data. The post-processing techniques are based on the understanding of geometrical structures of the pre-trained word embedding space. Mu, Bhat, and Viswanath (2017) showed that pre-trained word embeddings are distributed in a narrow cone in the embedding space and they suggested that removing dominant principal components can improve the isotropy and the quality of pre-trained embeddings. Meanwhile, Kaneko and Bollegala (2020) showed that it is theoretically possible to improve pre-trained word embeddings by retaining the top principal components. Hence, it is not necessary to remove the largest principle components from pre-trained embeddings. Instead, they proposed to use an autoencoder to post-process the pre-trained word embeddings. The relationship between principal components and autoencoders has been explored by several prior works (Ladjal, Newson, and Pham 2019; Plaut 2018; Kunin et al. 2019). In line with those studies' findings, autoencoders can recover the principal components; this means the autoencoder can learn the subspace spanned by the top principal directions by minimizing the distance between the data and its reconstruction during training. However, the linear autoencoder only recovers the direction of principal components (Kunin et al. 2019), while the regularized autoencoder can find the exact principle components.

Variational dropout (Kingma, Salimans, and Welling 2015) is an effective regularization method, which uses Bayesian principles to define a variational dropout probability specifically for each neuron and weight. Practically, variational dropout provides a way to train the dropout rate by optimizing the variational lower bound of the loss function. As a result, the dropout rate becomes a variational parameter to be optimized rather than a hyperparameter. This technique allows us to train individual dropout rate for each layer, each neuron, and even each weight. Using a variational dropout rate can improve pre-trained embedding quality, especially when reducing embedding size. This can produce high-quality embeddings even when the embedding size is significantly reduced in size by half, which is not the case for a simple autoencoder model. Therefore, in this paper, we propose a novel post-processing method for pre-trained word embeddings using an orthogonal autoencoder with variational dropouts. Specifically, the proposed method utilizes variational dropout to increase its efficiency by producing word embeddings with better quality, even after drastically reducing the embedding's size. For example, when the size is reduced by half, the linear autoencoder and the non-linear autoencoder perform poorly while our method still retains its high performance. The experiments suggest that orthogonalization enhances the quality of pre-trained word embeddings. The evaluations on a variety of downstream NLP tasks such as semantic similarity, text classification, and concept categorization verify the effectiveness of the proposed method.

Related Work

Post-processing of word embeddings. Many post-processing algorithms have been developed to improve the

Copyright © 2023 by the authors. All rights reserved.

performance of pre-trained word embeddings. Different normalization techniques have been developed, including length normalization (Levy, Goldberg, and Dagan 2015), centering the mean (Sahlgren et al. 2016), etc. Wang et al. (2019) presented two normalization methods - variance normalization and dynamic embedding to learn orthogonal embeddings. The goal of these techniques is to reduce the variance of pre-trained word embeddings and make them distribute more evenly in high-dimensional space. Mu, Bhat, and Viswanath (2017) investigated the geometry of pre-trained word embeddings and showed that word embeddings are distributed in a narrow cone. They demonstrated that the performance can be improved by removing the top principal components. However, removing these components is likely to cause a loss on useful information, which may negatively impact different downstream tasks. Furthermore, post-processing algorithms are effective in improving the quality of contextual word embeddings, since they are far from being isotropic. Various studies (Rajaee and Pilehvar 2021; Liang et al. 2021) show that post-processing of BERT improves the isotropy and the performance of word embeddings. In addition, Raunak et al. (2020) showed that using PCA post-processing does not improve the performance of some downstream applications such as machine translation.

The connection between PCA and neural networks is well known. (Oja 1982) established that a neural network with a linear activation function essentially learns the principal component representation of the input data. Furthermore, (Oja 1992) extended this concept by using neural networks for learning independent components as well as minor components. A large number of studies have examined the relationship between autoencoders and PCA. These studies showed that an autoencoder with a single fully-connected layer, a linear activation function, and a squared error function can train the weights that span the same subspace as the principal subspace. Kaneko and Bollegala (2020) verified theoretically that retaining the top principle component is useful for improving pre-trained word embeddings. They experimented with a linear autoencoder and a non-linear autoencoder on a range of NLP tasks. An autoencoder can learn the direction of the word embeddings. Nonetheless, autoencoders suffer from overfitting to identity functions by generating output from input (Steck 2020). Therefore, we propose a regularized autoencoder that can learn optimal representations for pre-trained word embeddings using variational dropout with orthogonalization constraints.

Autoencoders in NLP. Autoencoders have been used in many successful applications. Several studies suggest that autoencoders follow the direction of the principal component (Kunin et al. 2019; Rippel, Gelbart, and Adams 2014; Bao et al. 2020) indicating that the autoencoder is capable of learning the direction of the principal components, but not able to learn the individual component and the corresponding eigenvectors. Kunin et al. (2019) suggested that applying l_2 regularization on the encoder and the decoder of the autoencoder can reduce the symmetry of the stationary point solutions to the group of orthogonal transformations. The individual principal component can be recovered by applying singular value decomposition (SVD). However, l_2 regularization convergence is ill-conditioned and worsens with the increased latent dimension (Bao et al. 2020). Selecting the appropriate regularized and gradient-based optimization can further break the symmetry of the stationary points solutions to the group of orthogonal transformations. Rippel, Gelbart, and Adams (2014) found that the exact PCA can be determined by applying nested dropout to the hidden units. Nested dropout highlights ordered information content in the hidden units. Therefore, we propose to use orthogonalization regularization and variational dropout, which have the potential to improve the performance. We show that these two regularizations can learn optimal representations and improve the accuracy in several downstream applications.

Variational Dropout. Variational Dropout (Kingma, Salimans, and Welling 2015) is a technique for training the dropout rate by optimizing the variational lower bound. In essence, the variational dropout rate becomes a variational parameter to be optimized rather than a simple hyperparameter. Molchanov, Ashukha, and Vetrov (2017) examined the effects of training individual dropout rates and concluded that variational dropout could effectively sparsify deep neural networks. Variational Dropout techniques have been adapted for a variety of applications, such as building a sparse and efficient model and avoiding overfitting in NLP and computer vision tasks (Gal and Ghahramani 2015; Sang and Hung 2019; Du et al. 2018). In addition, Variational Dropout was used for saliency maps and explainability applications (Chang et al. 2017).

Preliminary Information

Bayesian Inference

Bayesian techniques have been developed over many years in a wide range of domains, but have only recently applied to the problem of learning in neural network. The Bayesian approach can offer several potential advantages, including a solution to overfitting. It is worth noting that bayesian neural networks represent the weight as a distribution not scalar. The uncertainty of weight estimates makes the model more robust to overfitting (Hernández-Lobato and Adams 2015). From a probabilistic perspective, given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, our goal is to find a set of parameters $\mathbf{w}$ which can approximate the correct distribution of the output for a given input $(x_i, y_i) \in \mathcal{D}$. Bayesian learning assumes that there is some prior knowledge of model parameters $\mathbf{w}$ in the form of a prior distribution $p(\mathbf{w})$. Following Bayes' rule, we can obtain the posterior distribution after observing some data, which is $p(\mathbf{w}|\mathcal{D}) = p(\mathcal{D}|\mathbf{w})p(\mathbf{w})/p(\mathcal{D})$. Unfortunately, this is generally computationally intractable as it involves computing the marginal likelihood which is an intractable integral for a complex model and thus approximation approaches are needed. One popular approximation approach is variational inference, which uses a parametric distribution $q_\phi(\mathbf{w})$ to approximate the true posterior distribution $p(\mathbf{w}|\mathcal{D})$ (Kingma and Welling 2013) by minimizing the Kullback-Leibler divergence $D_{KL}(q_\phi(\mathbf{w})\|p(\mathbf{w}|\mathcal{D}))$. This can be realized by maxi-

mizing the variational lower bound $\mathcal{L}(\phi)$ defined as follows (Kingma, Salimans, and Welling 2015).

$$\mathcal{L}(\phi) = -D_{KL}\left(q_\phi(\mathbf{w}) \| p(\mathbf{w})\right) + L_{\mathcal{D}}(\phi) \tag{1}$$

where $L_{\mathcal{D}}(\phi)$ is called the expected log-likelihood and is calculated as follows.

$$\mathcal{L}_{\mathcal{D}}(\phi) = \sum_{i=1}^{N} \mathbb{E}_{q_\phi(\mathbf{w})}[\log p(y_i|\mathbf{x}_i, \mathbf{w})] \tag{2}$$

As pointed out by (Kingma, Salimans, and Welling 2015), the lower bound $\mathcal{L}(\phi)$ plus $D_{KL}(q_\phi(\mathbf{w})\|p(\mathbf{w}|\mathcal{D}))$ equals the marginal likelihood which is a constant, that means maximizing the lower bound will minimize $D_{KL}(q_\phi(\mathbf{w})\|p(\mathbf{w}|\mathcal{D}))$.

Variational Dropout

Consider a single fully-connected layer with I input neurons and O output neurons before the non-linear activation function and M is the minibatch size. We denote the input matrix as $A \in \mathbb{R}^{M\times I}$, the output matrix as $B \in \mathbb{R}^{M\times O}$ and the weight matrix as $W \in \mathbb{R}^{I\times O}$. Dropout is a popular regularization method for neural networks, which injects multiplicative random noise Ξ to the input layer at each iteration of the training procedure. Thus, the output matrix B can be specified as follows.

$$B = (A \odot \Xi)W, \text{ with } \xi_{ij} \sim p(\xi_{ij}) \tag{3}$$

where $\odot$ represents the element-wise (Hadamard) product. Dropout is a widely used technique to regularize deep neural networks. A dropout regularization scheme is initially referred to as Bernoulli or Binary Dropout with Bernoulli distribution $\mathcal{B}(1-p)$, which means that a unit is removed from the network, with all its incoming and outgoing connections, with probability p. Later, Gaussian dropout is introduced with continuous noise, which is found more beneficial than discrete noise with the same relative mean and variance, i.e., Gaussian distribution $\mathcal{N}(1, \alpha)$ with $\alpha = p/(1-p)$. Variational dropout was proposed with the idea of dropout rate becoming a variational parameter instead of a hyperparameter. In this case, the model is trained to learn individual dropout rates for each layer, neuron, or even weight.

Variational dropout is one of the most effective regularization methods for neural networks. In (Kingma, Salimans, and Welling 2015), the authors proposed to connect the dropout rate with Bayesian neural networks and show that different dropout rates ξ_{ij} can be learned for individual weight if ξ_{ij} is a Gaussian noise $\xi_{ij} \sim \mathcal{N}(1, \alpha_{ij})$ with $\alpha_{ij} = p_{ij}/(1-p_{ij})$. Therefore, each weight $w_{i,j}$ has the following Gaussian distribution parameterized by $\phi_{ij} = (\theta_{ij}, \alpha_{ij})$.

$$w_{ij} \sim \mathcal{N}\left(\theta_{ij}, \alpha_{ij}\theta_{ij}^2\right) = q_{\phi_{ij}}(w_{ij}) \tag{4}$$

where $\alpha_{ij}\theta_{ij}^2$ is the variance of the Gaussian distribution, and α_{ij} is enforced to be greater than zero. More details about the variational dropout technique are available in (Kingma, Salimans, and Welling 2015; Molchanov, Ashukha, and Vetrov 2017). Given in Eq.(4), the first term in Eq.(1) can be approximated by Monto Carlo method (Kingma, Salimans, and Welling 2015). To compute the second term in Eq.(1), the authors utilized a prior distribution $p(\mathbf{w})$ such that $D_{KL}\left(q_\phi(\mathbf{w})\|p(\mathbf{w})\right)$ depend only on α_{ij}. Therefore, the second term in Eq.(1) can be approximated as:

$$\begin{aligned} D_{KL}(q_\phi(\mathbf{w})\|p(\mathbf{w})) \approx \sum_{i,j}(-0.64\sigma(1.87 \\ + 1.49\log\alpha_{ij}) + 0.5\log(1+\alpha_{ij}^{-1})) + C \end{aligned} \tag{5}$$

where C is a constant and $\sigma(\cdot)$ is the sigmoid function (Molchanov, Ashukha, and Vetrov 2017).

Method

We present here our proposed model - Orthogonal AutoEncoder with Variational Dropout (OAEVD). Given a set of d-dimensional pre-trained word embeddings $\{\boldsymbol{e}_i\}_{i=1}^{|\mathcal{V}|}$ for a vocabulary $\mathcal{V}$, we post-process these word embeddings using the proposed model OAEVD, which adopts the autoencoder as a basic framework. An autoencoder, in its simplest form, has only one hidden layer shared by the encoder and decoder. The encoder projects the input data into the hidden space with lower dimensions and the decoder projects it back to the original feature space aiming to reconstruct the input data. We denote the encoder by $E : \mathcal{X} \rightarrow \mathcal{Z}$ and the decoder by $G : \mathcal{Z} \rightarrow \mathcal{X}$, where $\mathcal{X}$ and $\mathcal{Z}$ are original and hidden embedding spaces, respectively.

Training an autoencoder involves optimizing the parameters θ to minimize the reconstruction loss on the given dataset $\mathcal{D}$ and the reconstructed dataset $\mathcal{D}'$. Thus, the objective function is given as

$$L(\theta) = L(\mathcal{D}, \mathcal{D}') = L(\mathcal{D}, G(E(\mathcal{D}))) \tag{6}$$

where the loss function L$(\cdot, \cdot)$ denotes the squared Euclidean distance and the loss $L(\theta)$ is the average loss over all the word embedding samples and their reconstructions. To make the post-processed word embeddings more robust and enhance the expressiveness and consequently tasks' performance, we apply orthogonality to the hidden embedding space $\mathcal{Z}$. By regularizing the hidden representations with orthogonality loss during training, we gain consistent performance improvement in various downstream tasks. The orthogonality encourages more diversity in the latent feature space and increases the semantic similarity between similar words. Furthermore, we adopt the variational dropout technique to the proposed model to make it more robust to overfitting. Variational dropout can also make the neural network more sparse and reduce the noise in the model. Interestingly, variational dropout is helpful for finding the most important dimensions for word embeddings, especially with

Orthogonality

Motivated by PCA which enforces the orthogonality of the embeddings, we attempt to learn orthogonal embeddings. Orthogonality is a desirable property for various reasons, including better numerical stability and improved generalization. The orthogonality constraints encourage more diversity in the latent space and increases the semantic similarity between embeddings (Choi, Som, and Turaga 2020). It also preserves the vector norms and the isometry of the Euclidean

distance, which produces better performance (Smith et al. 2017). However, the optimization problem with orthogonality constraints, often referred to as the optimization problem in the Stiefel manifold, is difficult to solve except in very specific circumstances. A generalized backpropagation algorithm (GBP) is recently proposed in order to accommodate orthogonality constraints on the network's weights. Nevertheless, the GBP algorithm is not relevant to our model as it adds orthogonality constraints to the embeddings but not to the weights (Harandi and Fernando 2016).

To make the hidden representation $\mathcal{Z}$ matrix as orthogonal as possible, we need to regularize the off-diagonal elements to zero. The condition for the orthogonality is described below:

$$\langle \mathbf{z}_i, \mathbf{z}_{i'} \rangle = \begin{cases} 1, & \text{if } i = i' \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

$$L_{\mathrm{O}} = \sum_{i=1}^{N} \left\| {\mathbf{z}_i}^{\top} \mathbf{z}_i - \mathbf{I} \right\|_F^2 \tag{8}$$

Here, L_{O} is our orthogonal regularization, $\| \cdot \|_F$ represents the Frobenius norm and $\mathbf{I}$ is the $d \times d$ identity matrix.

Regularization with Variational Dropout

We utilize a variational dropout technique to avoid overfitting and make the neural network more sparse. By learning different dropout rates for individual weight, Molchanov, Ashukha, and Vetrov (2017) showed that the variational dropout technique can sparsify Bayesian neural networks by pruning weights with high learned dropout rates while still achieving the accuracy comparable to that of the unpruned weights. We use Bayesian neural network layers instead of standard neural network layers in the encoder and decoder of the autoencoder model. In section we provided a description specifics of this method. Utilizing variational dropout significantly improves the performance, particularly when the hidden representation space is set to be of smaller dimension than the input space.

On these grounds, the overall objective function of the proposed model (OAEVD) is defined as follows.

$$\begin{aligned} L_{\text{Total}} =& L(\theta) + \mu L_{\mathrm{O}} + \lambda D_{KL_{enc}}(q_\phi(\mathbf{w}) \| p(\mathbf{w})) \\ &+ \lambda D_{KL_{dec}}(q_\phi(\mathbf{w}) \| p(\mathbf{w})) \end{aligned} \tag{9}$$

where μ and λ are hyperparameters controlling the importance of each loss term. The values of μ and λ should be less than 1. $L(\theta)$ is the mean square error loss and L_{O} is the orthogonality constraint. $D_{KL_{enc}}(q_\phi(\mathbf{w}) \| p(\mathbf{w}))$ is the KL-divergence for the encoder layer, and $D_{KL_{dec}}(q_\phi(\mathbf{w}) \| p(\mathbf{w}))$ is the KL-divergence for the decoder layer.

Experiments

We conducted extensive experiments to assess the effectiveness of our proposed model (OAEVD), considered as a method to improve the quality of word embeddings in various downstream NLP tasks including semantic similarity, text classification, and concept categorization, and as a method to reduce the dimensionality of word embeddings. To evaluate our proposed method, we use three main pre-trained word embeddings: **Word2Vec** (300 dimensions) trained on Google News corpus (Mikolov et al. 2013), **GloVe** (300 dimensions) trained on Wikipedia 2014 and the Gigaword 5 corpus (400K vocabulary) (Pennington, Socher, and Manning 2014), and **fastText** (300 dimensions) trained on one million words from Common Crawl (Bojanowski et al. 2017).

We use a linear autoencoder and a non-linear autoencoder as baselines. The linear autoencoder is implemented as a single neural network layer for encoder and decoder, and the squared L2 regularization is minimized using the Adam optimizer. The tanh function is used for the non-linear autoencoder. Each encoder is implemented as a neural network with a hidden layer using Variations Dropout. We use the Adam optimizer with mini-batch size 256 and learning rate 0.00001 and train the model for 20 epochs. The hyperparameters are chosen based on experiments using the GloVe pre-trained embedding. The pre-trained word embedding used as input for the encoder's hidden layer is extracted as a new representation for the word embedding. The size of the hidden layer is 300D. We also implemented our proposed method with different hidden layer sizes, such as 200D and 150D.

Semantic Similarity

For evaluation in downstream NLP tasks, we start with the standard word similarity benchmarks described in (Faruqui and Dyer 2014). The datasets cover a wide range of similarity tasks in various domains, used to evaluate word similarity measures. The data set has word pairs (WP) that have been assigned similarity scores by human annotators. Cosine similarity is used to calculate the similarity between each pair of words, and then the Spearman's correlation coefficient $(Rho \times 100)$ between the model rankings and the human rankings is calculated. The detailed performance on the twelve datasets is reported in table 1, we can observe that our proposed method produces consistent improvement of semantic similarity across all the datasets over the original pre-trained word embedding and baseline methods linear autoencoder (LAE) and non-linear autoencoder(AE) with an average improvement of 2.99%. This improvement is suggested by orthogonalized hidden representation of autoencoder which leads to a strong semantic similarity between words. In addition, the use of variational dropout technique produces a better dropout rate for the autoencoder model and reduces the overfitting to the identity function.

Text Classification

For the classification task, we used the SentEval toolkit (Conneau and Kiela 2018). SentEval is a standard evaluation dataset for text classification and semantic analysis. It covers a wide range of domains. In SentEval, the sentences are represented by the mean of their word embeddings. Logistic regression and multi-layer perceptron are used as the main classifiers. The datasets allow binary classification (MR, CR, SUBJ, MPQA) and multi-class classification (SST-FG, TREC, Entailment Sick-E), and semantic relatedness (STS-B). Table 2 compares the accuracy of our method to the original pre-trained word embeddings and the baseline methods. It can be observed that our method improves the accuracy with an average performance of 0.5% across nine datasets.

Model	WS -353 -ALL	MTurk -771	WS -353 -REL	YP -130	RW -STAN -FORD	WS -353 -SIM	VE -RB -143	MTurk -287	SIM -LEX -999	SIM -MC-30	RG -65	MEN -TR -3k
Word2Vec-original	**69.18**	67.13	**62.19**	55.90	53.42	77.71	49.73	**68.40**	44.20	78.80	74.98	77.08
LAE	62.01	66.23	55.50	54.57	53.04	72.55	49.59	64.68	45.06	72.10	70.62	75.14
AE	68.53	66.10	61.12	55.44	53.70	**77.85**	44.87	66.92	44.00	77.07	75.83	76.75
OAEVD	66.74	**67.94**	58.41	**56.36**	**55.20**	76.40	**51.62**	65.00	**47.09**	**78.71**	**77.00**	**77.44**
GloVe-original	60.54	65.01	57.26	56.13	41.18	66.38	30.51	63.32	37.05	70.26	76.62	73.75
LAE	59.81	62.38	57.24	56.54	38.93	64.93	31.72	62.38	36.08	72.28	73.46	71.65
AE	64.14	64.63	60.51	**59.12**	44.25	70.17	30.00	64.43	39.18	**79.94**	76.67	74.27
OAEVD	**66.10**	**68.43**	**62.87**	58.67	46.09	**71.56**	**34.07**	**66.18**	**40.65**	73.31	**77.72**	**77.09**
fastText-original	70.84	71.01	65.00	51.87	52.33	81.02	46.27	70.50	45.00	83.63	**84.51**	79.06
LAE	68.62	70.73	61.68	52.05	52.64	79.79	46.95	70.61	45.24	82.70	82.63	78.33
AE	67.18	68.71	61.97	51.59	49.37	78.61	43.37	66.88	45.97	**88.63**	82.39	76.20
OAEVD	**73.69**	**73.99**	**68.76**	**55.77**	**56.45**	**81.81**	**46.45**	**70.65**	**48.61**	86.86	84.34	**80.99**

Table 1: Performance on a similarity task for the original embeddings and their post-processed versions by linear autoencoder (LAE), nonlinear autoencoder (AE) and our model (OAEVD) for the pre-trained Word2Vec, GloVe and fastText embeddings.

Model	MR	CR	MPQA	SUBJ	STS-B	SST-FG	TREC	SICK-E	MRPC
Word2Vec-orginal	**76.97**	**79.34**	**88.29**	90.46	81.11	42.51	82.60	77.92	71.65
LAE	76.50	78.94	88.03	90.01	80.89	43.76	82.60	78.71	71.83
AE	76.19	77.54	87.69	89.88	81.23	43.76	80.60	77.53	69.62
OAEVD	76.86	78.28	88.18	**90.61**	**81.56**	**42.63**	**83.80**	**78.88**	**72.52**
GloVe-orginal	74.99	75.81	86.35	91.04	**78.20**	40.77	66.60	77.19	72.46
LAE	75.29	76.05	86.45	**91.26**	78.31	41.58	69.40	77.76	72.35
AE	74.31	76.71	86.05	90.72	76.94	41.31	69.40	**78.51**	**72.78**
OAEVD	**75.56**	**75.95**	**86.93**	91.02	77.92	**42.08**	**71.20**	78.24	72.59
fastText-orginal	**77.65**	**80.48**	87.78	92.10	82.15	44.30	84.40	**79.60**	**74.38**
LAE	76.94	77.83	87.75	90.74	81.33	42.94	82.80	77.35	72.04
AE	75.62	77.99	86.82	91.26	80.67	44.48	80.00	77.43	72.06
OAEVD	76.92	78.53	**87.92**	**92.33**	**82.47**	**45.16**	**85.27**	78.77	73.64

Table 2: Classification performance of the original pre-trained embedding models (Word2Vec, GloVe, and fastText) and their post-processed versions by linear autoencoder (LAE), autoencoder (AE), and our model (OAEVD).

Concept Categorization

Given a set of concepts, the algorithm needs to cluster words into different categories. The clustering performance is evaluated based on the cluster purity measure (Christopher, Prabhakar, and Hinrich 2008) that is based on the fraction of the total number of objects that were classified correctly. We selected three datasets: the Almuhareb-Poesio (AP) dataset (Almuhareb 2006) which contains 402 concepts in 21 categories; the ESSLLI 2008 Distributional Semantic Index; and the ESSLLI 2008 Conceptual Analysis Workshop shared-task dataset (Katrenko, Adriaans, and others 2008) that contains 44 concepts in 6 categories; and the Batting test set (Baroni and Lenci 2010) that contains 83 words in 10 categories. We follow the same setting and algorithms as (Mu, Bhat, and Viswanath 2017; Baroni, Dinu, and Kruszewski 2014) to cluster words (based upon their representations) using k-means (for fixed k) and report the results in Table 3. We found that our method consistently outperformed the baseline method (AE) on the three datasets with an average improvement of 3.70%. The results in Table 3 indicate that our method contributes to improving concept categorization as it produces a steady and remarkably better purity score compared to baseline method AE.

Model	AP	Batting	Bliss
Word2Vec	47.85	71.26	71.85
AE	48.86	70.79	74.83
OAEVD	**51.38**	**72.76**	**74.87**
GloVe	55.97	**59.99**	73.52
AE	56.64	59.47	**77.59**
OAEVD	**56.78**	57.4	77.53
fastText	83.58	87.13	97.01
AE	83.10	86.24	95.53
OAEVD	**85.32**	**88.59**	**99.15**

Table 3: The performance of original embeddings, AE, and OAEVD in concept categorization task with three main embedding algorithms (Word2Vec, GloVe and fastText.

Dimensionality Reduction

We evaluate the performance of the proposed method with smaller embedding dimensions. To train the model, we select 300 dimensions as input, and we select two different hidden dimensions (200D and 150D). These numbers were chosen based on the trade-off between complexity and performance. We use semantic similarity measures to evaluate the effectiveness of our model in dimensionality reduction. The proposed method is compared with autoencoder approaches and the original pre-trained embeddings on the three main embeddings algorithms (Word2vec, GloVe, and fastText). The experimental results are compared in Table 4 and Table 5. We can observe that the autoencoder approaches perform well when the embedding size is reduced to 200D with accuracy comparable to the original 300D on semantic similarity tasks. However, they perform poorly when the embedding size is

Model	WS -353 -ALL	MTurk -771	WS -353 -REL	YP -130	RW -STAN -FORD	WS -353 -SIM	VE -RB -143	MTurk -287	SIM -LEX -999	SIM -MC -30	RG -65	MEN -TR -3k
Word2Vec-300D	**69.18**	67.13	62.19	**55.90**	53.42	**77.71**	**49.73**	**68.40**	44.20	78.80	74.98	77.08
LAE-200D	68.84	65.83	**62.30**	52.22	52.68	76.83	43.18	67.87	42.01	76.93	77.51	77.65
AE-200D	68.43	66.67	61.04	52.86	50.00	76.06	40.41	62.13	42.30	73.64	76.08	75.56
OAEVD-200D	69.06	**68.44**	62.27	55.12	**53.69**	77.15	48.95	65.98	**44.96**	**82.25**	**79.45**	**78.68**
GloVe-300D	60.54	65.01	57.26	56.13	41.18	66.38	30.51	63.32	37.05	70.26	**76.62**	73.75
LAE-200D	60.51	63.86	56.17	56.08	40.08	65.93	31.88	62.00	35.69	73.04	76.36	73.43
AE-200D	63.99	63.29	59.43	58.17	42.01	68.80	36.06	64.84	40.86	70.61	76.32	76.02
OAEVD-200D	**65.38**	**67.98**	**61.36**	**63.54**	**45.11**	**71.01**	**38.38**	**65.24**	**40.64**	**70.64**	74.89	**76.97**
fastText-300D	70.84	71.01	65.00	51.87	52.33	81.02	46.27	**70.50**	45.00	83.63	84.51	79.06
LAE-200D	68.35	69.79	61.86	48.58	50.02	79.07	46.99	69.37	42.25	85.97	86.11	77.61
AE-200D	65.69	70.02	60.92	51.51	50.56	79.51	47.46	68.35	41.41	84.17	84.83	78.43
OAEVD-200D	**74.67**	**73.24**	**70.73**	**55.30**	**54.30**	**81.40**	**49.68**	70.07	**49.68**	**91.18**	**89.03**	**81.49**

Table 4: Comparison of the proposed model (OAEVD) with 200 dimensions with original embeddings, LAE and AE of the three main algorithms Word2Vec, GloVe, and fastText.

Model	WS -353 -ALL	MTurk -771	WS -353 -REL	YP -130	RW -STAN -FORD	WS -353 -SIM	VE -RB -143	MTurk -287	SIM -LEX -999	SIM -MC -30	RG -65	MEN -TR -3k
Word2Vec-300D	69.18	67.13	62.19	**55.90**	53.42	77.71	**49.73**	**68.40**	**44.20**	78.80	74.98	77.08
LAE-150D	67.82	64.91	60.89	50.04	50.65	75.58	37.16	64.98	39.92	83.41	78.25	77.18
AE-150D	64.44	66.14	61.25	47.79	50.30	73.36	40.77	68.25	40.79	79.49	80.11	77.35
OAEVD-150D	**69.59**	**68.72**	**63.48**	53.10	**53.53**	**78.20**	44.52	66.29	42.62	**85.79**	**81.69**	**78.16**
GloVe-300D	60.54	**65.01**	57.26	56.13	41.18	66.38	30.51	63.32	37.05	70.26	76.62	73.75
LAE-150D	54.20	57.82	48.94	49.93	35.67	60.58	26.90	57.77	32.80	72.10	75.25	69.72
AE-150D	60.37	58.79	56.70	56.42	37.91	65.73	37.34	**63.46**	37.29	73.30	76.04	70.12
OAEVD-150D	**64.43**	64.59	**59.05**	**59.55**	**42.99**	**70.68**	**41.74**	61.98	**38.62**	**75.46**	**80.12**	**75.38**
fastText-300D	70.84	**71.01**	65.00	51.87	52.33	**81.02**	46.27	70.50	**45.00**	83.63	84.51	79.06
LAE-150D	63.42	65.67	55.15	44.17	47.57	75.34	47.35	69.05	39.17	82.36	81.17	76.00
AE-150D	65.06	63.92	57.44	45.61	47.95	74.17	44.56	69.60	39.61	85.21	85.51	75.91
OAEVD-150D	**73.38**	70.74	**68.09**	**54.08**	**52.73**	80.19	**47.09**	**71.50**	43.51	**91.06**	**89.40**	**80.80**

Table 5: Comparison of the proposed model (OAEVD) with 150 dimensions with original embeddings, LAE, and AE of the three main algorithms Word2Vec, GloVe, and fastText.

reduced to 150D. At the same time, our method achieves better performance and outperforms the autoencoder approaches on most datasets with 200D and 150D embeddings. We obtain consistent performance improvement in dimensionality reduction with an average improvement of 4.5% for 200D and 2.99% for 150D.

Isotrophy of Word Embeddings

We demonstrate the effectiveness of our method on the geometric properties of the word embeddings by enhancing the isotropy of the word embeddings. Isotropy is a useful property for vectors in any data not only for word embeddings. For a vector to be isotropic, its values have to be uniformly distributed in all directions. Previous research showed that isotropy for word embeddings can improve their performance. We utilize the isotropy measure (Arora et al. 2016).

$$\frac{\min_{c\in\mathcal{C}} F(\boldsymbol{c})}{\max_{\boldsymbol{c}\in\mathcal{C}} F(\boldsymbol{c})} \tag{10}$$

where C is the set of principle component vectors for a given set of pre-trained word embedding. $F(\boldsymbol{c}) = \sum_{\boldsymbol{e}\in\mathcal{V}} \exp(\boldsymbol{c}^\top \boldsymbol{e})$ is the normalization coefficient in the partition function. Table 6 shows the comparison between the baseline autoencoders (LAE and AE) and our proposed method (OAEVD). All techniques based on autoencoders improve the isotropy of the word embedding over the original embedding.

Model	Word2Vec	GloVe	fastText
Original	0.489	0.096	0.600
LAE	0.960	0.703	0.970
AE	**0.976**	0.782	**0.990**
OAEVD	0.954	**0.908**	0.900

Table 6: The isotropy measure of word embeddings for original embeddings, LAE, AE and OAEVD.

Conclusion and Future Work

This paper presents an effective method for post-processing word embedding using a regularized autoencoder based on orthogonality constraints and variational dropout. Our approach outperforms simple autoencoders in semantic similarity and text classification tasks. Additionally, we produce more isotropic representations, which are desirable properties for representing words. The proposed method successfully reduces dimensionality and produces smaller-size embeddings with good performance, which is a critical factor when the resources are limited. In an online environment, autoencoders could be implemented by using a small batch of words at a time, allowing them to handle new data without processing all the data at once as in a post-processing method based on the PCA algorithm. Further research along this line will be to investigate using this approach to post-process contextual word embeddings and sentence embeddings to improve their isotropy, and, eventually, the performance of downstream applications.

References

Almuhareb, A. 2006. *Attributes in lexical acquisition*. Ph.D. Dissertation, University of Essex.

Arora, S.; Li, Y.; Liang, Y.; Ma, T.; and Risteski, A. 2016. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*.

Bao, X.; Lucas, J.; Sachdeva, S.; and Grosse, R. 2020. Regularized linear autoencoders recover the principal components, eventually. *arXiv preprint arXiv:2007.06731*.

Baroni, M., and Lenci, A. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*.

Baroni, M.; Dinu, G.; and Kruszewski, G. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*.

Chang, C.-H.; Creager, E.; Goldenberg, A.; and Duvenaud, D. 2017. Interpreting neural network classifications with variational dropout saliency maps. In *Advances in neural information processing systems*.

Choi, H.; Som, A.; and Turaga, P. 2020. Role of orthogonality constraints in improving properties of deep networks for image classification. *arXiv preprint arXiv:2009.10762*.

Christopher, D. M.; Prabhakar, R.; and Hinrich, S. 2008. Introduction to information retrieval.

Conneau, A., and Kiela, D. 2018. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*.

Du, W.; Zeng, X.; Yan, M.; and Zhang, M. 2018. Efficient federated learning via variational dropout.

Faruqui, M., and Dyer, C. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*.

Gal, Y., and Ghahramani, Z. 2015. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*.

Harandi, M., and Fernando, B. 2016. Generalized backpropagation, Etude de cas: Orthogonality. *arXiv preprint arXiv:1611.05927*.

Hernández-Lobato, J. M., and Adams, R. 2015. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International conference on machine learning*.

Kaneko, M., and Bollegala, D. 2020. Autoencoding improves pre-trained word embeddings. In *International Committee on Computational Linguistics*.

Katrenko, S.; Adriaans, P.; et al. 2008. Qualia structures and their impact on the concrete noun categorization task. In *Proceedings of the ESSLLI Workshop on Distributional Lexical Semantics*.

Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kingma, D. P.; Salimans, T.; and Welling, M. 2015. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*.

Kunin, D.; Bloom, J.; Goeva, A.; and Seed, C. 2019. Loss landscapes of regularized linear autoencoders. In *International Conference on Machine Learning*.

Ladjal, S.; Newson, A.; and Pham, C.-H. 2019. A pca-like autoencoder. *arXiv preprint arXiv:1904.01277*.

Levy, O.; Goldberg, Y.; and Dagan, I. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the association for computational linguistics*.

Liang, Y.; Cao, R.; Zheng, J.; Ren, J.; and Gao, L. 2021. Learning to remove: Towards isotropic pre-trained bert embedding. *arXiv preprint arXiv:2104.05274*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*.

Molchanov, D.; Ashukha, A.; and Vetrov, D. 2017. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning*.

Mu, J.; Bhat, S.; and Viswanath, P. 2017. All-but-the-top: Simple and effective postprocessing for word representations. *arXiv preprint arXiv:1702.01417*.

Oja, E. 1982. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*.

Oja, E. 1992. Principal components, minor components, and linear neural networks. *Neural networks*.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing*.

Plaut, E. 2018. From principal subspaces to principal components with linear autoencoders. *arXiv preprint arXiv:1804.10253*.

Rajaee, S., and Pilehvar, M. T. 2021. An isotropy analysis in the multilingual bert embedding space. *arXiv preprint arXiv:2110.04504*.

Raunak, V.; Kumar, V.; Gupta, V.; and Metze, F. 2020. On dimensional linguistic properties of the word embedding space. *Association for Computational Linguistics*.

Rippel, O.; Gelbart, M.; and Adams, R. 2014. Learning ordered representations with nested dropout. In *International Conference on Machine Learning*.

Sahlgren, M.; Gyllensten, A. C.; Espinoza, F.; Hamfors, O.; Karlgren, J.; Olsson, F.; Persson, P.; Viswanathan, A.; and Holst, A. 2016. The gavagai living lexicon. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*.

Sang, D. V., and Hung, D. V. 2019. Yolov3-vd: A sparse network for vehicle detection using variational dropout. In

Proceedings of the Tenth International Symposium on Information and Communication Technology.

Smith, S. L.; Turban, D. H.; Hamblin, S.; and Hammerla, N. Y. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*.

Steck, H. 2020. Autoencoders that don't overfit towards the identity. *Advances in Neural Information Processing Systems*.

Wang, B.; Chen, F.; Wang, A.; and Kuo, C.-C. J. 2019. Post-processing of word representations via variance normalization and dynamic embedding. In *2019 IEEE International Conference on Multimedia and Expo*.