Evaluating Unsupervised Dimensionality Reduction Methods for Pretrained Sentence Embeddings

Gaifan Zhang[1], **Yi Zhou**[2], **Danushka Bollegala**[3]
Columbia University[1], Cardiff University[2], University of Liverpool[3]
zgaifan@gmail.com, zhouy131@cardiff.ac.uk, danushka@liverpool.ac.uk

Abstract

Sentence embeddings produced by Pretrained Language Models (PLMs) have received wide attention from the NLP community due to their superior performance when representing texts in numerous downstream applications. However, the high dimensionality of the sentence embeddings produced by PLMs is problematic when representing large numbers of sentences in memory- or compute-constrained devices. As a solution, we evaluate unsupervised dimensionality reduction methods to reduce the dimensionality of sentence embeddings produced by PLMs. Our experimental results show that simple methods such as Principal Component Analysis (PCA) can reduce the dimensionality of sentence embeddings by almost 50%, without incurring a significant loss in performance in multiple downstream tasks. Surprisingly, reducing the dimensionality further *improves* performance over the original high dimensional versions for the sentence embeddings produced by some PLMs in some tasks.

Keywords: Dimensionality Reduction, Sentence Embeddings, Pre-trained Sentence Encoders

1. Introduction

Sentence embedding models represent a given input sentence using a fixed dimensional vector, which is independent of the length of the input sentence (Reimers and Gurevych, 2019a; Gao et al., 2021a). Sentence embeddings have significantly improved performance in numerous downstream NLP tasks such as information retrieval (Kong et al., 2022; Palangi et al., 2016), question answering (Hao et al., 2019), and machine translation (Wang et al., 2017) to name a few (Choi et al., 2021). However, compared to static word embeddings (Pennington et al., 2014; Mikolov et al., 2013), which typically have a smaller number of dimensions (ca. 50-300), sentence embeddings produced by PLMs are usually high dimensional (ca. 1024-4096). This is problematic due to several reasons as described next.

First, storing pre-computed sentence embeddings requires larger memory/disk space. For example, in dense retrieval systems (Kong et al., 2022; Nigam et al., 2019), documents must be pre-embedded and stored in order to efficiently process queries at retrieval time. Therefore, the storage requirements increase linearly with the number of documents in the collection. Although it is efficient to store all the embeddings in memory to reduce the search latency as opposed to reading from a disk-based storage, this is not possible due to the high dimensionality of the sentence/document embeddings. Moreover, training deep learning models with high-dimensional sentence embeddings is problematic because GPUs have limited memory buffers. This requires carefully selecting training data batches and results in latency overheads when transferring sentence embeddings back-and-forth between GPUs.

Second, the computation time of the inner-products between two sentence embeddings increases linearly with the dimensionality of the embedding. For example, in dense retrieval, we must compute the inner-product (or equivalently cosine similarity if the embeddings are ℓ_2 normalised) between the query embedding and each of the document embeddings to rank the documents that are relevant to a query (Menon et al., 2022; Nigam et al., 2019). This is prohibitively expensive when a large number of documents must be ranked within millisecond order retrieval times.

Given this trade-off between the dimensionality and the accuracy of sentence embeddings, in this paper, we consider the following question: **Can we reduce the dimensionality of pre-computed sentence embeddings without significantly sacrificing the performance in downstream tasks that use those *dimensionality-reduced* sentence embeddings?** Although a diverse set of dimensionality reduction methods have been proposed, from a real-world large-scale application perspective, dimensionality reduction methods that satisfy the following properties are desirable:
(a) In contrast to training lower-dimensional sentence embeddings from scratch or training lower-dimensional student models via distillation (Anil et al., 2018), we would prefer dimensionality reduction methods that are *computationally lightweight* such that they can be applied in a *post-processing* stage to reduce the dimensionality of pre-computed sentence embeddings. Distillation-based methods

Work done while the first author was a student at the University of Liverpool.

LREC-COLING 2024, pages 6530–6543
20-25 May, 2024.

require additional training data for training the student model, as well as a larger memory to load the teacher model. Moreover, the inference time required by the student model, which will be used after distillation must also be taken into account. On the other hand, dimensionality reduction methods are particularly desirable when we have a large and continuously growing number of sentences as in the case of dense information retrieval.
(b) We would prefer *unsupervised* dimensionality reduction methods over supervised ones such that no labelled data are required for a specific downstream task (Zhao et al., 2022). Such labelled instances might not be available in specialised domains and in larger quantities. This helps us to produce dimensionality-reduced sentence embeddings that are independent of a particular task, thus more likely to generalise well to multiple different tasks.

We present a novel analysis of unsupervised dimensionality reduction methods for sentence embeddings: truncated Singular Value Decomposition (**SVD**; Petersen and Petersen, 2012), Principal Component Analysis (**PCA**; Andrews, 2016), Kernel PCA (**KPCA**; Schölkopf et al., 1998), Gaussian Random Projections (**GRP**; Bingham and Mannila, 2001) and Autoencoders (Vincent et al., 2008). Lower-dimensional projections can be learnt in an *inductive* setting (uses only the *train* sentences to learn the projection), or in a *transductive* setting (uses unlabelled *test* sentences in addition to the train sentences). The inductive setting is desirable in scenarios where we have a continuous stream of test sentences such as in dense retrieval, whereas the transductive setting is sufficient when the test sentences are fixed and known in advance during projection learning time.

We use six popular sentence encoders in three tasks: semantic textual similarity (STS) measurement (Cer et al., 2017), entailment prediction (Marelli et al., 2014), and TREC question-type classification (Hovy et al., 2001). In particular, semantic textual similarity is a popular NLP task that is frequently used to measure the accuracy of sentence embeddings (Cer et al., 2017). The ability to recognise textual entailment is also considered a fundamental task for evaluating natural language understanding (Dzikovska et al., 2013; Yokote et al., 2012). Moreover, question classification is a sentence classification task, which requires an accurate sentence representation to obtain good accuracy (Li and Roth, 2002). Overall, PCA proves to be the most effective method for sentence embedding compression. Previous research has demonstrated the effectiveness of PCA for various compression tasks (Raunak et al., 2019; Reddy et al., 2020), but we are the first to conduct a systematic study specifically for sentence embedding compression.

Our experimental results show that in both transductive as well as inductive settings, we can use PCA to reduce the dimensionality of diverse sentence embeddings by ca. 50% within a 1% loss in performance in the target tasks. Interestingly, reducing dimensionality *improves* accuracy for some sentence encoders such as *all-mpnet-base-v2* for STS and *msmarco-roberta-base-v2* for TREC.

2. Related Work

Neural Network Compression: The majority of work for compression focuses on learning neural network models with fewer parameters. Different techniques have been proposed for this purpose such as pruning (neuron/layer/weight dropout) (Han et al., 2015; Li et al., 2017; Lee et al., 2019), quantization (Han et al., 2016; Chen et al., 2016), distillation (Jiao et al., 2020; Sanh et al., 2019), etc. However, in this paper we *do not* consider the problem of compressing PLMs, but focus only on the dimensionality reduction of the sentence embeddings produced by a given sentence encoder.

Word Embedding Compression: Several prior work has considered the problem of compressing pre-trained static word embeddings. Andrews (2016) adds sparsity and non-negative encoding to a specific autoencoder scheme, thereby compressing original dense vector embeddings. Kim et al. (2020) propose an adaptive compression method that uses code-book to represent words as discrete codes of different lengths. To reduce the number of parameters, Shu and Nakayama (2018) assign a small set of basis vectors to each word, with the storage efficiency maximised by a composition coding approach. However, our focus in this paper is sentence embeddings and *not* word embeddings.

Sentence Embedding Compression: Compared to model/word embedding compression work described above, lower-dimensional sentence embeddings are understudied. Zhao et al. (2022) proposed homomorphic projective distillation to compress sentence embeddings using labelled training instances for textual entailment to learn a projection layer for a transformer-based encoder. In contrast, the dimensionality reduction methods that we evaluate are all *unsupervised*, thus not requiring labelled data. Moreover, the methods we consider do not require student network training, which can be computationally expensive.

3. Dimensionality Reduction Methods

Let us assume that we are given a sentence embedding model, M, which returns a d-dimensional embedding, $\boldsymbol{s}(\in \mathbb{R}^d)$ to a sentence s. Given a

set of (train) sentences, $\mathcal{D}_{\text{train}}$, we will learn a d'-dimensional projection ($d' < d$), $f : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$ that would project a given sentence embedding. We consider SVD, PCA, GRP, KPCA and Autoencoders as the unsupervised dimensionality reduction methods for the purpose of learning this projection.[1] These methods have been selected due to their superior performance and popular applications. For example, PCA has been used for compressing word embeddings (Raunak et al., 2019), KPCA has been used for feature extraction (Ayesha et al., 2020; Gupta et al., 2019), SVD has shown excellent performance in diverse word embedding-related tasks (Levy et al., 2015), random projection is a popular lightweight lower-dimensional projection method (Schmidt, 2018), and autoencoders have been used to learn embeddings (Socher et al., 2011; Chandar AP et al., 2014). In our preliminary experiments, we observed that an autoencoder with a single hidden layer was producing comparable performance to ones with more than two hidden layers, despite the former being fast to train and infer with. Therefore, in the remainder of our experiments, we use autoencoders with a single hidden layer. The learnt f is then used to project test sentences, $\mathcal{D}_{\text{test}}$, to a d'-dimensional space.

4. Experiments

To evaluate the efficiency and effectiveness of the dimensionality reduction methods described in §3, we apply them to reduce the dimensionality of sentence embeddings which are produced by six transformer PLMs[2]: *all-mpnet-base-v2* (**mpnet**), *stsb-bert-base* (**sbert-b**), *msmarco-roberta-base-v2* (**roberta**), *paraphrase-xlm-r-multilingual-v1* (**xml-r**), *stsb-bert-large* (**sbert-l**), and *sup-simcse-roberta-large* (**simcse**). We evaluate using three datasets: STS-B for STS, TREC for question-type classification, and SICK-E for entailment prediction. We use an NVIDIA RTX A6000 GPU. The scikit-learn (0.24.2) is used for SVD, PCA, KPCA and GRP, and Keras (2.2.4) is used for the autoencoder. We fix these settings in all experiments.

4.1. Evaluation Tasks and Datasets

We use SentEval (Conneau and Kiela, 2018), an evaluation toolkit, to evaluate the quality of sentence embeddings. Following the implementation of sentence transformers to encode the sentences, we apply dimensionality reduction methods as a post-processing step to map the embeddings into a lower-dimensional space.

Three different tasks and corresponding datasets are selected.

Semantic Textual Similarity Prediction: STS-B is a set of pairwise sentences, provided with a standard benchmark of semantic similarity. The standard protocol when evaluating sentence embeddings on STS datasets is to first independently encode each sentence in a pair of sentences into a sentence embedding, and then compute the similarity between the two sentences in the pair using cosine similarity. The predicted similarity scores are compared against the human similarity ratings in the STS datasets using some correlation coefficient. The Spearman rank correlation coefficient is often used for this purpose. It is a metric between -1 and +1, where higher positive correlations indicate better agreement with human similarity ratings. Therefore, a sentence embedding method that produces high positive Spearman correlations on an STS dataset is considered to be better at preserving the semantic information of the sentences in the embedding space.

Question Classification: To correctly answer free-form factual questions given a large collection of texts, one must first understand the type of information that should be included in the answer to a question. For example, given the question *What Canadian city has the largest population?*, we would classify it as having the answer type **city**, implying that only cities are valid as the candidate answers. Li and Roth (2002) created a dataset from four question datasets including TREC collections and manually labelled into a hierarchical taxonomy of question types. Each question is classified into the the following six top-level types: ABBREVIATION, ENTITY, DESCRIPTION, HUMAN, LOCATION and NUMERIC.[3] The classification accuracy is measured for the effectiveness of embeddings to grasp the underlying intent of questions.

Textual Entailment: Given a premise and a hypothesis expressed by two sentences, in NLI, we are expected to predict whether the hypothesis has an entailment, contradiction or a neutral relation with the premise. For example, given the premise "*A small girl wearing a pink jacket is riding on a carousel*" and the hypothesis "*The carousel is moving*", there exists an entailment relationship between the premise and the hypothesis. Unlike the NLP

[1] Details of each method are in Appendix

[2] All models are available at huggingface.co/sentence-transformers and https://huggingface.co/princeton-nlp

[3] Details of the full hierarchy is available at https://cogcomp.seas.upenn.edu/Data/QA/QC/definition.html.

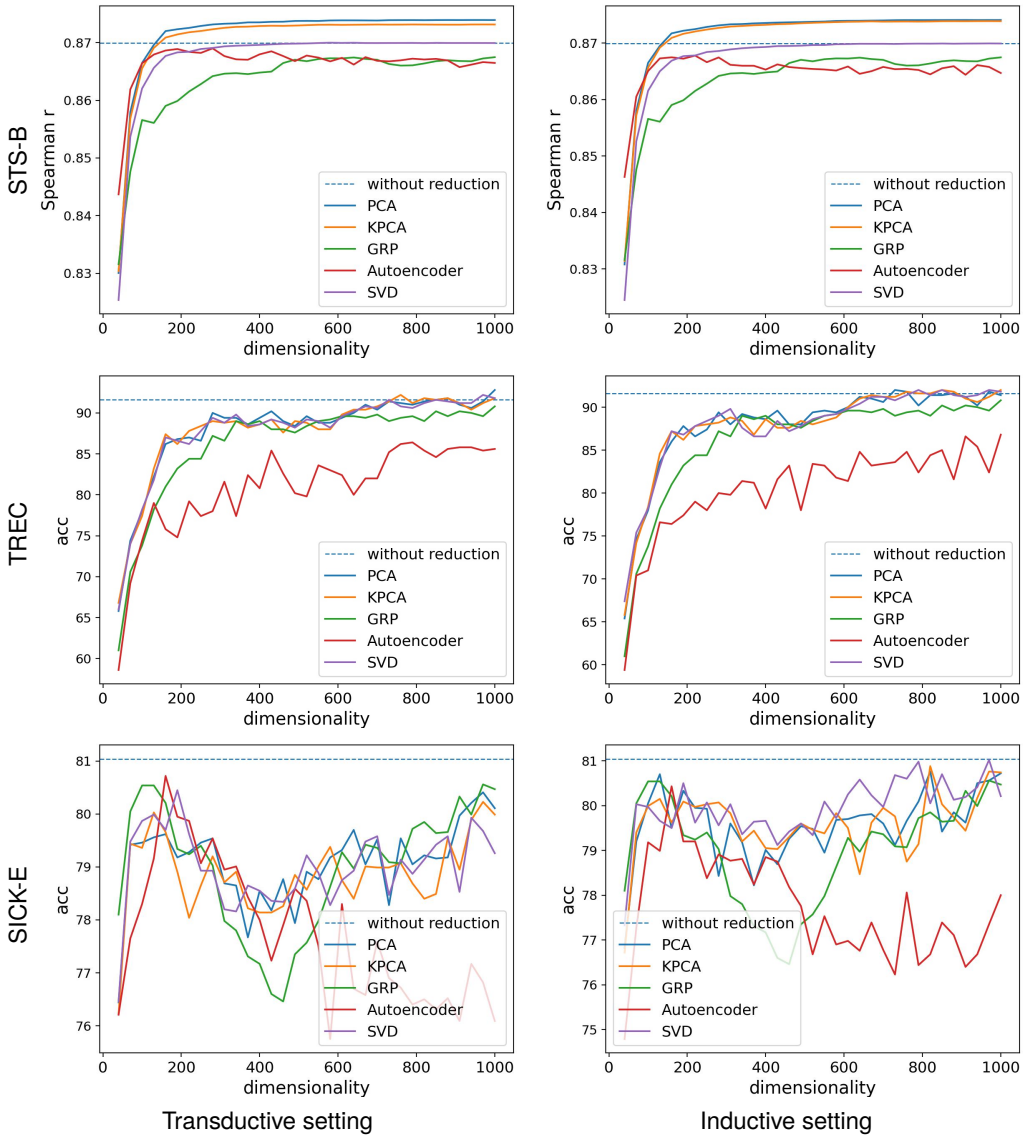

Figure 1: Performance of the original *sup-simcse-roberta-large* sentence embeddings and its dimensionality reduced versions produced using different methods on STS-B (top), TREC (middle) and SICK-E (bottom) datasets. Results for the transductive and inductive settings are shown respectively on the left and right.

tasks discussed above, NLI prediction requires us to represent each sentence in a pair of sentences, and make a prediction for the pair as a whole instead of individual sentences in the pair. We use the SICK Entailment (SICK-E) dataset (Marelli et al., 2014) for evaluating the performance of sentence embeddings for NLI. SICK-E is a set of pairwise sentences, provided with a standard benchmark of relationships (contradiction, neutral and entailment). The accuracy evaluates the ability of embeddings to discern semantic relations.

4.2. Results

Three factors are considered in our evaluations: tasks, sentence embeddings, and dimensions. Results for the SoTA simcse sentence embeddings are shown in Figure 1 (Other models are shown in Appendix). Overall, PCA reports good performance with a smaller number of dimensions across tasks and sentence embeddings. In particular, PCA reduces dimensionality by almost 50% without incurring a significant loss in task performance. Surprisingly, when the dimensionality is greater than 150, PCA even performs better than the original sentence embeddings in STS-B.

SVD's performance is similar to PCA in TREC, but SVD underperforms PCA for mpnet and simcse

Method	Training Time (s)	Inference time (s)
PCA	2.08	**0.0049**
KPCA	37.98	0.7883
SVD	2.57	0.0089
Autoencoder	101.16	0.1479
GRP	**0.03**	0.0080

Table 1: Training and inference times (wall-clock) for the different dimensionality reduction methods measured on the test set of STS-B under the inductive setting, with *all-mpnet-base-v2* reduced to 300 dimensions.

on STS-B. KPCA has comparable performance to PCA in TREC and STS-B with simcse. However, KPCA performs poorly in TREC and SICK-E with mpnet. The number of elements in the kernel matrix grows quadratically with the number of training instances. In particular, we observed that the kernel matrix does not always become positive definite due to the noise in the sentence embeddings, resulting in negative eigenvalues. Although it is possible to overcome this instability of the kernel matrix to an extent by applying small random perturbation prior to approximate eigenvalue decomposition (Halko et al., 2010), it still affects the performance of KPCA.

GRP never outperforms the original sentence embeddings in any task with simcse. Its performance is identical in both inductive and transductive settings because GRP does not learn the projection from the data. The performance of the Autoencoder with a single layer is unstable across different dimensionalities, compared to other methods, especially in TREC and SICK-E. Overall, the transductive setting is superior to the inductive setting due to its test data awareness.

We see that for some tasks (e.g. simcse and mpnet on STS-B, roberta on TREC, roberta and xlm-r on SICK-E) the performance *increases* when dimensions have been reduced. Such trends have been previously observed when SVD was used to obtain lower dimensional embeddings from co-occurrence-based word embeddings (Deerwester et al., 1990; Turney, 2005; Duc et al., 2010). Co-occurrences between word embeddings tend to be sparse, and applying SVD collapses dimensions that are similar, thereby creating dense word embeddings that produce non-zero cosine similarity scores. However, this does not explain the behavior observed with dense sentence embeddings used in our experiments. Although further investigations are required as the trend is observed with specific sentence encoders and on some datasets only, we believe this is due to a form of *noise reduction* due to PCA.

Table 1 compares the training and inference times for the dimensionality reduction methods measured on the test data from STS-B. We reduce sentence embeddings produced by mpnet from 768 to 300 dimensions, selected according to Figure 1, where most methods converge at. GRP does not use training data to learn the projection, hence reports the lowest training time among all methods.

On the other hand, KPCA reports the lowest inference time. Indeed, all PCA, SVD and GRP can be seen as multiplying a projection matrix onto the sentence embeddings corresponding to the test sentences to reduce the output dimensionality. This operation can be naively parallelised via vectorisation, which results in extremely fast and comparable inference times for those methods. On the other hand, Autoencoders and KPCA are both slow to train and infer with. Autoencoders are trained with mini-batch backpropagation and require multiple iterations to converge. Moreover, training and inference times of autoencoders further increase with the number of hidden layers. KPCA must first compute the kernel matrix, which requires all pairwise inner products to be computed for the training instances, resulting in increased training times.

5. Conclusion

We evaluated unsupervised dimensionality reduction methods for pre-trained sentence embeddings using multiple NLP tasks and benchmarks under transductive and inductive settings. The experimental results show that PCA performs consistently well across encoders and tasks. We hope our findings will encourage the use of sentence encoders in memory/compute-constrained applications and devices.

6. Limitations

In this paper, we considered the problem of reducing the dimensionality of sentence embeddings computed from PLMs. All PLMs we considered were trained specifically on the English language, with the exception of `paraphrase-xlm-r-multilingual-v1` , which is a multilingual sentence embedding model. However, all benchmark datasets that we used for evaluations (i.e. STS-B, TREC and SICK-E) cover only English, which is a morphologically limited language. Therefore, whether the findings reported in this paper scale to sentence embeddings for languages other than English remains an open question. Nevertheless, we note that all dimensionality reduction methods considered in this paper are unsupervised and hence do not use any labelled data for a particular task nor a language.

Although we focused on unsupervised dimensionality reduction methods, which can be applied

as a post-processing stage in this paper, as we already noted in §2 there are other methods for learning models that produce lower-dimensional sentence embeddings such as supervised dimensionality reduction methods and knowledge distillation-based methods. Conducting a comparison against all such methods is beyond the scope of this short paper and is deferred to future work. On the other hand, our experimental results show for the first time in published literature that even with simple unsupervised dimensionality reduction methods such as PCA one can obtain surprisingly accurate lower-dimensional sentence embeddings.

7. Ethics and Broader Impact

All datasets we used in our evaluations are collected, annotated and made publicly available in prior work on evaluating sentence embeddings. In particular, we have not collected nor annotated any data during this project. However, it has been reported that unfair social biases are found in STS (Rudinger et al., 2017; Webster et al., 2021) and NLI (Dev et al., 2019) datasets such as gender and racial biases. It is possible that such biases are reflected in our evaluations.

We use a broad range of pretrained sentence embedding models as inputs in this work. Unfortunately, it has been reported that sentence encoders have unfair social biases (May et al., 2019; Kurita et al., 2019; Kaneko et al., 2022). It remains unclear how such social biases are affected by the dimensionality reduction methods we evaluate in this paper. Although there has been prior work on evaluating social biases in text embeddings, to the best knowledge of ours, no work has evaluated the effect of dimensionality reduction in social biases. Therefore, we consider it to be an important task to evaluate the effect on social biases due to dimensionality reduction before the methods we consider in this paper are used widely in downstream NLP tasks that require compressed sentence embeddings.

References

Martin Andrews. 2016. Compressing word embeddings. In *International Conference on Neural Information Processing*, pages 413–422. Springer.

Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E. Dahl, and Geoffrey E. Hinton. 2018. Large scale distributed neural network training through online distillation. In *International Conference on Learning Representations*.

Farzana Anowar, Samira Sadaoui, and Bassant Selim. 2021. Conceptual and empirical comparison of dimensionality reduction algorithms (pca, kpca, lda, mds, svd, lle, isomap, le, ica, t-sne). *Computer Science Review*, 40:100378.

Shaeela Ayesha, Muhammad Kashif Hanif, and Ramzan Talib. 2020. Overview and comparative study of dimensionality reduction techniques for high dimensional data. *Information Fusion*, 59:44–58.

Ella Bingham and Heikki Mannila. 2001. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Sarath Chandar AP, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. *Advances in neural information processing systems*, 27.

Yunchuan Chen, Lili Mou, Yan Xu, Ge Li, and Zhi Jin. 2016. Compressing neural language models by sparse word representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 226–235, Berlin, Germany. Association for Computational Linguistics.

Hyunjin Choi, Judong Kim, Seongho Joe, and Youngjune Gwon. 2021. Evaluation of bert and albert sentence embedding performance on downstream nlp tasks. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 5482–5487.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzman, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis.

JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE, 41(6):391–407.

Sunipa Dev, Tao Li, Jeff Phillips, and Vivek Srikumar. 2019. On Measuring and Mitigating Biased Inferences of Word Embeddings.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Nguyen Tuan Duc, Danushka Bollegala, and Mitsuru Ishizuka. 2010. Using relational similarity between word pairs for latent relational search on the web. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 196 – 199.

Myroslava Dzikovska, Rodney Nielsen, Chris Brew, Claudia Leacock, Danilo Giampiccolo, Luisa Bentivogli, Peter Clark, Ido Dagan, and Hoa Trang Dang. 2013. SemEval-2013 task 7: The joint student response analysis and 8th recognizing textual entailment challenge. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 263–274, Atlanta, Georgia, USA. Association for Computational Linguistics.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021a. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021b. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Vishwani Gupta, Sven Giesselbach, Stefan Rüping, and Christian Bauckhage. 2019. Improving word embeddings using kernel PCA. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 200–208, Florence, Italy. Association for Computational Linguistics.

N. Halko, P. G. Martinsson, and J. A. Tropp. 2010. Finding structure with randomness: Probabilistic algorithms for constructung approximate matrix decompositions. *SIAM REVIEW*, 53(2):217 – 288.

Song Han, Huizi Mao, and William J. Dally. 2016. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both weights and connections for efficient neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1135–1143, Cambridge, MA, USA. MIT Press.

Yu Hao, Xien Liu, Ji Wu, and Ping Lv. 2019. Exploiting sentence embedding for medical question answering. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Masahiro Kaneko, Aizhan Imankulova, Danushka Bollegala, and Naoaki Okazaki. 2022. Gender bias in masked language models for multiple languages. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2740–2750, Seattle, United States. Association for Computational Linguistics.

Yeachan Kim, Kang-Min Kim, and SangKeun Lee. 2020. Adaptive compression of word embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3950–3959, Online. Association for Computational Linguistics.

Weize Kong, Swaraj Khadanga, Cheng Li, Shaleen Kumar Gupta, Mingyang Zhang, Wensong Xu, and Michael Bendersky. 2022. Multi-aspect dense retrieval. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 3178–3186, New York, NY, USA. Association for Computing Machinery.

Keita Kurita, Nidhi Vyas, Ayush Pareek, Alan W Black, and Yulia Tsvetkov. 2019. Measuring bias in contextualized word representations. In *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*, pages 166–172, Florence, Italy. Association for Computational Linguistics.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. 2019. Snip: single-shot network pruning based on connection sensitivity. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the association for computational linguistics*, 3:211–225.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2017. Pruning filters for efficient convnets. In *International Conference on Learning Representations*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland. European Language Resources Association (ELRA).

Chandler May, Alex Wang, Shikha Bordia, Samuel R. Bowman, and Rachel Rudinger. 2019. On measuring social biases in sentence encoders. 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 622–628, Minneapolis, Minnesota. Association for Computational Linguistics.

Aditya Menon, Sadeep Jayasumana, Ankit Singh Rawat, Seungyeon Kim, Sashank Reddi, and Sanjiv Kumar. 2022. In defense of dual-encoders for neural ranking. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 15376–15400. PMLR.

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

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. In *CoCo@ NIPs*.

Priyanka Nigam, Yiwei Song, Vijai Mohan, Vihan Lakshman, Weitian, Ding, Ankit Shingavi, Choon Hui Teo, Hao Gu, and Bing Yin. 2019. Semantic Product Search.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 24(4):694–707.

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

Kaare Brandt Petersen and Michael Syskind Petersen. 2012. *The Matrix Cookbook*. Online.

Vikas Raunak, Vivek Gupta, and Florian Metze. 2019. Effective dimensionality reduction for word embeddings. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 235–243.

G Thippa Reddy, M Praveen Kumar Reddy, Kuruva Lakshmanna, Rajesh Kaluri, Dharmendra Singh Rajput, Gautam Srivastava, and Thar Baker. 2020. Analysis of dimensionality reduction techniques on big data. *IEEE Access*, 8:54776–54788.

Nils Reimers and Iryna Gurevych. 2019a. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3980–3990, Hong Kong, China. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019b. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of*

the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics.

Rachel Rudinger, Chandler May, and Benjamin Van Durme. 2017. Social bias in elicited natural language inferences. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 74–79, Valencia, Spain. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *Proc. of the 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing at NeurIPS-2019*.

Benjamin Schmidt. 2018. Stable random projection: Lightweight, general-purpose dimensionality reduction for digitized libraries. *Journal of Cultural Analytics*, 3(1):11033.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. 1998. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10(5):1299–1319.

Raphael Shu and Hideki Nakayama. 2018. Compressing word embeddings via deep compositional code learning. In *International Conference on Learning Representations*.

Richard Socher, Eric Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. *Advances in neural information processing systems*, 24.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. MPNet: Masked and Permuted Pre-training for Language Understanding.

P.D. Turney. 2005. Measuring semantic similarity by latent relational analysis. In *Proc. of IJCAI'05*, pages 1136–1141.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antonie Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *ICML'08*, pages 1096 – 1103.

Rui Wang, Andrew Finch, Masao Utiyama, and Eiichiro Sumita. 2017. Sentence embedding for neural machine translation domain adaptation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 560–566, Vancouver, Canada. Association for Computational Linguistics.

Yasi Wang, Hongxun Yao, and Sicheng Zhao. 2016. Auto-encoder based dimensionality reduction. *Neurocomputing*, 184:232–242.

Kellie Webster, Xuezhi Wang, Ian Tenney, Alex Beutel, Emily Pitler, Ellie Pavlick, Jilin Chen, Ed Chi, and Slav Petrov. 2021. Measuring and reducing gendered correlations in pre-trained models.

Ken-ichi Yokote, Danushka Bollegala, and Mitsuru Ishizuka. 2012. Similarity is not entailment – jointly learning similarity transformations for textual entailment. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 1720 – 1726.

Xuandong Zhao, Zhiguo Yu, Ming Wu, and Lei Li. 2022. Compressing sentence representation for semantic retrieval via homomorphic projective distillation. *arXiv preprint arXiv:2203.07687*.

Appendix

A. Dimensionality Reduction Methods

A.1. Truncated SVD

Singular value decomposition (SVD) is a matrix factorization method that finds a lower-rank approximation to the original matrix by minimizing the squared Frobenious norm between the original matrix and its low-rank factorization. SVD generalizes the eigenvalue decomposition of square matrices to rectangular matrices. Specifically, given a matrix $\mathbf{X}(\in R^{m\times n})$ where m is the number of observations and n is the number of features, SVD decomposes it into the product of three matrices as given by (1).

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^* \tag{1}$$

Here, $\mathbf{U}$ is an $m \times m$ unitary matrix (columns are known as left singular vectors), Σ is an $m \times n$ diagonal matrix (diagonal values are known as singular values), $\mathbf{V}$ is an $n \times n$ unitary matrix, and $\mathbf{V}^*$ is the conjugate transpose of $\mathbf{V}$ (rows are known as right singular vectors). To obtain a lower-dimensional approximation, suppose the projection dimension is selected as k ($k \leq n$) and the first k columns of $\mathbf{V}^*$ of the training vector space are retained as a $n \times k$ matrix $\mathbf{V}_k$, known as the projection matrix. The target vector space of the original $m \times n$ matrix $\mathbf{X}$ will be projected as $\mathbf{X}\mathbf{V}_k$ The final dimensionality then becomes $m \times k$. For each sample, dimensionality is reduced to k.

A.2. PCA

Principal component analysis (PCA) is a process of projecting data into a new basis, by comput-

ing the principal components. PCA is an unsupervised orthogonal statistical technique where the interactions between features can be extracted by computing the covariance matrix of the original data. Specifically, suppose an $m \times n$ matrix $\mathbf{X}$ is represented by the original points where m is the number of observations and n is the number of features. PCA first standardizes $\mathbf{X}$. Suppose the target projection dimension is k ($k \leq n$). The normalised largest k eigenvectors of the covariance matrix ($\mathbf{X}\mathbf{X}^{\mathrm{T}}$) are chosen to form the new axes, forming a $n \times k$ vector space $\mathbf{U}$. The corresponding eigenvalues are used to order the eigenvectors. The variance of the projected data in the chosen direction is maximised to preserve the diversity of data in each dimension. This is, the information structure in the data is retained to the maximum extent. The target vector space of the original $m \times n$ matrix $\mathbf{X}$ will be projected as

$$\mathbf{X}\mathbf{U} \tag{2}$$

The final dimensionality then becomes $m \times k$.

A.3. Kernel PCA

Kernel principal component analysis (KPCA) is a non-linear dimensionality reduction method, which is an extension of PCA. Since PCA is limited to linear projections, KPCA uses kernel functions to deal with non-linear data and make it linearly separable (Anowar et al., 2021). KPCA works by first projecting original data to higher-dimensional space with kernel functions such as Polynomial and Sigmoid kernels. Specifically, suppose an $m \times n$ matrix $\mathbf{X}$ is represented by the original points. Suppose the target projection dimension is k ($k \leq n$). KPCA maps $\mathbf{X}$ into a higher feature space by function $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^d$ ($d > n$), where Φ makes data linearly separable. Then a kernel matrix is generated

$$\mathbf{K} = \Phi(\mathbf{X})^T \Phi(\mathbf{X}).$$

After centering the kernel matrix, eigendecomposition is utilized to compute the eigenvectors and eigenvalues like PCA. Procedures of PCA is thus applied in the following calculation to reduce the sample dimension to k ($k < n$).

A.4. Gaussian random projection

Gaussian Random Projection is a technique used to reduce the dimensions by projecting the original high-dimensional input space using a randomly generated matrix. The core idea behind random projection is Johnson-Lindenstrauss lemma, which states that distances between points can be nearly preserved when embedding high-dimensional space into a much lower-dimensional space. Specifically, suppose the original dataset is defined as a $m \times n$ matrix $\mathbf{X}$. Suppose the target projection dimension is k ($k \leq n$). To obtain a lower-dimensional projection of the input data, first a set of k-dimensional normally distributed random vectors is generated. Then the projection matrix $n \times k$ matrix $\mathbf{R}$ is produced by stacking those vectors. Finally, the original matrix $m \times n$ matrix $\mathbf{X}$ will be projected as in (3).

$$\mathbf{X}\mathbf{R} \tag{3}$$

The final dimensionality then becomes $m \times k$.

A.5. Autoencoders

Autoencoders can be seen as a non-linear version of PCA, where an encoder network first projects the inputs to a (possibly lower-dimensional) space. Next, some non-linear function is applied elementwise on the encoded input. Finally, a decoder network attempts to reconstruct the input. The encoder and decoder network parameters are jointly learned such that the reconstruction loss (e.g. measured by the squared ℓ_2 distance between the input and its corresponding reconstruction) is minimized. To compress the input, we can constrain the dimensionality of the hidden layer (Wang et al., 2016) to be smaller than the input dimensionality.

B. Sentence Encoders

SentenceTransformers (Reimers and Gurevych, 2019b) is a Python framework to embed sentences with state-of-the-art models. Spcific sentence encoders we use are *all-mpnet-base-v2* (Song et al., 2020), *msmarco-roberta-base-v2* (Reimers and Gurevych, 2019b), *paraphrase-xlm-r-multilingual-v1* (Conneau et al., 2020), *stsb-bert-base* (Reimers and Gurevych, 2019b) and *stsb-bert-large* (Reimers and Gurevych, 2019b). Additionally, *all-mpnet-base-v2* is fine-tuned on a sentence pairs dataset, based on *microsoft/mpnet-base* model (Song et al., 2020). *stsb-bert-base* and *stsb-bert-large* are fine-tuned on STS-B dataset, based on BERT (Devlin et al., 2018). *msmarco-roberta-base-v2* is fine-tuned on Microsoft Machine Reading Comprehension (MS MARCO) dataset (Nguyen et al., 2016), based on RoBERTa (Liu et al., 2019). Princeton-NLP provides a Python-version sentence encoder *sup-simcse-roberta-large* (Gao et al., 2021b).

C. Experiment Results for Sentence Encoders

PCA outperforms uncompressed embeddings for some sentence encoders such as xlm-r on TREC. SVD performs comparably to PCA in most cases.

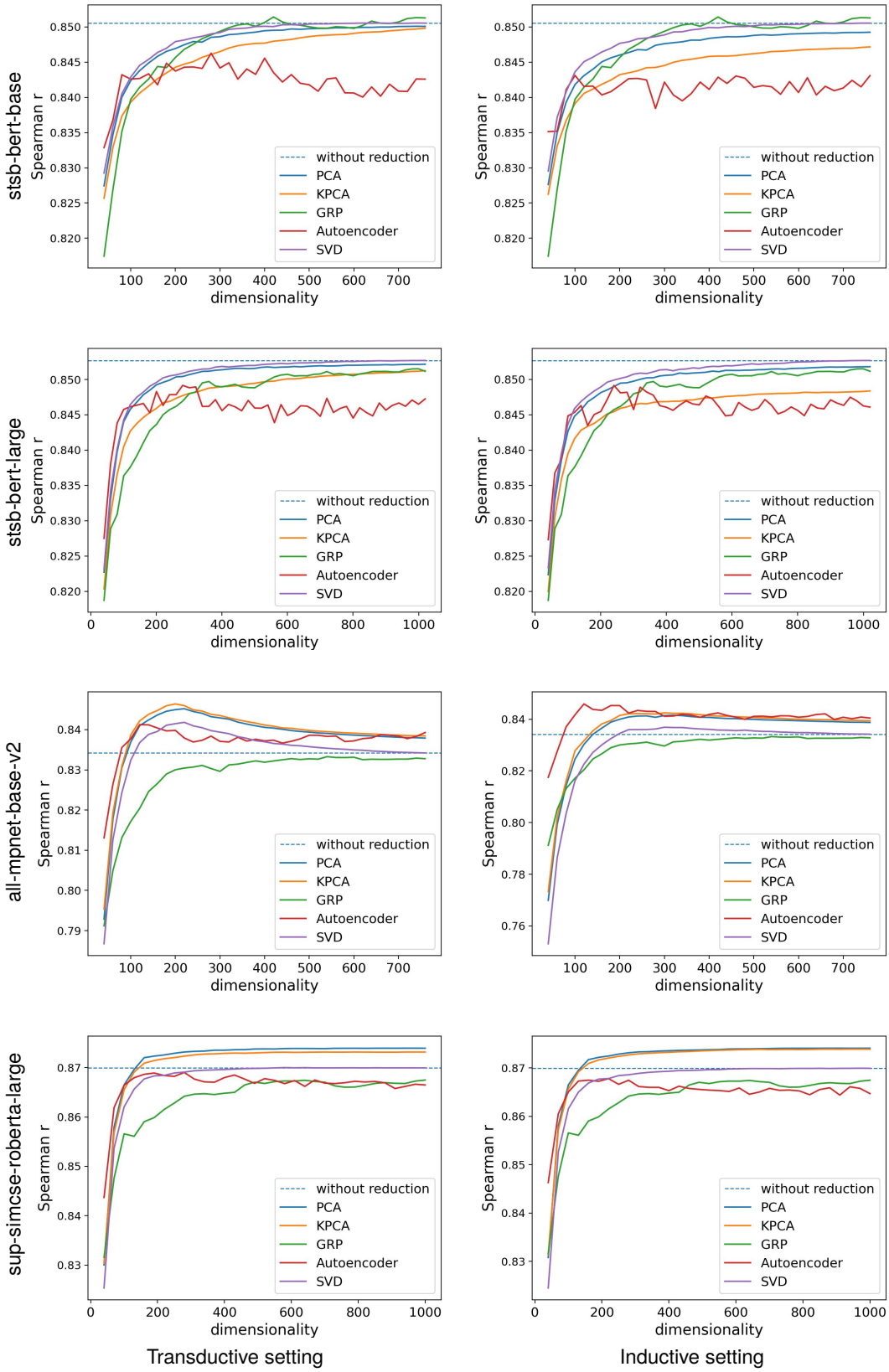

Figure 2: Spearman correlation coefficients on STS-B vs. the dimensionality of the sentence embeddings produced by applying different dimensionality reduction methods. Sentence embeddings are created using pre-trained `stsb-bert-base` (uppermost), `stsb-bert-large` (supper-middle), `all-mpnet-base-v2` (lower-middle) and `sup-simcse-roberta-large` (lowermost) models. Results for the transductive and inductive settings are shown respectively on the left and right.

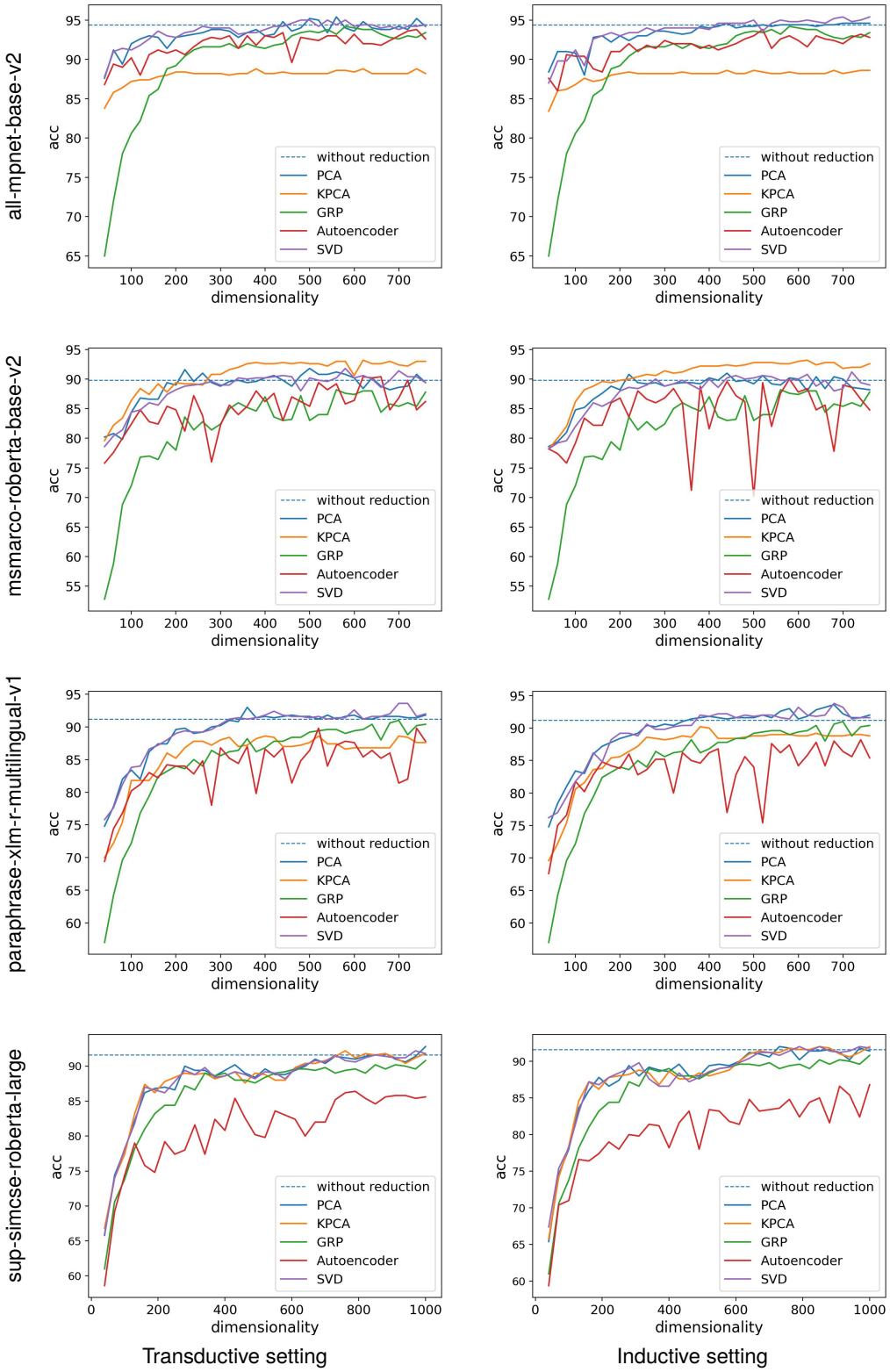

Figure 3: Classification accuracy on TREC vs. the dimensionality of the sentence embeddings produced by applying different dimensionality reduction methods. Sentence embeddings are created using pre-trained `all-mpnet-base-v2` (uppermost), `msmarco-roberta-base-v2` (supper-middle), `paraphrase-xlm-r-multilingual-v1` (lower-middle) and `sup-simcse-roberta-large` (lowermost) models. Results for the transductive and inductive settings are shown respectively on the left and right.

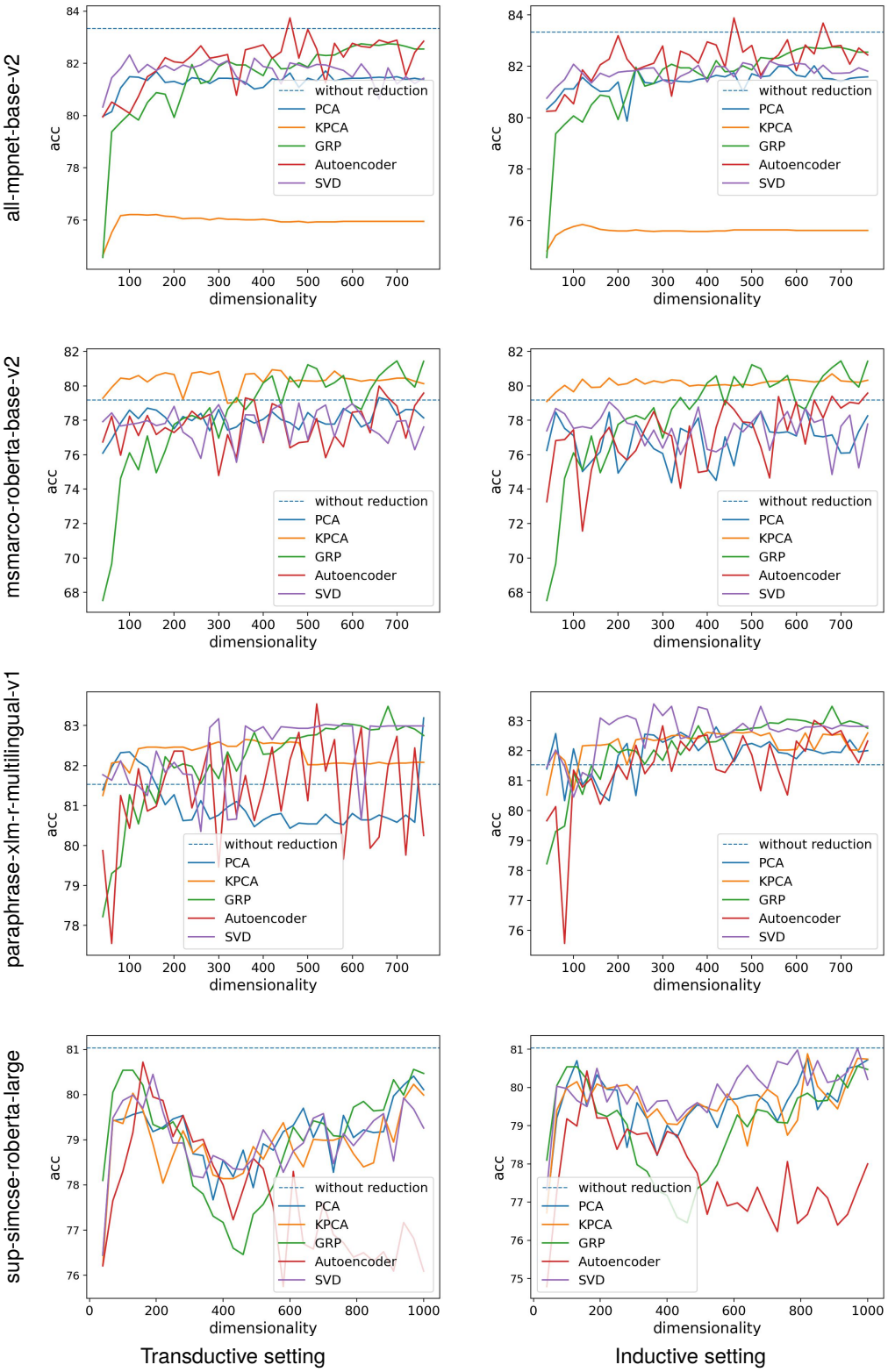

Figure 4: Accuracy on SICK-E vs. the dimensionality of the sentence embeddings produced by applying different dimensionality reduction methods. Sentence embeddings are created using pre-trained `all-mpnet-base-v2` (uppermost), `msmarco-roberta-base-v2` (supper-middle), `paraphrase-xlm-r-multilingual-v1` (lower-middle) and `sup-simcse-roberta-large` (lowermost) models. Results for the transductive and inductive settings are shown respectively on the left and right.

However, SVD improves accuracy for both transductive and inductive settings for xlm-r on SICK-E. Meanwhile, PCA shows poor performance in the transductive setting in this case compared to the remainder of the methods.

KPCA shows varying levels of performance for different sentence encoders on different tasks. KPCA performs similarly to PCA in some situations, especially for mpnet and simcse on STS-B. KPCA reports excellent performance for some sentence encoders such as roberta on TREC and SICK-E and xlm-r on SICK-E, by even outperforming the original uncompressed embeddings.

GRP reports suboptimal performance in many settings. In particular, mpnet on STS-B, GRP preserves much information in the original embeddings without a significant loss for dimensionalities over 200, but shows a much lower performance than other methods. However, its performance on roberta and xlm-r for SICK-E is noteworthy.

Autoencoder gives relatively poor performance in many cases, especially for sentence encoders simcse, sbert-b, sbert-l and mpnet. Additionally, autoencoder has the most expensive train and infer costs among all the dimensionality reduction compared in this paper.