WhiteningBERT: An Easy Unsupervised Sentence Embedding Approach

Junjie Huang[1*], **Duyu Tang**[4], **Wanjun Zhong**[2*], **Shuai Lu**[3*],
Linjun Shou[5], **Ming Gong**[5], **Daxin Jiang**[5], **Nan Duan**[4]

[1]Beihang University   [2]Sun Yat-sen University   [3]Peking University
[4]Microsoft Research Asia   [5]Microsoft STC Asia

huangjunjie@buaa.edu.cn
zhongwj25@mail2.sysu.edu.cn, lushuai96@pku.edu.cn
{dutang,lisho,migon,djiang,nanduan}@microsoft.com

Abstract

Producing the embedding of a sentence in an unsupervised way is valuable to natural language matching and retrieval problems in practice. In this work, we conduct a thorough examination of pretrained model based unsupervised sentence embeddings. We study on four pretrained models and conduct massive experiments on seven datasets regarding sentence semantics. We have three main findings. First, averaging all tokens is better than only using $[CLS]$ vector. Second, combining both top and bottom layers is better than only using top layers. Lastly, an easy whitening-based vector normalization strategy with less than 10 lines of code consistently boosts the performance. [1]

1 Introduction

Pre-trained language models (PLMs) (Devlin et al., 2019; Liu et al., 2019) perform well on learning sentence semantics when fine-tuned with supervised data (Reimers and Gurevych, 2019; Thakur et al., 2020). However, in practice, especially when a large amount of supervised data is unavailable, an approach that provides sentence embeddings in an unsupervised way is of great value in scenarios like sentence matching and retrieval. While there are attempts on unsupervised sentence embeddings (Arora et al., 2017; Zhang et al., 2020), to the best of our knowledge, there is no comprehensive study on various PLMs with regard to multiple factors. Meanwhile, we aim to provide an easy-to-use toolkit that can be used to produce sentence embeddings upon various PLMs.

In this paper, we investigate PLMs-based unsupervised sentence embeddings from three aspects. First, a standard way of obtaining sentence embedding is to pick the vector of $[CLS]$ token. We explore whether using the hidden vectors of other tokens is beneficial. Second, some works suggest producing sentence embedding from the last layer or the combination of the last two layers (Reimers and Gurevych, 2019; Li et al., 2020). We seek to figure out whether there exists a better way of layer combination. Third, recent attempts transform sentence embeddings to a different distribution with sophisticated networks (Li et al., 2020) to address the problem of non-smooth anisotropic distribution. Instead, we aim to explore whether a simple linear transformation is sufficient.

To answer these questions, we conduct thorough experiments upon 4 different PLMs and evaluate on 7 datasets regarding semantic textual similarity. We find that, first, to average the token representations consistently yields better sentence representations than using the representation of the $[CLS]$ token. Second, combining the embeddings of the bottom layer and the top layer performs than using top two layers. Third, normalizing sentence embeddings with whitening, an easy linear matrix transformation algorithm with less than 10 lines of code (§A.3), consistently brings improvements.

2 Transformer-based PLMs

Multi-layer Transformer architecture (Vaswani et al., 2017) has been widely used in pre-trained language models (e.g. Devlin et al., 2019; Liu et al., 2019) to encode sentences. Given an input sequence $S = \{s_1, s_2, \ldots, s_n\}$, a transformer-based PLM produces a set of hidden representations $H^{(0)}, H^{(1)}, \ldots, H^{(L)}$, where $H^{(l)} = [\mathbf{h}_1^{(l)}, \mathbf{h}_2^{(l)}, \ldots, \mathbf{h}_n^{(l)}]$ are the per-token embeddings of S in the l-th encoder layer and $H^{(0)}$ corresponds to the non-contextual word(piece) embeddings.

In this paper, we use four transformer-based PLMs to derive sentence embeddings, i.e. BERT-base (Devlin et al., 2019), RoBERTa-base (Liu et al., 2019), DistilBERT (Sanh et al., 2019), and LaBSE (Feng et al., 2020). They vary in the model

---

architecture and pre-training objectives. Specifically, BERT-base, RoBERTa-base, and LaBSE follow an architecture of twelve layers of transformers but DistilBERT only contains six layers. Additionally, LaBSE is pre-trained with a unique translation ranking task which forces the sentence embeddings of a parallel sentence pair to be closer, while the other three PLMs do not include such a pre-training task for sentence embeddings.

3   WhiteningBERT

In this section, we introduce the three strategies to derive the sentence embedding $\mathbf{s}$ from PLMs.

3.1 $[CLS]$ Token v.s. Average Tokens

Taking the last layer of token representations as an example, we compare the following two methods to obtain sentence embeddings: (1) using the vector of $[CLS]$ token which is the first token of the sentence, i.e., $\mathbf{s} = \mathbf{s}^L = \mathbf{h}_1^L$; (2) averaging the vectors of all tokens in the sentence, including the $[CLS]$ token, i.e., $\mathbf{s} = \mathbf{s}^L = \frac{1}{n}\sum_{i=1}^{N}\mathbf{h}_i^L$.

3.2   Layer Combination

Most works only take the last layer to derive sentence embeddings, while rarely explore which layer of semantic representations can help to derive a better one. Here we explore how to best combine layers of embeddings to obtain sentence embeddings. Specifically, we first compute the vectors of each layer following §3.1. Then we perform layer combinations as $\mathbf{s} = \sum_l \mathbf{s}^l$ to acquire the sentence embedding. For example, for the combination of L1+L12 with two layers, we obtain sentence embeddings by averaging the vector representations of layer one and layer twelve, i.e., $\mathbf{s} = \frac{1}{2}(\mathbf{s}^1 + \mathbf{s}^{12})$.

3.3   Whitening

Whitening is a linear transformation that transforms a vector of random variables with a known covariance matrix into a new vector whose covariance is an identity matrix, and has been verified effective to improve the text representations in bilingual word embedding mapping (Artetxe et al., 2018) and image retrieval (Jégou and Chum, 2012).

In our work, we explore to address the problem of non-smooth anisotropic distribution (Li et al., 2020) by a simple linear transformation called whitening. Specifically, given a set of d-dimensional embeddings of N sentences $\mathbf{E} = \{\mathbf{s}_1, \ldots, \mathbf{s}_N\} \in \mathbb{R}^{N \times d}$, we transform $\mathbf{E}$ linearly

as in Eq. 1 such that $\hat{\mathbf{E}} \in \mathbb{R}^{N \times d}$ is the whitened sentence embeddings,

$$\hat{\mathbf{E}} = (\mathbf{E} - m)UD^{-\frac{1}{2}}, \qquad (1)$$

where $m \in \mathbb{R}^d$ is the mean vector of $\mathbf{E}$, D is a diagonal matrix with the eigenvalues of the covariance matrix $Cov(\mathbf{E}) = (\mathbf{E} - m)^T(\mathbf{E} - m) \in \mathbb{R}^{d \times d}$ and U is the corresponding orthogonal matrix of eigenvectors, satisfying $Cov(\mathbf{E}) = UDU^T$.

4   Experiment

We evaluate sentence embeddings on the task of unsupervised semantic textual similarity. We show experimental results and report the best way to derive unsupervised sentence embedding from PLMs.

4.1   Experiment Settings

Task and Datasets   The task of unsupervised semantic textual similarity (STS) aims to predict the similarity of two sentences without direct supervision. We experiment on seven STS datasets, namely the STS-Benchmark (STS-B) (Cer et al., 2017), the SICK-Relatedness (Marelli et al., 2014), and the STS tasks 2012-2016 (Agirre et al., 2012, 2013, 2014, 2015, 2016). These datasets consist of sentence pairs with labeled semantic similarity scores ranging from 0 to 5.

Evaluation Procedure   Following the procedures in SBERT (Reimers and Gurevych, 2019), we first derive sentence embeddings for each sentence pair and compute their cosine similarity score as the predicted similarity. Then we compute the Spearman's rank correlation coefficient between the predicted similarity and gold standard similarity scores as the evaluation metric. We average the Spearman's coefficients among the seven datasets as the final correlation score.

Baseline Methods   We compare our methods with five representative unsupervised sentence embedding models, including average GloVe embedding (Pennington et al., 2014), SIF (Arora et al., 2017), IS-BERT (Zhang et al., 2020) and BERT-flow (Li et al., 2020), SBERT-WK with BERT (Wang and Kuo, 2020).

4.2   Overall Results

Table 1 shows the overall performance of sentence embeddings. We can observe that:

(1) Averaging the token representations of the last layer to derive sentence embeddings performs

Models	STSB	SICK	STS-12	STS-13	STS-14	STS-15	STS-16	Avg.
Baselines								
Avg. GloVe (Reimers and Gurevych, 2019)	58.02	53.76	55.14	70.66	59.73	68.25	63.66	61.32
SIF (GloVe+WR) (Arora et al., 2017)	-	-	56.20	56.60	68.50	71.70	-	63.25
IS-BERT-NLI (Zhang et al., 2020)	69.21	64.25	56.77	69.24	61.21	75.23	70.16	66.58
BERT-flow (NLI) (Li et al., 2020)	58.56	65.44	59.54	64.69	64.66	72.92	71.84	65.38
SBERT WK (BERT) (Wang and Kuo, 2020)	16.07	41.54	26.66	14.74	24.32	28.84	34.37	26.65
WhiteningBERT (PLM=BERT-base)								
token=CLS, *layer*=L12, *whitening*=F	20.29	42.42	32.50	23.99	28.50	35.51	51.08	33.47
token=AVG, *layer*=L12, *whitening*=F	47.29	58.22	50.08	52.91	54.91	63.37	64.94	55.96
token=AVG, *layer*=L1, *whitening*=F	58.15	61.78	58.71	58.21	62.51	68.86	67.38	62.23
token=AVG, *layer*=L1+L12, *whitening*=F	59.05	63.75	57.72	58.38	61.97	70.28	69.63	62.97
token=AVG, *layer*=L1+L12, *whitening*=T	**68.68**	60.28	**61.94**	**68.47**	**67.31**	**74.82**	**72.82**	**67.76**
WhiteningBERT (PLM=RoBERTa-base)								
token=CLS, *layer*=L12, *whitening*=F	38.80	61.89	45.38	36.25	47.99	53.94	59.48	49.10
token=AVG, *layer*=L12, *whitening*=F	55.43	62.03	53.80	46.55	56.61	64.97	63.61	57.57
token=AVG, *layer*=L1, *whitening*=F	51.85	57.87	56.70	48.03	57.08	62.83	57.64	56.00
token=AVG, *layer*=L1+L12, *whitening*=F	57.54	**60.75**	58.56	50.37	59.62	66.64	63.21	59.53
token=AVG, *layer*=L1+L12, *whitening*=T	**69.43**	59.56	**62.46**	**66.29**	**68.44**	**74.89**	**72.94**	**67.72**
WhiteningBERT (PLM=DistilBERT)								
token=CLS, *layer*=L6, *whitening*=F	30.96	47.73	40.91	31.30	39.49	40.64	57.96	41.29
token=AVG, *layer*=L6, *whitening*=F	57.17	63.53	56.16	59.83	60.42	67.81	69.01	61.99
token=AVG, *layer*=L1, *whitening*=F	55.35	61.34	57.57	53.79	60.55	67.06	63.60	59.89
token=AVG, *layer*=L1+L6, *whitening*=F	61.45	**63.84**	59.67	59.50	63.54	70.95	69.90	64.12
token=AVG, *layer*=L1+L6, *whitening*=T	**70.37**	58.31	**62.09**	**68.78**	**68.99**	**75.06**	**74.52**	**68.30**
WhiteningBERT (PLM=LaBSE)								
token=CLS, *layer*=L12, *whitening*=F	67.18	69.43	66.99	61.26	68.36	77.13	73.10	69.06
token=AVG, *layer*=L12, *whitening*=F	71.02	68.36	67.81	63.94	70.56	77.93	75.07	70.67
token=AVG, *layer*=L1, *whitening*=F	53.70	55.25	54.81	44.62	56.97	60.30	54.57	54.32
token=AVG, *layer*=L1+L12, *whitening*=F	72.56	**68.36**	68.30	65.75	71.41	78.90	75.68	71.56
token=AVG, *layer*=L1+L12, *whitening*=T	**73.32**	63.27	**68.45**	**71.11**	**71.66**	**79.30**	**74.87**	**71.71**

Table 1: Spearman's rank correlation coefficient ($\rho \times 100$) between similarity scores assigned by sentence embeddings and humans. *token*=AVG or *token*=CLS denote using the average vectors of all tokens or only the $[CLS]$ token. L1 or L12 (L6) means using the hidden vectors of layer one or the last layer. Since DistilBERT only contains six layers of transformers, we use L6 as the last layer. T and F denote applying whitening (T) or not (F). Bold numbers indicate the best performance w.r.t the PLM.

better than only using $[CLS]$ token in the last layer by a large margin, no matter which PLM we use, which indicates that single $[CLS]$ token embedding does not convey enough semantic information as a sentence representation, despite it has been proved effective in a number of supervised classification tasks. This finding is also consistent with the results in Reimers and Gurevych (2019). Therefore, we suggest inducing sentence embeddings by averaging token representations.

(2) Averaging the token representations in layer one and the last layer performs better than separately using only one layer, regardless of the PLM. Since PLMs capture a rich hierarchy of linguistic information in different layers (Tenney et al., 2019; Jawahar et al., 2019), layer combination is capable of fusing the semantic information in different layers and thus yields better performance. Therefore, we suggest averaging the last layer and layer one to perform layer combination and induce better

sentence embeddings.

(3) Introducing the whitening strategy produces consistent improvement of sentence embeddings on STS tasks. This result indicates the effectiveness of the whitening strategy in deriving sentence embeddings. (Results with more PLMs can be found in Appendix A.1.) Among the four PLMs, LaBSE achieves the best STS performance while obtains the least performance enhancement after incorporating whitening strategy. We attribute it to the good intrinsic representation ability because LaBSE is pre-trained by a translation ranking task which improves the sentence embedding quality.

4.3 Analysis of Layer Combination

To further investigate the effects of layer combination, we add up the token representations of different layers to induce sentence embeddings.

First, we explore whether adding up layer one and the last layer is consistently better than other

BERT-base

	L0	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12
L0	55.45												
L1	59.88	62.23											
L2	59.87	62.07	61.14										
L3	59.28	61.73	60.75	59.57									
L4	58.34	61.08	60.06	58.76	57.32								
L5	58.87	61.41	60.50	59.18	57.62	57.12							
L6	58.90	61.46	60.57	59.19	57.64	57.03	56.17						
L7	59.43	61.96	61.07	59.79	58.18	57.55	56.61	56.19					
L8	59.57	62.25	61.41	60.09	58.46	57.83	56.82	56.32	55.45				
L9	58.50	61.60	60.85	59.56	57.96	57.38	56.41	55.89	54.92	53.51			
L10	59.68	62.31	61.72	60.62	59.27	58.76	57.94	57.47	56.66	55.09	55.68		
L11	59.43	62.22	61.62	60.60	59.33	58.92	58.17	57.80	57.05	55.45	55.94	55.41	
L12	59.93	62.97	62.26	61.17	59.67	59.24	58.52	58.27	57.69	56.05	56.68	56.17	55.96

RoBERTa-base

	L0	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12
L0	54.93												
L1	55.95	56.00											
L2	57.05	57.13	57.10										
L3	58.63	58.57	58.28	58.36									
L4	58.81	58.98	58.64	58.54	57.71								
L5	58.57	59.13	58.71	58.53	57.62	56.88							
L6	58.40	59.17	58.78	58.62	57.71	56.97	56.19						
L7	59.82	60.30	59.95	59.81	58.86	58.11	57.39	57.80					
L8	59.42	60.14	59.75	59.61	58.73	58.03	57.31	57.77	57.21				
L9	59.97	60.70	60.31	60.05	59.26	58.54	57.84	58.29	57.78	57.85			
L10	59.95	60.81	60.35	60.19	59.33	58.55	57.92	58.46	57.96	58.00	57.70		
L11	60.68	61.36	60.91	60.80	59.97	59.16	58.54	59.07	58.62	58.69	58.32	58.67	
L12	59.60	59.53	59.60	60.02	59.18	58.37	57.72	58.70	58.20	58.42	58.12	58.61	57.57

DistilBERT

	L0	L1	L2	L3	L4	L5	L6
L0	56.42						
L1	59.01	59.90					
L2	57.88	58.84	56.13				
L3	58.55	59.36	56.68	56.11			
L4	60.14	60.84	58.24	57.36	57.21		
L5	64.12	64.53	62.47	61.44	61.07	63.09	
L6	63.34	64.12	61.96	61.12	60.96	63.16	61.99

LaBSE

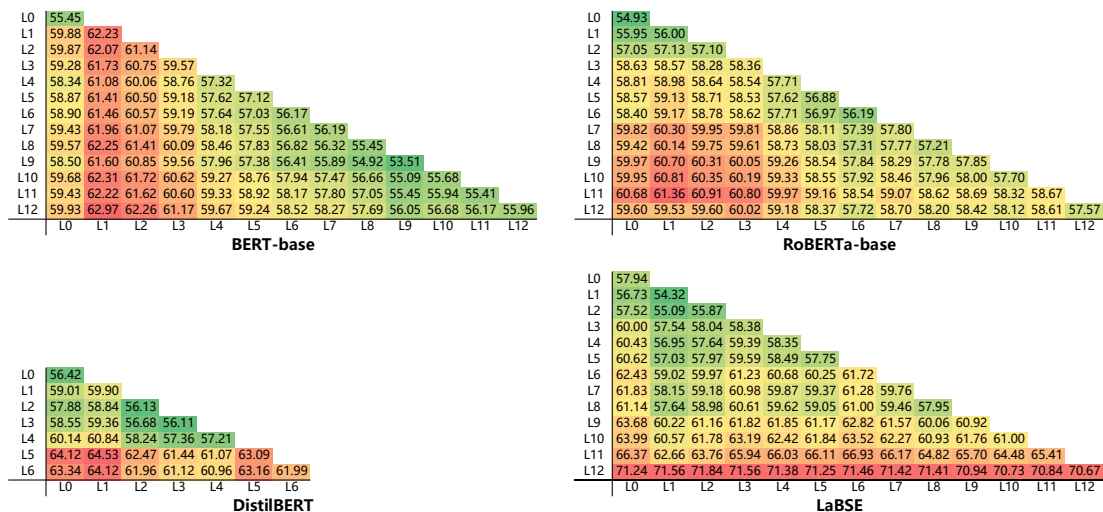|	L0	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12
L0	57.94												
L1	56.73	54.32											
L2	57.52	55.09	55.87										
L3	60.00	57.54	58.04	58.38									
L4	60.43	56.95	57.64	59.39	58.35								
L5	60.62	57.03	57.97	59.59	58.49	57.75							
L6	62.43	59.02	59.97	61.23	60.68	60.25	61.72						
L7	61.83	58.15	59.18	60.98	59.87	59.37	61.28	59.76					
L8	61.14	57.64	58.98	60.61	59.62	59.05	61.00	59.46	57.95				
L9	63.68	60.22	61.16	61.82	61.85	61.17	62.82	61.57	60.06	60.92			
L10	63.99	60.57	61.78	63.19	62.42	61.84	63.52	62.27	60.93	61.76	61.00		
L11	66.37	62.66	63.76	65.94	66.03	66.11	66.93	66.17	64.82	65.70	64.48	65.41	
L12	71.24	71.56	71.84	71.56	71.38	71.25	71.46	71.42	71.41	70.94	70.73	70.84	70.67

Figure 1: Performance of sentence embeddings of two layers of combinations. X-axis and Y-axis denote the layer index. Each cell is the average correlation score of seven STS tasks of two specific layer combinations. The redder the cell is, the better performance the corresponding sentence embeddings achieve.

combinations of two layers. Figure 1 shows the performance of all two-layer combinations. We find that adding up the last layer and layer one does not necessarily performs best among all PLMs, but could be a satisfying choice for simplicity.

Second, we explore the effects of the number of layers to induce sentence embeddings. We evaluate on BERT-base and figure 2 shows the maximum correlation score of each group of layer combinations. By increasing the number of layers, the maximum correlation score increases first but then drops. The best performance appears when the number of layers is three (L1+L2+L12). This indicates that combining three layers is sufficient to yield good sentence representations and we do not need to incorporating more layers which is not only complex but also poorly performed.

5 Related works

Unsupervised sentence embeddings are mainly composed with pre-trained (contextual) word embeddings (Pennington et al., 2014; Devlin et al., 2019). Recent attempts can be divided into two categories, according to whether the pre-trained embeddings are further trained or not. For the former, some works leverage unlabelled natural language inference datasets to train a sentence encoder without direct supervision (Li et al., 2020; Zhang et al., 2020; Mu and Viswanath, 2018). For the latter, some works propose weighted average word embeddings based on word features (Arora et al., 2017; Ethayarajh, 2018; Yang et al., 2019; Wang
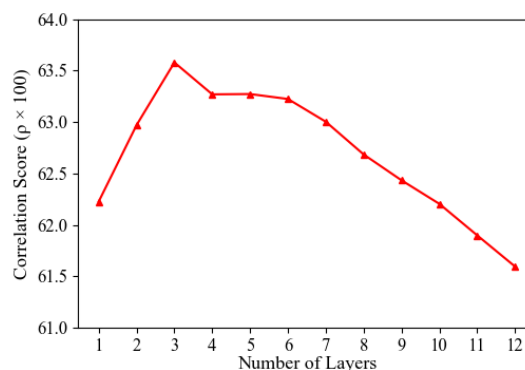
Figure 2: Maximum correlation scores of sentence embeddings from BERT-base with different numbers of combining layers. Combining three layers performs best than of other layer numbers. Especially the best combination is L1+L2+L12.

and Kuo, 2020). However, these approaches need further training or additional features, which limits the direct applications of sentence embeddings in real-world scenarios. Finally, we note that concurrent to this work, Su et al. (2021) also explored whitening sentence embedding, released to arXiv one week before our paper.

6 Conclusion

In this paper, we explore to find a simple and effective way to produce sentence embedding upon various PLMs. Through exhaustive experiments, we make three empirical conclusions here. First, averaging all token representations consistently induces better sentence representations than using the $[CLS]$ token embedding. Second, combining the

embeddings of the bottom layer and the top layer outperforms that using the top two layers. Third, normalizing sentence embeddings with a whitening algorithm consistently boosts the performance.

Acknowledgment

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Matthew Cer, Mona T. Diab, A. Gonzalez-Agirre, Weiwei Guo, I. Lopez-Gazpio, M. Maritxalar, R. Mihalcea, German Rigau, L. Uria, and J. Wiebe. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *SemEval@NAACL-HLT*.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Matthew Cer, Mona T. Diab, A. Gonzalez-Agirre, Weiwei Guo, R. Mihalcea, German Rigau, and J. Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *SemEval@COLING*.

Eneko Agirre, Carmen Banea, Daniel Matthew Cer, Mona T. Diab, A. Gonzalez-Agirre, R. Mihalcea, German Rigau, and J. Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *SemEval@NAACL-HLT*.

Eneko Agirre, Daniel Matthew Cer, Mona T. Diab, and A. Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *SemEval@NAACL-HLT*.

Eneko Agirre, Daniel Matthew Cer, Mona T. Diab, A. Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity. In *SEM@NAACL-HLT*.

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

M. Artetxe, Gorka Labaka, and Eneko Agirre. 2018. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *AAAI*.

T. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, G. Krüger, T. Henighan, R. Child, A. Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, E. Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, J. Clark,

Christopher Berner, Sam McCandlish, A. Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Daniel Matthew Cer, Mona T. Diab, Eneko Agirre, I. Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *SemEval@ACL*.

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S. Weld. 2020. Specter: Document-level representation learning using citation-informed transformers. *ArXiv*, abs/2004.07180.

J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Kawin Ethayarajh. 2018. Unsupervised random walk sentence embeddings: A strong but simple baseline. In *Rep4NLP@ACL*.

Fangxiaoyu Feng, Yin-Fei Yang, Daniel Matthew Cer, N. Arivazhagan, and Wei Wang. 2020. Language-agnostic bert sentence embedding. *ArXiv*, abs/2007.01852.

Xiang Gao, Yizhe Zhang, Michel Galley, Chris Brockett, and B. Dolan. 2020. Dialogue response ranking training with large-scale human feedback data. In *EMNLP*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *ArXiv*, abs/2006.03654.

Forrest N. Iandola, Albert Eaton Shaw, R. Krishna, and K. Keutzer. 2020. Squeezebert: What can computer vision teach nlp about efficient neural networks? *ArXiv*, abs/2006.11316.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bert learn about the structure of language? In *ACL*.

H. Jégou and O. Chum. 2012. Negative evidences and co-occurences in image retrieval: The benefit of pca and whitening. In *ECCV*.

Mandar Joshi, Danqi Chen, Y. Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2019. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. In *NeurIPS*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from pre-trained language models. In *EMNLP*.

Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

M. Marelli, S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, and Roberto Zamparelli. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *LREC*.

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Jeffrey Pennington, R. Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, W. Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP/IJCNLP*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

K. Song, Xu Tan, Tao Qin, Jianfeng Lu, and T. Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. *ArXiv*, abs/2004.09297.

Jianlin Su, Jiarun Cao, Weijie Liu, and Yangyiwen Ou. 2021. Whitening sentence representations for better semantics and faster retrieval.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. In *ACL*.

Nandan Thakur, N. Reimers, Johannes Daxenberger, and Iryna Gurevych. 2020. Augmented sbert: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. *ArXiv*, abs/2010.08240.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ArXiv*, abs/1706.03762.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. ICLR.

Bin Wang and C.-C. Jay Kuo. 2020. Sbert-wk: A sentence embedding method by dissecting bert-based word models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2146–2157.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and M. Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *ArXiv*, abs/2002.10957.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and M. Zhou. 2020. Layoutlm: Pre-training of text and layout for document image understanding. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Ziyi Yang, Chenguang Zhu, and Weizhu Chen. 2019. Parameter-free sentence embedding via orthogonal basis. In *EMNLP/IJCNLP*.

Yan Zhang, Ruidan He, Zuozhu Liu, Kwan Hui Lim, and Lidong Bing. 2020. An unsupervised sentence embedding method bymutual information maximization. In *EMNLP*.

A Appendix

A.1 More Results of WhiteningBERT

To further illustrate the effectiveness of the whitening algorithm in induce sentence embeddings for STS tasks, we experiment with more PLMs and report their performance with and without incorporating the whitening algorithm. From the results exhibited in Table 2, we find that no matter which PLM we use, the average performance on 7 STS tasks improves after incorporating the whitening strategy. This result again verifies the effectiveness of whitening in producing sentence embeddings.

A.2 Comparison with GPT-3

GPT-3 (Brown et al., 2020) is a powerful language model that is capable of sophisticated natural language understanding of tasks like classification in a zero-shot fashion. Here we report the results

PLM	STSB	SICK	STS-12	STS-13	STS-14	STS-15	STS-16	Average
BERT-base (Devlin et al., 2019)	59.05 → 68.72	63.75 → 60.43	57.72 → 62.20	58.38 → 68.52	61.97 → 67.35	70.28 → 74.73	69.63 → 72.42	62.97 → 67.77 (+4.80)
RoBERTa-base (Liu et al., 2019)	57.54 → 68.18	60.75 → 58.80	58.56 → 62.21	50.37 → 67.13	59.62 → 67.63	66.64 → 74.78	63.21 → 71.43	59.53 → 67.17 (+7.64)
SpanBERT-base (Joshi et al., 2019)	59.10 → 69.82	60.28 → 58.48	58.27 → 63.16	54.27 → 69.00	61.37 → 68.71	67.84 → 75.37	66.54 → 73.24	61.10 → 68.25 (+7.16)
DeBERTa-base (He et al., 2020)	56.55 → 67.60	61.66 → 59.38	57.55 → 62.54	54.78 → 67.62	61.43 → 66.76	68.84 → 74.97	67.51 → 71.13	61.19 → 67.14 (+5.95)
ALBERT-base (Lan et al., 2020)	46.18 → 61.76	54.99 → 58.03	51.02 → 58.33	43.94 → 62.89	50.79 → 59.92	60.83 → 68.84	55.35 → 65.90	51.87 → 62.24 (+10.37)
T5-base (Raffel et al., 2020)	42.39 → 68.32	51.85 → 56.13	46.38 → 61.92	42.15 → 68.50	49.75 → 67.94	58.22 → 74.88	55.09 → 72.90	49.41 → 67.23 (+17.82)
LayoutLM-base (Xu et al., 2020)	25.14 → 61.77	38.99 → 56.50	33.22 → 58.33	19.63 → 59.63	26.19 → 63.41	31.50 → 69.65	30.16 → 65.90	29.26 → 62.17 (+32.91)
XLM-base (Lample and Conneau, 2019)	54.47 → 69.51	54.65 → 55.54	54.52 → 62.26	43.15 → 66.46	56.50 → 69.41	61.10 → 75.09	57.30 → 73.95	54.53 → 67.46 (+12.93)
DistilBERT (Sanh et al., 2019)	61.45 → 69.41	63.84 → 59.43	59.68 → 61.82	59.50 → 66.90	63.54 → 67.69	70.95 → 74.27	69.90 → 72.81	64.12 → 67.48 (+3.35)
M-BERT (Devlin et al., 2019)	57.67 → 69.09	58.60 → 56.85	58.71 → 61.13	53.14 → 65.74	61.72 → 67.18	68.78 → 73.64	67.09 → 72.53	60.82 → 66.60 (+5.78)
MPNet (Song et al., 2020)	58.58 → 69.30	62.22 → 59.58	58.21 → 62.18	53.93 → 68.99	60.78 → 67.76	67.26 → 75.51	63.05 → 71.62	60.58 → 67.85 (+7.27)
SqueezeBERT (Iandola et al., 2020)	54.86 → 67.80	60.57 → 58.43	56.36 → 61.43	53.05 → 64.57	60.59 → 66.96	67.81 → 73.57	64.68 → 71.24	59.70 → 66.29 (+6.58)
LaBSE (Feng et al., 2020)	72.56 → 73.32	68.36 → 63.27	68.29 → 68.45	65.75 → 71.11	71.41 → 71.66	78.90 → 79.30	75.68 → 74.87	71.56 → 71.71 (+0.15)
SPECTER (Cohan et al., 2020)	62.37 → 68.90	57.37 → 56.42	62.91 → 63.62	52.93 → 67.43	62.77 → 68.82	67.76 → 74.47	66.81 → 71.04	61.85 → 67.24 (+5.40)
MiniLM (Wang et al., 2020)	50.59 → 67.91	58.40 → 59.79	55.21 → 60.32	44.92 → 65.00	54.44 → 66.35	64.27 → 73.79	59.27 → 72.38	55.30 → 66.51 (+11.21)
BERT-large (Devlin et al., 2019)	59.13 → 69.81	60.38 → 59.62	58.13 → 62.92	57.70 → 69.49	60.19 → 67.19	66.89 → 74.45	70.07 → 73.67	61.78 → 68.16 (+6.38)
RoBERTa-large (Liu et al., 2019)	60.43 → 69.44	59.13 → 57.33	58.78 → 61.66	54.31 → 67.02	61.10 → 68.21	66.40 → 75.81	65.28 → 73.29	60.78 → 67.54 (+6.76)
SpanBERT-large (Joshi et al., 2019)	59.51 → 70.06	61.10 → 58.53	60.85 → 63.46	58.36 → 71.17	63.24 → 69.09	70.43 → 75.40	68.24 → 73.70	63.10 → 68.77 (+5.67)
DeBERTa-large (He et al., 2020)	57.98 → 70.28	62.13 → 59.11	58.50 → 63.48	55.20 → 70.10	62.04 → 69.10	70.24 → 76.76	68.57 → 74.56	62.09 → 69.06 (+6.96)
ALBERT-large (Lan et al., 2020)	50.49 → 63.45	57.16 → 57.98	55.01 → 60.29	49.44 → 63.15	53.73 → 60.81	65.02 → 70.16	60.71 → 66.37	55.94 → 63.17 (+7.24)
T5-large (Raffel et al., 2020)	35.57 → 69.16	40.31 → 55.75	37.83 → 62.33	29.33 → 70.70	39.63 → 68.41	45.72 → 74.82	47.52 → 72.01	39.42 → 71.71 (+28.18)
LayoutLM-large (Xu et al., 2020)	45.04 → 68.16	49.94 → 56.32	49.48 → 59.50	32.83 → 64.28	42.65 → 67.60	47.77 → 73.14	49.10 → 71.81	45.26 → 65.83 (+20.57)
XLM-large (Lample and Conneau, 2019)	56.76 → 70.04	56.34 → 55.06	57.35 → 61.53	46.84 → 66.08	60.38 → 69.63	64.41 → 75.38	61.18 → 73.89	57.61 → 67.37 (+9.76)
DialogRPT (Gao et al., 2020)	52.92 → 69.08	54.65 → 55.16	56.93 → 62.75	43.37 → 67.06	51.27 → 67.88	55.72 → 75.44	56.25 → 72.44	53.02 → 67.12 (+14.10)

Table 2: Experimental results of WhiteningBERT with different PLMs without (to the left of the arrow) or with (to the right of the arrow) whitening strategy. We report the Spearman's rank correlation coefficient ($\rho \times 100$) between similarity scores assigned by sentence embeddings and humans. The embeddings are produced by averaging tokens representations (*token*=AVG) and combining layer one and the last layer (*layer*=L1 + L12(L24 or L6)). The average performance improves after incorporating the whitening algorithm.

Model	Accuracy	# Param
GPT-3 (125M)	47.7	125M
GPT-3 (350M)	49.8	350M
GPT-3 (760M)	48.4	760M
GPT-3 (1.3B)	56.0	1.3B
GPT-3 (2.7B)	46.6	2.7B
GPT-3 (6.7B)	55.2	6.7B
GPT-3 (13B)	62.8	13B
GPT-3 (175B)	63.5	175B
whiteningBERT (PLM=BERT)	52.7	110M

Table 3: Experiment results on RTE. The embeddings are produced by averaging tokens representations (*token*=AVG), combining layer one and the last layer (*layer*=L1 + L12), and incorporating whitening *whitening*=T.

of whiteningBERT (PLM=BERT) on RTE dev set (Wang et al., 2019). Specifically, we first compute the cosine similarity of the two sentence embeddings and then manually set a threshold of 0.5 to predict the label of each sentence pairs. The results are shown in Table 3.

A.3 Code for Whitening

Figure 3 displays the source code for whitening algorithm in PyTorch (Paszke et al., 2019).

```python
def whitening_torch(embeddings):
    mu = torch.mean(embeddings, dim=0, keepdim=True)
    cov = torch.mm((embeddings - mu).t(), embeddings - mu)
    u, s, vt = torch.svd(cov)
    W = torch.mm(u, torch.diag(1/torch.sqrt(s)))
    embeddings = torch.mm(embeddings - mu, W)
    return embeddings
```

Figure 3: Pytorch code for whitening strategy.