WhiteningBERT: An Easy Unsupervised Sentence Embedding Approach

Junjie Huang1, Duyu Tang4, Wanjun Zhong2, Shuai Lu3, Linjun Shou5, Ming Gong6, Daxin Jiang6, Nan Duan1
1Beihang University 2Sun Yat-sen University 3Peking University
4Microsoft Research Asia 5Microsoft 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 ($\S A.3$), 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 $S = \{s_1, s_2, ..., s_n\}$, a transformer-based PLM produces a set of hidden representations $H^{(0)}, H^{(1)}, ..., H^{(L)}$, where $H^{(l)} = [h_1^{(l)}, h_2^{(l)}, ..., h_n^{(l)}]$ 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...
Whitening is a linear transformation that transforms word embedding mapping (Artetxe et al., 2018) as in Eq. 1 such that $\hat{E} \in \mathbb{R}^{N \times d}$ is the whitened sentence embeddings,

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

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

### 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., 2012), 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
which indicates that single token embeddings and humans.

Therefore, we suggest averaging the last layer and layer one and the last layer performs better than 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 obtaining 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

<table>
<thead>
<tr>
<th>Models</th>
<th>STSB</th>
<th>SICK</th>
<th>STS-12</th>
<th>STS-13</th>
<th>STS-14</th>
<th>STS-15</th>
<th>STS-16</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baselines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. GloVe (Reimers and Gurevych, 2019)</td>
<td>58.02</td>
<td>53.76</td>
<td>55.14</td>
<td>70.66</td>
<td>59.73</td>
<td>68.25</td>
<td>63.66</td>
<td>61.32</td>
</tr>
<tr>
<td>SIF (GloVe-WR) (Arora et al., 2017)</td>
<td>-</td>
<td>-</td>
<td>56.20</td>
<td>56.60</td>
<td>68.50</td>
<td>71.70</td>
<td>-</td>
<td>63.25</td>
</tr>
<tr>
<td>IS-BERT-NLI (Zhang et al., 2020)</td>
<td>69.21</td>
<td>64.25</td>
<td>57.71</td>
<td>69.24</td>
<td>61.21</td>
<td>75.23</td>
<td>70.16</td>
<td>66.58</td>
</tr>
<tr>
<td>BERT-flow (NLI) (Li et al., 2020)</td>
<td>58.56</td>
<td>65.44</td>
<td>59.54</td>
<td>64.69</td>
<td>64.66</td>
<td>72.92</td>
<td>71.84</td>
<td>65.38</td>
</tr>
<tr>
<td>SBERT WK (BERT) (Wang and Kuo, 2020)</td>
<td>16.07</td>
<td>41.54</td>
<td>26.66</td>
<td>14.74</td>
<td>24.32</td>
<td>28.84</td>
<td>34.37</td>
<td>26.65</td>
</tr>
</tbody>
</table>

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.*
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 perform best among all PLMs, but could be satisfying 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 incorporate 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 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**

We thank Daya Guo for helpful discussions. We thank the anonymous reviewers for valuable comments.

**References**


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.


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


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


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.


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.


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


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


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, Zuo Zhu Liu, Kwan Hui Lim, and Lidong Bing. 2020. An unsupervised sentence embedding method by mutual 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
def whitening_torch(embeddings):
    mu = torch.mean(embeddings, dim=0, keepdim=True)
    cov = torch.mm((embeddings - mu).t(), embeddings - mu)
    u, s = torch.svd(cov)
    W = torch.mm(u, torch.diag(1/torch.sqrt(s)))
    embeddings = torch.mm(embeddings - mu, W)
    return embeddings

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 \( \rho \) between similarity scores assigned by sentence embeddings and the embeddings are produced by averaging tokens representations \( \text{tokens}(\text{AVG}) \) and combining layer one and the last layer \( (layer=1 + L12 (L24 or L6)) \). The average performance improves after incorporating the whitening algorithm.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th># Param</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-3 (125M)</td>
<td>47.7</td>
<td>125M</td>
</tr>
<tr>
<td>GPT-3 (350M)</td>
<td>49.8</td>
<td>350M</td>
</tr>
<tr>
<td>GPT-3 (760M)</td>
<td>48.4</td>
<td>760M</td>
</tr>
<tr>
<td>GPT-3 (1.3B)</td>
<td>56.0</td>
<td>1.3B</td>
</tr>
<tr>
<td>GPT-3 (2.7B)</td>
<td>46.6</td>
<td>2.7B</td>
</tr>
<tr>
<td>GPT-3 (6.7B)</td>
<td>55.2</td>
<td>6.7B</td>
</tr>
<tr>
<td>GPT-3 (13B)</td>
<td>62.8</td>
<td>13B</td>
</tr>
<tr>
<td>GPT-3 (175B)</td>
<td>63.5</td>
<td>175B</td>
</tr>
<tr>
<td>whiteningBERT (PLM=BERT)</td>
<td>52.7</td>
<td>110M</td>
</tr>
</tbody>
</table>

Table 3: Experiment results on RTE. The embeddings are produced by averaging tokens representations \( \text{tokens}(\text{AVG}) \), combining layer one and the last layer \( (layer=1 + L12) \), and incorporating whitening \( \text{whitening}=\text{T} \).

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).