Autoencoding Improves Pre-trained Word Embeddings

Masahiro Kaneko
Tokyo Metropolitan University
kaneko-masahiro@ed.tmu.ac.jp

Danushka Bollegala*
University of Liverpool, Amazon
danushka@liverpool.ac.uk

Abstract

Prior work investigating the geometry of pre-trained word embeddings have shown that word embeddings to be distributed in a narrow cone and by centering and projecting using principal component vectors one can increase the accuracy of a given set of pre-trained word embeddings. However, theoretically this post-processing step is equivalent to applying a linear autoencoder to minimise the squared ℓ_2 reconstruction error. This result contradicts prior work (Mu and Viswanath, 2018) that proposed to *remove* the top principal components from pre-trained embeddings. We experimentally verify our theoretical claims and show that retaining the top principal components is indeed useful for improving pre-trained word embeddings, without requiring access to additional linguistic resources or labeled data.

1 Introduction

Pre-trained word embeddings have been successfully used as features for representing input texts in many NLP tasks (Dhillon et al., 2015; Mnih and Hinton, 2009; Collobert et al., 2011; Huang et al., 2012; Mikolov et al., 2013; Pennington et al., 2014). Mu and Viswanath (2018) showed that the accuracy of pre-trained word embeddings can be further improved in a post-processing step, without requiring additional training data, by removing the mean of the word embeddings (*centering*) computed over the set of words (i.e. vocabulary) and projecting onto the directions defined by the principal component vectors, excluding the top principal components. They empirically showed that pre-trained word embeddings are distributed in a narrow cone around the mean embedding vector, and centering and projection help to reinstate isotropy in the embedding space. This post-processing operation has been repeatedly proposed in different contexts such as with distributional (counting-based) word representations (Sahlgren et al., 2016) and sentence embeddings (Arora et al., 2017).

Independently to the above, autoencoders have been widely used for fine-tuning pre-trained word embeddings such as for removing gender bias (Kaneko and Bollegala, 2019), meta-embedding (Bao and Bollegala, 2018), cross-lingual word embedding (Wei and Deng, 2017) and domain adaptation (Chen et al., 2012), to name a few. However, it is unclear whether better performance is obtained simply by applying an autoencoder (a self-supervised task, requiring no labelled data) on pre-trained word embeddings, without performing any task-specific fine-tuning (requires labelled data for the task).

A connection between principal component analysis (PCA) and linear autoencoders was first proved by Baldi and Hornik (1989), extending the analysis by Bourlard and Kamp (1988). We revisit this analysis and theoretically prove that one must *retain* the largest principal components instead of removing them as proposed by Mu and Viswanath (2018) in order to minimise the squared ℓ_2 reconstruction loss.

Next, we experimentally show that by applying a non-linear autoencoder we can post-process a given set of pre-trained word embeddings and obtain more accurate word embeddings than by the method proposed by Mu and Viswanath (2018). Although Mu and Viswanath (2018) motivated the removal of

*Danushka Bollegala holds concurrent appointments as a Professor at University of Liverpool and as an Amazon Scholar. This paper describes work performed at the University of Liverpool and is not associated with Amazon.

Proceedings of the 28th International Conference on Computational Linguistics, pages 1699–1713
Barcelona, Spain (Online), December 8-13, 2020

largest principal components as a method to improve the isotropy of the word embeddings, our empirical findings show that autoencoding automatically improves isotropy.

2 Autoencoding as Centering and PCA Projection

Let us consider a set of n-dimensional pre-trained word embeddings, $\{\boldsymbol{x}_i\}_{i=1}^{N}$ for a vocabulary, $\mathcal{V}$, consisting of N words. We post-process these pre-trained word embeddings using an autoencoder consisting of a single $p(< n)$ dimensional hidden layer, an encoder (defined $\mathbf{W}_e \in \mathbb{R}^{n \times p}$ and bias $\boldsymbol{b}_e \in \mathbb{R}^p$) and a decoder (defined by $\mathbf{W}_d \in \mathbb{R}^{p \times n}$ and bias $\boldsymbol{b}_d \in \mathbb{R}^n$). Let $\mathbf{X} \in \mathbb{R}^{n \times N}$ be the embedding matrix. Using matrices $\mathbf{B} \in \mathbb{R}^{p \times N}$, $\mathbf{H} \in \mathbb{R}^{p \times N}$ and $\mathbf{Y} \in \mathbb{R}^{n \times N}$ respectively denoting the activations, hidden states and reconstructed output embeddings, the autoencoder can be specified as follows.

$$\mathbf{B} = \mathbf{W}_e\mathbf{X} + \boldsymbol{b}_e\boldsymbol{u}^\top, \qquad \mathbf{H} = F(\mathbf{B}), \qquad \mathbf{Y} = \mathbf{W}_d\mathbf{H} + \boldsymbol{b}_d\boldsymbol{u}^\top$$

Here, $\boldsymbol{u} \in \mathbb{R}^N$ is a vector consisting of ones and F is an element-wise activation function. The squared ℓ_2 reconstruction loss, J, for the autoencoder is given by (1).

$$J(\mathbf{W}_e, \mathbf{W}_d, \boldsymbol{b}_e, \boldsymbol{b}_d) = \left|\left|\mathbf{W}_d F(\mathbf{W}_e\mathbf{X} + \boldsymbol{b}_e\boldsymbol{u}^\top) + \boldsymbol{b}_d\boldsymbol{u}^\top\right|\right|^2 \tag{1}$$

The reconstruction loss of the autoencoder is given by Lemma 1, proved in the appendix.

Lemma 1. *Let $\mathbf{X}'$ and $\mathbf{H}'$ respectively denote the centred embedding and hidden state matrices. Then,* (1) *can be expressed using $\mathbf{X}'$ and $\mathbf{H}'$ as $J(\mathbf{W}_e, \mathbf{W}_d, \boldsymbol{b}_e, \hat{\boldsymbol{b}}_d) = ||\mathbf{X}' - \mathbf{W}_d\mathbf{H}'||^2$, where the decoder's optimal bias vector is given by $\hat{\boldsymbol{b}}_d = \frac{1}{N}(\mathbf{X} - \mathbf{W}_d\mathbf{H})\,\boldsymbol{u}$.*

Lemma 1 holds even for non-linear autoencoders and claims that the centering happens automatically during the minimisation of the reconstruction error. Following Lemma 1, we can assume that the embedding matrix, $\mathbf{X}$, to be already centred and can limit further discussions to this case. Moreover, after centering the input embeddings, the biases can be *absorbed* into the encoder/decoder matrices by setting an extra dimension that is always equal to 1 in the pre-trained word embeddings. This has the added benefit of simplifying the notations and proofs. Under these conditions Theorem 2 shows an important connection between linear autoencoders and PCA.

Theorem 2. *Assume that $\mathbf{\Sigma}_{xx} = \mathbf{X}\mathbf{X}^\top$ is full-rank with n distinct eigenvalues $\lambda_1 > \ldots > \lambda_n$. Let $\mathcal{I} = \{i_1, \ldots, i_p\}$ $(1 \leq i_1 < \ldots < i_p \leq n)$ be any ordered p-index set, and $\mathbf{U}_\mathcal{I} = [\boldsymbol{u}_{i_1}, \ldots \boldsymbol{u}_{i_p}]$ denote the matrix formed by the orthogonal eigenvectors of $\mathbf{\Sigma}_{xx}$ associated with the eigenvalues $\lambda_{i_1}, \ldots, \lambda_{i_p}$. Then, two full-rank matrices $\mathbf{W}_d$ and $\mathbf{W}_e$ define a critical point of* (1) *for a linear autoencoder if and only if there exists an ordered p-index set $\mathcal{I}$ and an invertible matrix $\mathbf{C} \in \mathbb{R}^{p \times p}$ such that*

$$\mathbf{W}_d = \mathbf{U}_\mathcal{I}\mathbf{C} \tag{2}$$

$$\mathbf{W}_e = \mathbf{C}^{-1}\mathbf{U}_\mathcal{I}^{-1}. \tag{3}$$

Moreover, the reconstruction error, $J(\mathbf{W}_e, \mathbf{W}_d)$ can be expressed as

$$J(\mathbf{W}_e, \mathbf{W}_d) = \mathrm{tr}(\mathbf{\Sigma}_{xx}) - \sum_{t \in \mathcal{I}} \lambda_t. \tag{4}$$

Proof of Theorem 2 and approximations for non-linear activations are given in the appendix. Because $\mathbf{\Sigma}_{xx}$ is a covariance matrix, it is positive semi-definite. Strict positivity corresponds to it being full-rank and is usually satisfied in practice for pre-trained word embeddings, which are dense and use a small $n(\ll N)$ independent dimensions for representing the semantics of the words. Moreover, $\mathbf{W}_e$, $\mathbf{W}_d$ are randomly initialised in practice making them full-rank as assumed in Theorem 2.

The connection between linear autoencoders and PCA was first proved by Baldi and Hornik (1989), extending the analysis by Bourlard and Kamp (1988). Reconstructing the principal component vectors from an autoencoder has been discussed by Plaut (2018) without any formal proofs. However, to the

best of our knowledge, a theoretical justification for post-processing pre-trained word embeddings by autoencoding has not been provided before.

According to Theorem 2, we can minimise (4) by selecting the largest eigenvalues as λ_t. This result contradicts the proposal by Mu and Viswanath (2018) to project the word embeddings away from the largest principal component vectors, which is motivated as a method to improve isotropy in the word embedding space. They provided experimental evidence to the effect that largest principal component vectors encode word frequency and removal of them is not detrimental to semantic tasks such as semantic similarity measurement and analogy detection. However, the frequency of a word is an important piece of information for tasks that require differentiating stop words and content words such as in information retrieval. Raunak et al. (2020) demonstrated that removing the top principal components does not necessarily lead to performance improvement. Moreover, contextualised word embeddings such as BERT (Devlin et al., 2019) and Elmo (Peters et al., 2018) have shown to be anisotropic despite their superior performance in a wide-range of NLP tasks (Ethayarajh, 2019). Therefore, it is not readily obvious whether removing the largest principal components to satisfy isotropy is a universally valid strategy. On the other hand, our experimental results show that by autoencoding not only we obtain better embeddings than Mu and Viswanath (2018), but also it improves the isotropy of the pre-trained word embeddings.

3 Experiments

Parameter	Value
Optimizer	Adam
Learning rate	0.0002
Dropout rate	0.2
Batch size	256
Activation function	$\tanh$

Table 1: Hyperparameter values of the autoencoder.

To evaluate the proposed post-processing method, we use the following pre-trained word embeddings: **Word2Vec**[1] (300-dimensional embeddings for ca. 3M words learnt from the Google News corpus), **GloVe**[2] (300-dimensional word embeddings for ca. 2.1M words learnt from the Common Crawl), and **fastText**[3] (300-dimensional embeddings for ca. 2M words learnt from the Common Crawl).

We use the following benchmarks datasets: for semantic similarity **WS-353**; Agirre et al. (2009), **SIMLEX-999**; Hill et al. (2015), **RG-65**; Rubenstein and Goodenough (1965), **MTurk-287**; Radinsky et al. (2011), **MTurk-771**; Halawi et al. (2012) and **MEN**; Bruni et al. (2014), for analogy **Google**, **MSR** (Mikolov et al., 2013), and **SemEval**; Jurgens et al. (2012)) and for concept categorisation **BLESS**; Baroni and Lenci (2011) and **ESSLI**; Baroni et al. (2008)) to evaluate word embeddings.

Table 1 lists the hyperparameters and their values for the autoencoder-based post-processing method used in the experiments. We used the syntactic analogies in the **MSR**; Mikolov et al. (2013) dataset for setting the hyperparameters. We input each set of embeddings separately to an autoencoder with one hidden layer and minimise the squared ℓ_2 error using Adam as the optimiser. The pre-trained embeddings are then sent through the trained autoencoder and its hidden layer outputs are used as the post-processed word embeddings. We train an autoencoder (denoted as **AE**) with a 300-dimensional hidden layer and a $\tanh$ activation. Moreover, to study the effect of nonlinearities we train the a linear autoencoder (**LAE**) without using any nonlinear activation functions in its 300-dimensional hidden layer. Due to space limitations, we show results for autoencoders with different hidden layer sizes in the appendix. We compare the embeddings post-processed using **ABTT** (stands for *all-but-the-top*) (Mu and Viswanath, 2018), which removes the top principal components from the pre-trained embeddings.

Table 2 compares the performance of the **Original** embeddings against the embeddings post-processed using **ABTT**, **LAE** and **AE**. For the semantic similarity task, a high degree of Spearman correlation between human similarity ratings and the cosine similarity scores computed using the word embeddings is considered as better. From Table 2 we see that **AE** improves word embeddings and outperforms **ABTT** in

[1] https://code.google.com/archive/p/word2vec/

[2] https://github.com/stanfordnlp/GloVe

[3] https://fasttext.cc/docs/en/english-vectors.html

Embedding	Word2Vec				GloVe				fastText			
Dataset	Original	ABTT	LAE	AE	Original	ABTT	LAE	AE	Original	ABTT	LAE	AE
WS-353	**62.4**	61.2	61.8	61.8	60.6	61.5	64.0	**65.8**	65.9	67.7	**69.0**	**69.0**
SIMLEX-999	44.7	45.4	**45.5**	**45.5**	39.5	41.5	40.8	**42.2**	46.2	47.4	**48.8**	**48.8**
RG-65	75.4	76.0	76.2	**76.3**	68.1	68.0	71.4	**72.3**	78.4	**81.4**	80.4	80.5
MTurk-287	**69.0**	68.9	**69.0**	68.9	71.8	71.9	73.6	**74.4**	73.3	73.8	**74.7**	**74.7**
MTurk-771	63.1	63.7	63.8	**63.9**	62.7	63.7	66.2	**67.7**	69.6	71.8	72.3	**72.4**
MEN	68.1	68.3	69.2	**69.3**	67.7	69.5	73.0	**74.8**	71.1	75.7	75.9	**76.0**
MSR	**73.6**	73.2	73.5	73.4	73.8	73.2	74.3	**74.4**	87.1	**88.0**	87.3	87.3
Google	74.0	**74.8**	74.3	74.3	76.8	76.9	**77.2**	77.1	85.3	**88.0**	86.4	86.4
SemEval	20.0	19.9	**20.4**	20.3	15.4	17.2	17.2	**17.6**	21.0	23.2	23.2	**23.3**
BLESS	70.5	**71.0**	68.5	70.0	76.5	76.5	75.0	**79.5**	75.5	79.0	79.5	**80.5**
ESSLI	75.5	73.7	73.8	**76.2**	72.2	72.2	**73.0**	**73.0**	74.7	76.2	76.1	**77.0**

Table 2: Results are shown for the original embeddings and their post-processed versions by **ABTT**, linear autoencoder (**LAE**) and nonlinear autoencoder (**AE**) for pre-trained Word2Vec, GloVe and fastText embeddings.

almost all semantic similarity datasets. For the word analogy task, we use the PairDiff method (Levy and Goldberg, 2014) to predict the fourth word needed to complete a proportional analogy and the accuracy of the prediction is reported. For the word analogy task, we see that for the GloVe embeddings **AE** reports the best performance but **ABTT** performs better for fastText. Overall, the improvements due to post-processing are less prominent in the word analogy task. This behaviour was also observed by Mu and Viswanath (2018) and is explained by the fact that analogy solving is done using vector difference, which is not influenced by centering.

In the concept categorisation task, we measure the Euclidean distance between two words, computed using their embeddings as the distance measure, and use the k-means clustering algorithm to group words into clusters separately in each benchmark dataset. Cluster purity (Manning et al., 2008) is computed as the evaluation measure using the gold category labels provided in each benchmark dataset. High values of purity would indicate that the word embeddings capture information related to the semantic classes of words. From Table 2 we see that **AE** outperforms **ABTT** in all cases, except on BLESS with Word2Vec embeddings.

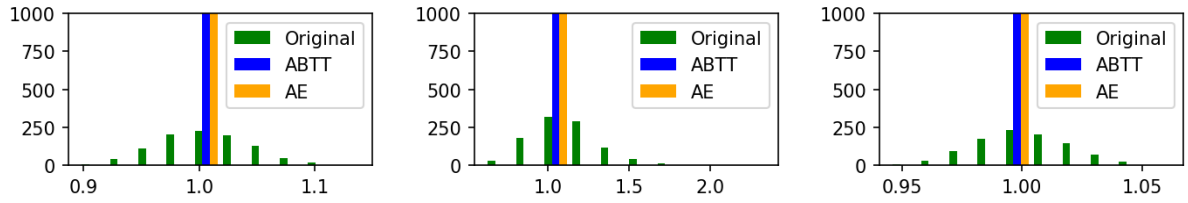
From Table 2 we see that for the pre-trained GloVe and fastText embeddings, both linear (**LAE**) and non-linear autoencoders (**AE**) yield consistently better post-processed embeddings than the original embeddings. For the pre-trained Word2Vec embeddings, we see that using **LAE** or **AE** produces better embeddings in seven out of the eleven benchmark datasets. However, according to $p < 0.05$ Fisher transformation we see that the performance difference between **LAE** and **AE** is not statistically significant in most datasets. Considering the theoretical equivalence between PCA and linear autoencoders, this result shows that it is more important to perform centering and apply PCA rather than using a non-linear activation in the hidden layer of the autoencoder.

	Original	ABTT	LAE	AE
Word2Vec	0.489	0.981	0.963	0.976
GloVe	0.018	0.943	0.782	0.884
fastText	0.773	0.995	0.992	0.990

Table 3: The measure of isotropy of original embeddings and after post-processed using **ABTT** and **AE**.

Following the definition given by Mu and Viswanath (2018), we empirically estimate the isotropy of a set of embeddings as $\gamma = \frac{\min_{\boldsymbol{c}\in\mathcal{C}} Z(\boldsymbol{c})}{\max_{\boldsymbol{c}\in\mathcal{C}} Z(\boldsymbol{c})}$, where $\mathcal{C}$ is the set of principal component vectors computed for the given set of pre-trained word embeddings and $Z(\boldsymbol{c}) = \sum_{x\in\mathcal{V}} \exp(\boldsymbol{c}^\top \boldsymbol{x})$ is the normalisation coefficient in the partition function defined by Arora et al. (2016). γ values close to one indicate a high level of isotropy in the embedding space. From Table 3 we see that compared to the original embeddings **ABTT**, **LAE** and **AE** all improve isotropy.

An alternative approach to verify isotropy is to check whether $Z(c)$ is a constant independent of c, which is also known as the *self-normalisation* property (Andreas and Klein, 2015). Figure 1 shows



(a) Word2Vec (b) GloVe (c) fastText

Figure 1: The histogram of $Z(c)$ on Word2Vec, GloVe and fastText for 1,000 random vectors $\boldsymbol{c}$ of unit norm. The x-axis is normalised by the mean of the values.

the histogram of $Z(c)$ of the original pre-trained embeddings, post-processed embeddings using ABTT and AE for pre-trained (a) Word2Vec, (b) GloVe and (c) fastText embeddings for a set of randomly chosen 1000 words c with unit ℓ_2 norm. Horizontal axes are normalized by the mean of the values. From Figure 1, we see that the original word embeddings in all Word2Vec, GloVe and fastText are far from being isotropic. On the other hand, AE word embeddings are isotorpic, similar to ABTT word embeddings, in all Word2Vec, GloVe and fastText. This result shows that isotropy materialises automatically during autoencoding and does not require special processing such as removing the top pricipal components as done by ABTT.

In addition to the theoretical and empirical advantages of autoencoding as a post-processing method, it is also practically attractive. For example, unlike PCA, which must be computed using the embeddings for all the words in the vocabulary, autoencoders could be run in an online fashion using only a small mini-batch of words at a time. Moreover, non-linear transformations and regularisation (e.g. in the form of dropout) can be easily incorporated into autoencoders, which can also be stacked for further post-processing. Although online (Warmuth and Kuzmin, 2007; Feng et al., 2013a; Feng et al., 2013b) and non-linear (Scholz et al., 2005) variants of PCA have been proposed, they have not been popular among practitioners due to their computational complexity, scalability and the lack of availability in deep learning frameworks.

4 Conclusion

We showed that autoencoding improves pre-trained word embeddings and outperforms the prior proposal for removing top principal components. Unlike PCA, which must be computed using the embeddings for all the words in the vocabulary, autoencoders could be run in an online fashion using only a small mini-batch of words at a time. Moreover, non-linear transformations and regularisation (e.g. in the from of dropout) can be easily incorporated into autoencoders, which can also be stacked for further post-processing. Although online (Warmuth and Kuzmin, 2007; Feng et al., 2013a; Feng et al., 2013b) and non-linear (Scholz et al., 2005) variants of PCA have been proposed, they are less attractive due to computational complexity, scalability and the lack of availability in deep learning frameworks.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

Jacob Andreas and Dan Klein. 2015. When and why are log-linear models self-normalizing? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 244–249, Denver, Colorado, May–June. Association for Computational Linguistics.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. A latent variable model ap-

proach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proc. of ICLR*.

Pierre Baldi and Kurt Hornik. 1989. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, Jan.

Cong Bao and Danushka Bollegala. 2018. Learning word meta-embeddings by autoencoding. In *Proc. of the 27th International Conference on Computational Linguistics (COLING)*, pages 1650–1661.

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *GEMS'11 Workshop on Models of Natural Language Semantics*.

Marco Baroni, Stefan Evert, and Alessandro Lenci. 2008. Esslli workshop on distributional lexical semantics bridging the gap between semantic theory and computational simulations.

H. Bourlard and Y. Kamp. 1988. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59(4-5):291–294, Sep.

E. Bruni, N. K. Tran, and M. Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47, Jan.

Minmim Chen, Zhixiang (Eddie) Xu, and Kilian Q. Weinberger. 2012. Marginalized denoising autoencoders for domain adaptation. In *Proc. of ICML*.

Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuska. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493 – 2537.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Paramveer S. Dhillon, Dean P. Foster, and Lyle H. Ungar. 2015. Eigenwords: Spectral word embeddings. *Journal of Machine Learning Research*, 16:3035–3078.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China, November. Association for Computational Linguistics.

Jiashi Feng, Huan Xu, Shie Mannor, and Shuicheng Yan. 2013a. Online pca for contaminated data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 764–772. Curran Associates, Inc.

Jiashi Feng, Huan Xu, and Shuicheng Yan. 2013b. Online robust pca via stochastic optimization. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 404–412. Curran Associates, Inc.

Gábor Gosztolya, Adam Pinter, László Tóth, Tamás Grósz, Alexandra Markó, and Tamás Csapó. 2019. Autoencoder-based articulatory-to-acoustic mapping for ultrasound silent speech interfaces. In *Proc. of IJCNN*, pages 1–8, 07.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proc. of KDD*, pages 1406–1414.

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

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *ACL'12*, pages 873 – 882.

David A. Jurgens, Saif Mohammad, Peter D. Turney, and Keith J. Holyoak. 2012. Measuring degrees of relational similarity. In *Proc. of SemEval*.

Masahiro Kaneko and Danushka Bollegala. 2019. Gender-preserving debiasing for pre-trained word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1641–1650, Florence, Italy, July. Association for Computational Linguistics.

Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. 2012. Efficient backprop. *Neural Networks: Tricks of the Trade*, pages 9–48.

Omer Levy and Yoav Goldberg. 2014. Linguistic regularities in sparse and explicit word representations. In *CoNLL*.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Tomas Mikolov, Kai Chen, and Jeffrey Dean. 2013. Efficient estimation of word representation in vector space. In *Proc. of International Conference on Learning Representations*.

Andriy Mnih and Geoffrey E. Hinton. 2009. A scalable hierarchical distributed language model. In *Proc. of NIPS*, pages 1081–1088.

Jiaqi Mu and Pramod Viswanath. 2018. All-but-the-top: Simple and effective postprocessing for word representations. In *International Conference on Learning Representations*.

Vinod Nair and Geoffrey Hinton. 2007. Rectified linear units improve restricted boltzmann machines. In *Proc. of ICML'07*.

Jeffery Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: global vectors for word representation. In *Proc. of EMNLP*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL-HLT*.

Elad Plaut. 2018. From Principal Subspaces to Principal Components with Linear Autoencoders.

Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. 2016. Variational autoencoder for deep learning of images, labels and captions. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2352–2360. Curran Associates, Inc.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: Computing word relatedness using temporal semantic analysis. In *WWW'11*, pages 337 – 346.

Vikas Raunak, Vaibhav Kumar, Vivek Gupta, and Florian Metze. 2020. On dimensional linguistic properties of the word embedding space. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 156–165, Online, July. Association for Computational Linguistics.

H. Rubenstein and J.B. Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8:627–633.

Magnus Sahlgren, Amaru Cuba Gyllensten, Fredrik Espinoza, Ola Hamfors, Jussi Karlgren, Fredrik Olsson, Per Persson, Akshay Viswanathan, and Anders Holst. 2016. The gavagai living lexicon. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 344–350, Portorož, Slovenia, May. European Language Resources Association (ELRA).

M. Scholz, F. Kaplan, C. L. Guy, J. Kopka, and J. Selbig. 2005. Non-linear pca: a missing data approach. *Bioinformatics*, 21(20):3887–3895, Aug.

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 721–732, Baltimore, Maryland, June. Association for Computational Linguistics.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Manfred K. K Warmuth and Dima Kuzmin. 2007. Randomized pca algorithms with regret bounds that are logarithmic in the dimension. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1481–1488. MIT Press.

Liangchen Wei and Zhi-Hong Deng. 2017. A variational autoencoding approach for inducing cross-lingual word embeddings. In *Proc. of IJCAI*, pages 4165–4171.

A Theoretical Proofs

The connection between linear autoencoders and principal component analysis (PCA) was first proved by (Baldi and Hornik, 1989), which provides the basis for Theorem 2. The vast applications of autoencoders such as in language (Socher et al., 2011; Silberer and Lapata, 2014), speech (Gosztolya et al., 2019) and vision (Pu et al., 2016) domains suggest that non-linear autoencoders can indeed learn better representations than PCA.

In this section, we first show that centering of pre-trained word embeddings happens automatically during the optimisation of an autoencoder. This result is stated as Lemma 1 in the paper and holds true irrespectively of the activation function used in the autoencoder, including non-linear activation functions. Next, for linear autoencoders, we recite and prove the connection between linear autoencoders and PCA in the form of Theorem 2. Finally, we discuss the approximations of the Theorem 2 for non-linear autoencoders.

Recall that we defined the autoencoder as follows:

$$\mathbf{B} = \mathbf{W}_e\mathbf{X} + \boldsymbol{b}_e\boldsymbol{u}^\top \tag{5}$$

$$\mathbf{H} = F(\mathbf{B}) \tag{6}$$

$$\mathbf{Y} = \mathbf{W}_d\mathbf{H} + \boldsymbol{b}_d\boldsymbol{u}^\top \tag{7}$$

Here, $\boldsymbol{u} \in \mathbb{R}^N$ is a vector consisting of ones and F is an element-wise activation function. The squared ℓ_2 reconstruction loss, J, for the autoencoder is given by (8).

$$J(\mathbf{W}_e, \mathbf{W}_d, \boldsymbol{b}_e, \boldsymbol{b}_d) = \left|\left|\mathbf{W}_d F(\mathbf{W}_e\mathbf{X} + \boldsymbol{b}_e\boldsymbol{u}^\top) + \boldsymbol{b}_d\boldsymbol{u}^\top\right|\right|^2 \tag{8}$$

For such an autoencoder, Lemma 1 holds.

Lemma 1. *Let $\mathbf{X}'$ and $\mathbf{H}'$ respectively denote the centred embedding and hidden state matrices. Then,* (8) *can be expressed using $\mathbf{X}'$ and $\mathbf{H}'$ as $J(\mathbf{W}_e, \mathbf{W}_d, \boldsymbol{b}_d, \hat{\boldsymbol{b}}_d) = ||\mathbf{X}' - \mathbf{W}_d\mathbf{H}'||^2$, where the optimal decoder bias is $\hat{\boldsymbol{b}}_d = \frac{1}{N}(\mathbf{X} - \mathbf{W}_d\mathbf{H})\,\boldsymbol{u}$.*

Proof. Note that the squared ℓ_2 reconstruction error can be written as in (9).

$$J(\mathbf{W}_e, \mathbf{W}_d, \boldsymbol{b}_e, \boldsymbol{b}_d) = ||\mathbf{X} - \mathbf{Y}||^2 \tag{9}$$

Substituting for $\mathbf{Y}$ from (7) in (9) we have

$$J(\mathbf{W}_e, \mathbf{W}_d, \boldsymbol{b}_e, \boldsymbol{b}_d) = \left|\left|\mathbf{X} - \mathbf{W}_d\mathbf{H} - \boldsymbol{b}_d\boldsymbol{u}^\top\right|\right|^2 \tag{10}$$

$$= \mathrm{tr}\left((\mathbf{X} - \mathbf{W}_d\mathbf{H} - \boldsymbol{b}_d\boldsymbol{u}^\top)^\top(\mathbf{X} - \mathbf{W}_d\mathbf{H} - \boldsymbol{b}_d\boldsymbol{u}^\top)\right) \tag{11}$$

From the definition, $\boldsymbol{u} \in \mathbb{R}^N$ is a vector with all elements set to 1, where N is the total number of words in the vocabulary $\mathcal{V}$ for which we are given pre-trained embeddings, arranged as columns in $\mathbf{X} \in \mathbb{R}^{n\times N}$. The minimiser of J, w.r.t. $\boldsymbol{b}_d$, $\hat{\boldsymbol{b}}_d$ satisfies $\frac{\partial J}{\partial \boldsymbol{b}_d} = \mathbf{0}$, and is given by (13).

$$\left(\mathbf{X} - \mathbf{W}_d\mathbf{H} - \boldsymbol{b}_d\boldsymbol{u}^\top\right)\boldsymbol{u} = \mathbf{0} \tag{12}$$

$$\hat{\boldsymbol{b}}_d = \frac{1}{N}(\mathbf{X} - \mathbf{W}_d\mathbf{H})\,\boldsymbol{u} \tag{13}$$

In (13) we used $\boldsymbol{u}^\top \boldsymbol{u} = N$. Substituting this minimiser $\hat{\boldsymbol{b}}_d$ back in (10) we obtain the following.

$$
\begin{aligned}
& J(\mathbf{W}_e, \mathbf{W}_d, \boldsymbol{b}_e, \hat{\boldsymbol{b}}_d) \\
&= \left|\left| \mathbf{X} - \mathbf{W}_d\mathbf{H} - \frac{1}{N}\left(\mathbf{X} - \mathbf{W}_d\mathbf{H}\right)\boldsymbol{u}\boldsymbol{u}^\top \right|\right|^2 && (14)\\
&= \left|\left| \left(\mathbf{X} - \frac{1}{N}\mathbf{X}\boldsymbol{u}\boldsymbol{u}^\top\right) - \mathbf{W}_d\left(\mathbf{H} - \frac{1}{N}\mathbf{H}\boldsymbol{u}\boldsymbol{u}^\top\right) \right|\right|^2 && (15)\\
&= \left|\left| \left(\mathbf{X} - \boldsymbol{\mu}_X\boldsymbol{u}^\top\right) - \mathbf{W}_d\left(\mathbf{H} - \boldsymbol{\mu}_H\boldsymbol{u}^\top\right) \right|\right|^2 && (16)\\
&= \left|\left| \mathbf{X}' - \mathbf{W}_d\mathbf{H}' \right|\right|^2 && (17)
\end{aligned}
$$

In (16) we use the mean vectors of embeddings, $\boldsymbol{\mu}_X$, and hidden states $\boldsymbol{\mu}_H$ given respectively by (18) and (19).

$$\boldsymbol{\mu}_X = \frac{1}{N}\mathbf{X}\boldsymbol{u} \tag{18}$$

$$\boldsymbol{\mu}_H = \frac{1}{N}\mathbf{H}\boldsymbol{u} \tag{19}$$

Moreover, we defined the mean-subtracted (i.e. *centred*) versions of $\mathbf{X}$ and $\mathbf{H}$ in (17) respectively by $\mathbf{X}'$ and $\mathbf{H}'$ defined as follows.

$$\mathbf{X}' = \mathbf{X} - \boldsymbol{\mu}_X\boldsymbol{u}^\top \tag{20}$$

$$\mathbf{H}' = \mathbf{H} - \boldsymbol{\mu}_H\boldsymbol{u}^\top \tag{21}$$

□

Using Lemma 1, we can replace the embedding matrix, $\mathbf{X}$, by its pre-centred version and further drop the biases as they can be absorbed into the encoder/decoder weight matrices by introducing a dimension set to 1 in the input and output embeddings. Theorem 2 holds under these transformations.

Theorem 2. *Assume that $\boldsymbol{\Sigma}_{xx}$ is full-rank with n distinct eigenvalues $\lambda_1 > \ldots > \lambda_n$. Let $\mathcal{I} = \{i_1, \ldots, i_p\}$ $(1 \leq i_1 < \ldots < i_p \leq n)$ is any ordered p-index set, and $\mathbf{U}_\mathcal{I} = [\boldsymbol{u}_{i_1}, \ldots \boldsymbol{u}_{i_p}]$ denote the matrix formed by the orthogonal eigenvectors of $\boldsymbol{\Sigma}_{xx}$ associated with the eigenvalues $\lambda_{i_1}, \ldots, \lambda_{i_p}$. Then two full-rank matrices $\mathbf{W}_d$ and $\mathbf{W}_e$ define a critical point of* (8) *for a linear autoencoder if and only if there exists an ordered p-index set $\mathcal{I}$ and an invertible matrix $\mathbf{C} \in \mathbb{R}^{p\times p}$ such that*

$$\mathbf{W}_d = \mathbf{U}_\mathcal{I}\mathbf{C}, \tag{22}$$

$$\mathbf{W}_e = \mathbf{C}^{-1}\mathbf{U}_\mathcal{I}^{-1}. \tag{23}$$

Moreover, the reconstruction error, $J(\mathbf{W}_e, \mathbf{W}_d)$ can be expressed as

$$J(\mathbf{W}_e, \mathbf{W}_d) = \operatorname{tr}(\boldsymbol{\Sigma}_{xx}) - \sum_{t\in\mathcal{I}} \lambda_t. \tag{24}$$

Proof. The squared ℓ_2 reconstruction error can be written by dropping the bias terms and using the centred word embedding matrix $\mathbf{X}$ as in (26).

$$
\begin{aligned}
& J(\mathbf{W}_e, \mathbf{W}_d) \\
&= \operatorname{tr}\left(\left(\mathbf{X} - \mathbf{W}_d\mathbf{W}_e\mathbf{X}\right)^\top \left(\mathbf{X} - \mathbf{W}_d\mathbf{W}_e\mathbf{X}\right)\right) && (25)\\
&= \operatorname{tr}(\mathbf{X}\mathbf{X}^\top) - 2\operatorname{tr}(\mathbf{X}\mathbf{X}^\top\mathbf{W}_d\mathbf{W}_e) \\
&\quad + \operatorname{tr}(\mathbf{X}^\top{\mathbf{W}_e}^\top{\mathbf{W}_d}^\top\mathbf{W}_d\mathbf{W}_e\mathbf{X}) && (26)
\end{aligned}
$$

When $\mathbf{W}_e$ and $\mathbf{W}_d$ are critical points, by setting $\frac{\partial J}{\partial \mathbf{W}_e} = \mathbf{0}$ we obtain

$$-2{\mathbf{W}_d}^\top \mathbf{X}\mathbf{X}^\top + 2{\mathbf{W}_d}^\top \mathbf{W}_d \mathbf{W}_e \mathbf{X}\mathbf{X}^\top = \mathbf{0} \tag{27}$$

$$({\mathbf{W}_d}^\top - {\mathbf{W}_d}^\top \mathbf{W}_d \mathbf{W}_e)\mathbf{X}\mathbf{X}^\top = \mathbf{0} \tag{28}$$

Because the covariance matrix for the pre-trained embeddings, $\mathbf{\Sigma}_{xx} = \mathbf{X}\mathbf{X}^\top$ is full-rank and is thus invertible, $\mathbf{W}_e$ is given by (23).

$$\mathbf{W}_e = ({\mathbf{W}_d}^\top \mathbf{W}_d)^{-1} {\mathbf{W}_d}^\top \tag{29}$$

Likewise, from $\frac{\partial J}{\partial \mathbf{W}_d} = \mathbf{0}$ we obtain

$$\mathbf{0} = -2\mathbf{X}\mathbf{X}^\top {\mathbf{W}_e}^\top + 2\mathbf{W}_d(\mathbf{W}_e\mathbf{X})(\mathbf{W}_e\mathbf{X})^\top \tag{30}$$

$$\mathbf{W}_d = \mathbf{X}\mathbf{X}^\top {\mathbf{W}_e}^\top (\mathbf{W}_e \mathbf{X}\mathbf{X}^\top {\mathbf{W}_e}^\top)^{-1} \tag{31}$$

$$\mathbf{W}_d = \mathbf{\Sigma}_{xx} {\mathbf{W}_e}^\top (\mathbf{W}_e \mathbf{\Sigma}_{xx} {\mathbf{W}_e}^\top)^{-1} \tag{32}$$

We will first show that (22) and (23) satisfy respectively (32) and (29), thereby proving the necessary condition. Specifically, from (29) and (22) we have the following.

$$\begin{aligned}
&({\mathbf{W}_d}^\top \mathbf{W}_d)^{-1} {\mathbf{W}_d}^\top \\
&= \left((\mathbf{U}_\mathcal{I}\mathbf{C})^\top (\mathbf{U}_\mathcal{I}\mathbf{C})\right)^{-1} (\mathbf{U}_\mathcal{I}\mathbf{C})^\top && (33)\\
&= \left(\mathbf{C}^\top {\mathbf{U}_\mathcal{I}}^\top \mathbf{U}_\mathcal{I}\mathbf{C}\right)^{-1} \mathbf{C}^\top {\mathbf{U}_\mathcal{I}}^\top && (34)\\
&= \mathbf{C}^{-1}(\mathbf{C}^\top)^{-1}\mathbf{C}^\top \mathbf{U}_\mathcal{I}^{-1} && (35)\\
&= \mathbf{C}^{-1}\mathbf{U}_\mathcal{I} = \mathbf{W}_e && (36)
\end{aligned}$$

In (34), from the orthogonality of $\mathbf{U}_\mathcal{I}$, we used ${\mathbf{U}_\mathcal{I}}^\top \mathbf{U}_\mathcal{I} = \mathbf{I}$, where $\mathbf{I} \in \mathbb{R}^{p\times p}$ is the identity matrix.

Likewise, from (32) and (23) we have the following.

$$\begin{aligned}
&\mathbf{\Sigma}_{xx} {\mathbf{W}_e}^\top (\mathbf{W}_e \mathbf{\Sigma}_{xx} {\mathbf{W}_e}^\top)^{-1} \\
&= \mathbf{\Sigma}_{xx} \left(\mathbf{C}^{-1}\mathbf{U}_\mathcal{I}^{-1}\right)^\top \left(\mathbf{C}^{-1}\mathbf{U}_\mathcal{I}^{-1}\mathbf{\Sigma}_{xx}(\mathbf{U}_\mathcal{I}^{-1})^\top (\mathbf{C}^{-1})^\top\right)^{-1} && (37)\\
&= \mathbf{\Sigma}_{xx}({\mathbf{U}_\mathcal{I}}^\top)^{-1}(\mathbf{C}^\top)^{-1}\mathbf{C}^\top {\mathbf{U}_\mathcal{I}}^\top \mathbf{\Sigma}_{xx}^{-1}\mathbf{U}_\mathcal{I}\mathbf{C} && (38)\\
&= \mathbf{U}_\mathcal{I}\mathbf{C} = \mathbf{W}_d && (39)
\end{aligned}$$

This completes the proof for the necessary condition.

Next, let us look at the sufficient condition. For this purpose, we define for a matrix $\mathbf{M} \in \mathbb{R}^{n\times p}$ $(p \ll n)$ the orthogonal projection onto the subspace spanned by the columns of $\mathbf{M}$ by $\mathbf{P}_\mathbf{M}$ given by (40).

$$\mathbf{P}_\mathbf{M} \triangleq \mathbf{M}(\mathbf{M}^\top \mathbf{M})^{-1}\mathbf{M}^\top \tag{40}$$

Therefore, for an orthogonal matrix $\mathbf{U}$, we can evaluate $\mathbf{P}_{\mathbf{U}^\top \mathbf{W}_d}$ as follows.

$$\begin{aligned}
&\mathbf{P}_{\mathbf{U}^\top \mathbf{W}_d} \\
&= \mathbf{U}^\top \mathbf{W}_d \left((\mathbf{U}^\top \mathbf{W}_d)^\top (\mathbf{U}^\top \mathbf{W}_d)\right)^{-1} \left(\mathbf{U}^\top \mathbf{W}_d\right)^\top && (41)\\
&= \mathbf{U}^\top \mathbf{W}_d \left({\mathbf{W}_d}^\top \mathbf{U}\mathbf{U}^\top \mathbf{W}_d\right)^{-1} {\mathbf{W}_d}^\top \mathbf{U} && (42)\\
&= \mathbf{U}^\top \mathbf{W}_d \left({\mathbf{W}_d}^\top \mathbf{W}_d\right)^{-1} {\mathbf{W}_d}^\top \mathbf{U} && (43)\\
&= \mathbf{U}^\top \mathbf{P}_{\mathbf{W}_d} \mathbf{U} && (44)
\end{aligned}$$

In (44), from the definition in (40) we used $\mathbf{P}_{\mathbf{W}_d} = \mathbf{W}_d \left(\mathbf{W}_d{}^\top \mathbf{W}_d\right)^{-1} \mathbf{W}_d{}^\top$. From (44) we can write $\mathbf{P}_{\mathbf{W}_d}$ as given in (45).

$$\mathbf{P}_{\mathbf{W}_d} = \mathbf{U}\mathbf{P}_{\mathbf{U}^\top \mathbf{W}_d}\mathbf{U}^\top \tag{45}$$

On the other hand, from (32) we have

$$\mathbf{W}_d = \mathbf{\Sigma}_{xx}\mathbf{W}_e{}^\top(\mathbf{W}_e\mathbf{\Sigma}_{xx}\mathbf{W}_e{}^\top)^{-1}, \tag{46}$$

$$\mathbf{W}_d \left(\mathbf{W}_e\mathbf{\Sigma}_{xx}\mathbf{W}_e{}^\top\right) = \mathbf{\Sigma}_{xx}\mathbf{W}_e{}^\top. \tag{47}$$

Right multiplying both sides in (47) by $\mathbf{W}_d{}^\top$ we arrive at

$$\mathbf{W}_d\mathbf{W}_e\mathbf{\Sigma}_{xx}\mathbf{W}_e{}^\top\mathbf{W}_d{}^\top = \mathbf{\Sigma}_{xx}\left(\mathbf{W}_d\mathbf{W}_e\right)^\top. \tag{48}$$

However, by left multiplying (29) by $\mathbf{W}_d$ we get

$$\mathbf{W}_d\mathbf{W}_e = \mathbf{W}_d(\mathbf{W}_d{}^\top\mathbf{W}_d)^{-1}\mathbf{W}_d{}^\top = \mathbf{P}_{\mathbf{W}_d}. \tag{49}$$

Substituting for $\mathbf{W}_d\mathbf{W}_e$ from (49) back in (48) we obtain

$$\mathbf{P}_{\mathbf{W}_d}\mathbf{\Sigma}_{xx}\mathbf{P}_{\mathbf{W}_d}{}^\top = \mathbf{\Sigma}_{xx}\mathbf{P}_{\mathbf{W}_d}{}^\top. \tag{50}$$

Note that by the definition in (40), $\mathbf{P}_{\mathbf{W}_d}$ is symmetric (i.e. $\mathbf{P}_{\mathbf{W}_d}{}^\top = \mathbf{P}_{\mathbf{W}_d}$). Therefore, (50) can be further simplified as given by (51).

$$\mathbf{P}_{\mathbf{W}_d}\mathbf{\Sigma}_{xx}\mathbf{P}_{\mathbf{W}_d} = \mathbf{\Sigma}_{xx}\mathbf{P}_{\mathbf{W}_d} \tag{51}$$

Taking the transpose of both sides in (51) we can further show that

$$\left(\mathbf{P}_{\mathbf{W}_d}\mathbf{\Sigma}_{xx}\mathbf{P}_{\mathbf{W}_d}\right)^\top = \left(\mathbf{\Sigma}_{xx}\mathbf{P}_{\mathbf{W}_d}\right)^\top \tag{52}$$

$$\mathbf{P}_{\mathbf{W}_d}\mathbf{\Sigma}_{xx}\mathbf{P}_{\mathbf{W}_d} = \mathbf{\Sigma}_{xx} \tag{53}$$

From (51) and (53) we can deduce that

$$\mathbf{P}_{\mathbf{W}_d}\mathbf{\Sigma}_{xx} = \mathbf{\Sigma}_{xx}\mathbf{P}_{\mathbf{W}_d} = \mathbf{P}_{\mathbf{W}_d}\mathbf{\Sigma}_{xx}\mathbf{P}_{\mathbf{W}_d} \tag{54}$$

Because $\mathbf{\Sigma}_{xx}$ is real and symmetric, it can be diagonalised using an orthogonal matrix $\mathbf{U}$, containing the eigenvectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ of $\mathbf{\Sigma}_{xx}$ in columns, corresponding to the eigenvalues $\lambda_1, \ldots, \lambda_n$. Specifically, we can write this as in (55).

$$\mathbf{\Sigma}_{xx} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top \tag{55}$$

Here, $\mathbf{\Lambda}$ is a diagonal matrix with non-increasing eigenvalues $\lambda_1, \ldots, \lambda_n$.

Let us substitute for $\mathbf{P}_{\mathbf{W}_d}$ from (45) and for $\mathbf{\Sigma}_{xx}$ from (55) in (54).

$$\mathbf{P}_{\mathbf{W}_d}\mathbf{\Sigma}_{xx} = \mathbf{\Sigma}_{xx}\mathbf{P}_{\mathbf{W}_d} \tag{56}$$

$$\mathbf{U}\mathbf{P}_{\mathbf{U}^\top\mathbf{W}_d}\mathbf{U}^\top\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top\mathbf{U}\mathbf{P}_{\mathbf{U}^\top\mathbf{W}_d}\mathbf{U}^\top \tag{57}$$

$$\mathbf{U}\mathbf{P}_{\mathbf{U}^\top\mathbf{W}_d}\mathbf{\Lambda}\mathbf{U}^\top = \mathbf{U}\mathbf{\Lambda}\mathbf{P}_{\mathbf{U}^\top\mathbf{W}_d}\mathbf{U}^\top \tag{58}$$

$$\mathbf{P}_{\mathbf{U}^\top\mathbf{W}_d}\mathbf{\Lambda} = \mathbf{\Lambda}\mathbf{P}_{\mathbf{U}^\top\mathbf{W}_d} \tag{59}$$

Because $\lambda_1 > \ldots > \lambda_n$, $\mathbf{P}_{\mathbf{U}^\top\mathbf{W}_d}$ must be a diagonal matrix to satisfy (59). Specifically, $\mathbf{P}_{\mathbf{U}^\top\mathbf{W}_d}$ contains 1 as an eigenvalue (p times) and 0 ($n - p$ times), and can be written using a diagonal matrix $\mathbf{I}_{\mathcal{I}}$ as follows.

$$\mathbf{P}_{\mathbf{U}^\top\mathbf{W}_d} = \mathbf{I}_{\mathcal{I}} \tag{60}$$

where the (i,i) diagonal element of $\mathbf{I}_{\mathcal{I}}$ is given by (61).

$$(\mathbf{I}_{\mathcal{I}})_{(i,i)} = \begin{cases} 1 & \text{if } i \in \mathcal{I} \\ 0 & \text{otherwise} \end{cases} \tag{61}$$

Therefore, there exists a unique index set $\mathcal{I} = \{i_1, \ldots, i_p\}$ with $(1 \leq i_1 < \ldots < i_p \leq n)$ such that $\mathbf{P}_{\mathbf{U}^\top \mathbf{W}_d}$ is a diagonal matrix as given by (60).

From (45) and (49) we can write,

$$\begin{aligned} \mathbf{P}_{\mathbf{W}_d} &= \mathbf{U}\mathbf{P}_{\mathbf{U}^\top \mathbf{W}_d}\mathbf{U}^\top \\ &= \mathbf{U}\mathbf{I}_{\mathcal{I}}\mathbf{U}^\top = \mathbf{U}_{\mathcal{I}}{\mathbf{U}_{\mathcal{I}}}^\top = \mathbf{W}_d\mathbf{W}_e, \end{aligned} \tag{62}$$

where $\mathbf{U}_{\mathcal{I}} = [\boldsymbol{u}_{i_1}, \ldots \boldsymbol{u}_{i_p}]$. Therefore, $\mathbf{P}_{\mathbf{W}_d}$ is the orthogonal projection onto the subspace spanned by the columns of $\mathbf{U}_{\mathcal{I}}$. Since the column space of $\mathbf{W}_d$ coincides with the column space of $\mathbf{U}_{\mathcal{I}}$, there exists an invertible $\mathbf{C} \in \mathbb{R}^{p \times p}$ matrix such that

$$\mathbf{W}_d = \mathbf{U}_{\mathcal{I}}\mathbf{C}, \tag{63}$$

$$\mathbf{W}_e = \mathbf{C}^{-1}{\mathbf{U}_{\mathcal{I}}}^{-1}. \tag{64}$$

Therefore, the parameters (i.e. encoder and decoder matrices) of the autoencoder is uniquely determined only upto the scaling matrix $\mathbf{C}$.

Next, we will consider the reconstruction loss. First, let us substitute $\mathbf{\Sigma}_{xx}$ in (26) and rearrange the terms inside the traces as follows.

$$\begin{aligned} J(\mathbf{W}_e, \mathbf{W}_d) = \operatorname{tr}(\mathbf{\Sigma}_{xx}) - 2\operatorname{tr}(\mathbf{\Sigma}_{xx}\mathbf{W}_d\mathbf{W}_e) \\ + \operatorname{tr}(\mathbf{\Sigma}_{xx}{\mathbf{W}_e}^\top{\mathbf{W}_d}^\top\mathbf{W}_d\mathbf{W}_e) \end{aligned} \tag{65}$$

In (65), we used $\operatorname{tr}(\mathbf{ABC}) = \operatorname{tr}(\mathbf{CAB}) = \operatorname{tr}(\mathbf{BCA})$ when the product of the three matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are suitably defined.

Substituting for the product $\mathbf{W}_d\mathbf{W}_e$ from (62) in (65) we obtain,

$$\begin{aligned} J(\mathbf{W}_e, \mathbf{W}_d) = \operatorname{tr}(\mathbf{\Sigma}_{xx}) - 2\operatorname{tr}(\mathbf{\Sigma}_{xx}\mathbf{P}_{\mathbf{W}_d}) \\ + \operatorname{tr}(\mathbf{\Sigma}_{xx}{\mathbf{P}_{\mathbf{W}_d}}^\top\mathbf{P}_{\mathbf{W}_d}). \end{aligned} \tag{66}$$

From the definition in (40), we see that $\mathbf{P}_{\mathbf{W}_d}$ is symmetric (hence, ${\mathbf{P}_{\mathbf{W}_d}}^\top = \mathbf{P}_{\mathbf{W}_d}$) and moreover that $\mathbf{P}_{\mathbf{W}_d}\mathbf{P}_{\mathbf{W}_d} = \mathbf{P}_{\mathbf{W}_d}$. Using this fact in (66) we can rewrite,

$$J(\mathbf{W}_e, \mathbf{W}_d) = \operatorname{tr}(\mathbf{\Sigma}_{xx}) - \operatorname{tr}(\mathbf{\Sigma}_{xx}\mathbf{P}_{\mathbf{W}_d}). \tag{67}$$

Substituting (55) and (45) in (67) we obtain,

$$J(\mathbf{W}_e, \mathbf{W}_d) = \operatorname{tr}(\mathbf{\Sigma}_{xx}) - \operatorname{tr}(\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top\mathbf{U}\mathbf{P}_{\mathbf{U}^\top\mathbf{W}_d}\mathbf{U}^\top) \tag{68}$$

$$= \operatorname{tr}(\mathbf{\Sigma}_{xx}) - \operatorname{tr}(\mathbf{U}\mathbf{\Lambda}\mathbf{P}_{\mathbf{U}^\top\mathbf{W}_d}\mathbf{U}^\top) \tag{69}$$

$$= \operatorname{tr}(\mathbf{\Sigma}_{xx}) - \operatorname{tr}(\mathbf{\Lambda}\mathbf{P}_{\mathbf{U}^\top\mathbf{W}_d}\mathbf{U}^\top\mathbf{U}) \tag{70}$$

$$= \operatorname{tr}(\mathbf{\Sigma}_{xx}) - \operatorname{tr}(\mathbf{\Lambda}\mathbf{P}_{\mathbf{U}^\top\mathbf{W}_d}) \tag{71}$$

$$= \operatorname{tr}(\mathbf{\Sigma}_{xx}) - \operatorname{tr}(\mathbf{\Lambda}\mathbf{I}_{\mathcal{I}}) \tag{72}$$

$$= \operatorname{tr}(\mathbf{\Sigma}_{xx}) - \sum_{t \in \mathcal{I}} \lambda_t \tag{73}$$

In (72) above we used (60). For a given set of word embeddings, $\mathbf{\Sigma}_{xx}$ is fixed. Therefore, to minimise $J(\mathbf{W}_e, \mathbf{W}_d)$ we must select the largest eigenvalues as λ_t (and their corresponding eigenvectors). This completes the proof of Theorem 2. □

Embedding	Word2Vec				GloVe				fastText			
Dataset	original	150d	300d	600d	original	150d	300d	600d	original	150d	300d	600d
WS-353	**62.4**	**62.4**	61.8	61.7	60.6	47.8	65.8	**66.9**	65.9	68.2	**69.0**	68.7
SIMLEX-999	44.7	40.8	45.5	**45.8**	39.5	30.8	42.2	**43.2**	46.2	43.3	48.8	**49.0**
RG-65	75.4	75.6	**76.3**	75.9	68.1	68.7	**72.3**	72.2	78.4	77.6	**80.5**	80.4
MTurk-287	69.0	**70.0**	68.9	68.5	71.8	56.8	74.4	**74.9**	73.3	**75.2**	74.7	74.8
MTurk-771	63.1	62.9	**63.9**	63.8	62.7	51.5	67.7	**68.3**	69.6	70.5	**72.4**	72.3
MEN	68.1	67.2	**69.3**	69.2	67.7	59.8	74.8	**75.4**	71.1	74.8	76.0	**76.1**
MSR	**73.6**	61.9	73.4	73.6	73.8	60.2	74.4	**74.6**	87.1	83.4	**87.3**	**87.3**
Google	74.0	68.1	74.3	**74.4**	76.8	69.6	77.1	**77.2**	85.3	83.4	**86.4**	86.3
SemEval	20.0	16.4	20.3	19.7	15.4	13.5	**17.6**	17.3	21.0	21.4	**23.3**	23.0
BLESS	**70.5**	65.0	70.0	68.5	76.5	74.0	**79.5**	78.5	75.5	80.0	**80.5**	80.0
ESSLI	75.5	**76.2**	**76.2**	74.5	72.2	56.7	**73.0**	**73.0**	74.7	72.4	**77.0**	76.9

Table 4: The autoencoder results using 150, 300 and 600 dimensions for the hidden layer in contrast to original word embeddings.

A.1 Linear approximations to non-linear activation functions

The autoencoder considered in Theorem 2 is linear in the sense that the elementwise activation function is assumed to be $\mathbf{H} = F(\mathbf{B}) = \mathbf{B}$. However, in practice autoencoders are used with nonlinear activation units such as rectified linear units ReLU; Nair and Hinton (2007), hyperbolic tangent ($\tanh$) and sigmoid (σ) functions (LeCun et al., 2012). Exact analysis of Theorem 2 in the general case is is complicated due to the non-linearity of the activation functions. Therefore, instead, we consider first-order linear approximations for the above-mentioned non-linear activation functions.

ReLU is a piece-wise linear function as given by (74).

$$F_{relu}(x) = \begin{cases} x & x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{74}$$

Therefore, when ReLU is in its active region, it can be seen as a linear unit.

For $\tanh$ and σ, when x is small, we use the first-order Taylor expansion to obtain linear approximations as follows.

$$F_{\tanh}(x) = \frac{1}{1+\exp(-x)} \approx \frac{1}{2} + \frac{1}{4}x \tag{75}$$

$$F_{\sigma}(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \approx 0 + x \tag{76}$$

Therefore, in their linear regions close to zero, both $\tanh$ and σ behave like linear functions.

As empirically investigated in the main paper, the performance difference in the word embeddings post-processed using linear vs. non-linear autoencoders is statistically insignificant. Considering the theoretical equivalence between PCA and linear autoencoders, this result shows that it is more important to perform centering and apply PCA rather than using a non-linear activation in the hidden layer of the autoencoder.

B Experimental Settings and Additional Results

We use semantic similarity (**WS-353**; Agirre et al. (2009), **SIMLEX-999**; Hill et al. (2015), **RG-65**; Rubenstein and Goodenough (1965), **MTurk-287**; Radinsky et al. (2011), **MTurk-771**; Halawi et al. (2012) and **MEN**; Bruni et al. (2014)), analogy (**Google**, **MSR** (Mikolov et al., 2013), and **SemEval**; Jurgens et al. (2012)) and concept categorisation **BLESS**; Baroni and Lenci (2011) and **ESSLI**; Baroni et al. (2008)) benchmark datasets that were already mentioned in the main article for additional experiments. Experiments are conducted using the same 300 dimensional pre-trained embedding learnt using Word2Vec, GloVe and fastText as described in the main body of the paper.

To evaluate the effect of the dimensionality of the hidden layer in the autoencoder on the performance of the post-processed embeddings, in Table 4 we train autoencoders with hidden layer dimensionalities of 150, 300 and 600 and compare the performance against the original (non-post-processed) word embeddings. For the pre-trained embeddings using Word2Vec and fastText, we see that setting the hidden layer's dimensionality to 300, which is equal to the dimensionality of the input word embeddings, produces better results than with 150 or 600 dimensions in the majority of the datasets. On the other hand, for pre-trained GloVe embeddings we see that overall the performance increases with the dimensionality of the hidden layer, and the best performance is reported with a 600 dimensional hidden layer in the majority of the datasets.