ConSERT: A Contrastive Framework for Self-Supervised Sentence Representation Transfer

Authors

- Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, Weiran Xu
  - Beijing University of Posts and Telecommunications
  - Meituan Inc., Beijing, China

Introduction

- Sentence representation (SR) plays a key role in many NLP tasks
  - Eg. Semantic Similarity, Information Retrieval

SNLI	train dev test	550,152 10,000 10,000	s_a: *Two men on bicycles competing in a race.* s_b: *Men are riding bicycles on the street.*	**entailment** neutral contradict
STS-2014	train dev test	7,592 - 3,750	s_a: *Then perhaps we could have avoided a catastrophe.* s_b: *Then we might have been able to avoid a disaster.*	score $[0, 5]$ **4.6**

- BERT is a *de facto* when tacking NLP tasks but its SR ability is somewhat *suspicious*
  1. SBERT (Reimers and Gurevych, 2019) shows two BERT embeddings (Average- and CLS-pooling) are worse than Glove emebedding on Semantic Textual Similarity (STS) tasks

Model	STS12	STS13	STS14	STS15	STS16	STSb	SICK-R	Avg.
Avg. GloVe embeddings	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
Avg. BERT embeddings	38.78	57.98	57.98	63.15	61.06	46.35	58.40	54.81
BERT CLS-vector	20.16	30.01	20.09	36.88	38.08	16.50	42.63	29.19
InferSent - Glove	52.86	66.75	62.15	72.77	66.87	68.03	65.65	65.01
Universal Sentence Encoder	64.49	67.80	64.61	76.83	73.18	74.92	**76.69**	71.22
SBERT-NLI-base	70.97	76.53	73.19	79.09	74.30	77.03	72.91	74.89
SBERT-NLI-large	72.27	**78.46**	**74.90**	80.99	76.25	**79.23**	73.75	76.55
SRoBERTa-NLI-base	71.54	72.49	70.80	78.74	73.69	77.77	74.46	74.21
SRoBERTa-NLI-large	**74.53**	77.00	73.18	**81.85**	**76.82**	79.10	74.29	**76.68**

Table 1: Spearman rank correlation ρ between the cosine similarity of sentence representations and the gold labels for various Textual Similarity (STS) tasks. Performance is reported by convention as $\rho \times 100$. STS12-STS16: SemEval 2012-2016, STSb: STSbenchmark, SICK-R: SICK relatedness dataset.

2. When directly adopt BERT to STS, almost all pairs of sentences achieve a similarity scores between 0.6 ~ 1.0, even if some pairs are regarded as completely unrelated by human annotators → SR from BERT are ***collapsed***

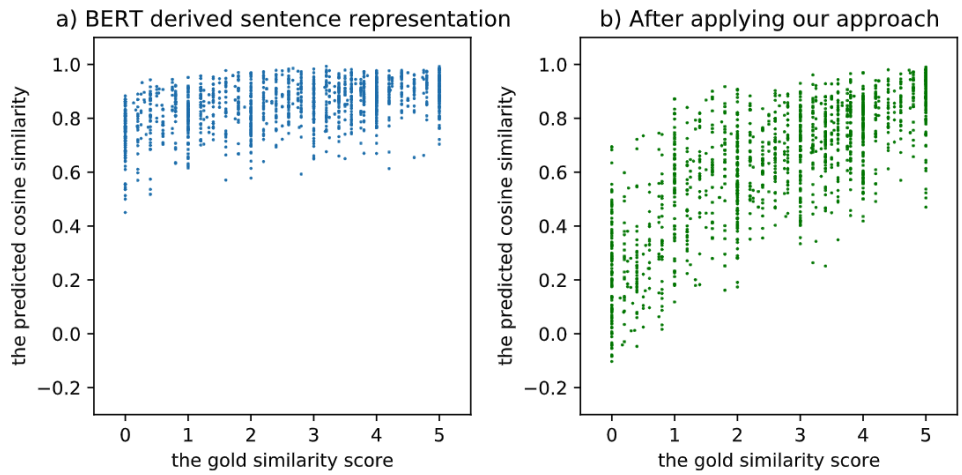


Figure 1: The correlation diagram between the gold similarity score (x-axis) and the model predicted cosine similarity score (y-axis) on the STS benchmark dataset.

3. Previous works (Gao et al., 2019; Li et al., 2020) find that the word representation space of BERT is *anisotropic*
   - High-frequency words are clustered and close to the origin
   - Low-frequency words disperse sparsely

Rank of word frequency	$(0, 100)$	$[100, 500)$	$[500, 5\text{K})$	$[5\text{K}, 1\text{K})$
Mean ℓ_2-norm	0.95	1.04	1.22	1.45
Mean k-NN ℓ_2-dist. ($k = 3$)	0.77	0.93	1.16	1.30
Mean k-NN ℓ_2-dist. ($k = 5$)	0.83	0.99	1.22	1.34
Mean k-NN ℓ_2-dist. ($k = 7$)	0.87	1.04	1.26	1.37
Mean k-NN dot-product. ($k = 3$)	0.73	0.92	1.20	1.63
Mean k-NN dot-product. ($k = 5$)	0.73	0.91	1.19	1.61
Mean k-NN dot-product. ($k = 7$)	0.72	0.90	1.17	1.60

Table 1: The mean ℓ_2-norm, as well as their distance to their k-nearest neighbors (among all the word embeddings) of the word embeddings of BERT, segmented by ranges of word frequency rank (counted based on Wikipedia dump; the smaller the more frequent).

    - When averaging token embeddings, those high-frequency words dominate the sentence representations
- As a result, it is *inappropriate* to directly apply BERT's native SR for semantic matching or text retrieval

ConSERT

Approach

- Given a BERT-like pre-trained language model M and an unsupervised dataset D drawn from the target distribution, fine-tune M on D → make SR more task relevant

Framework

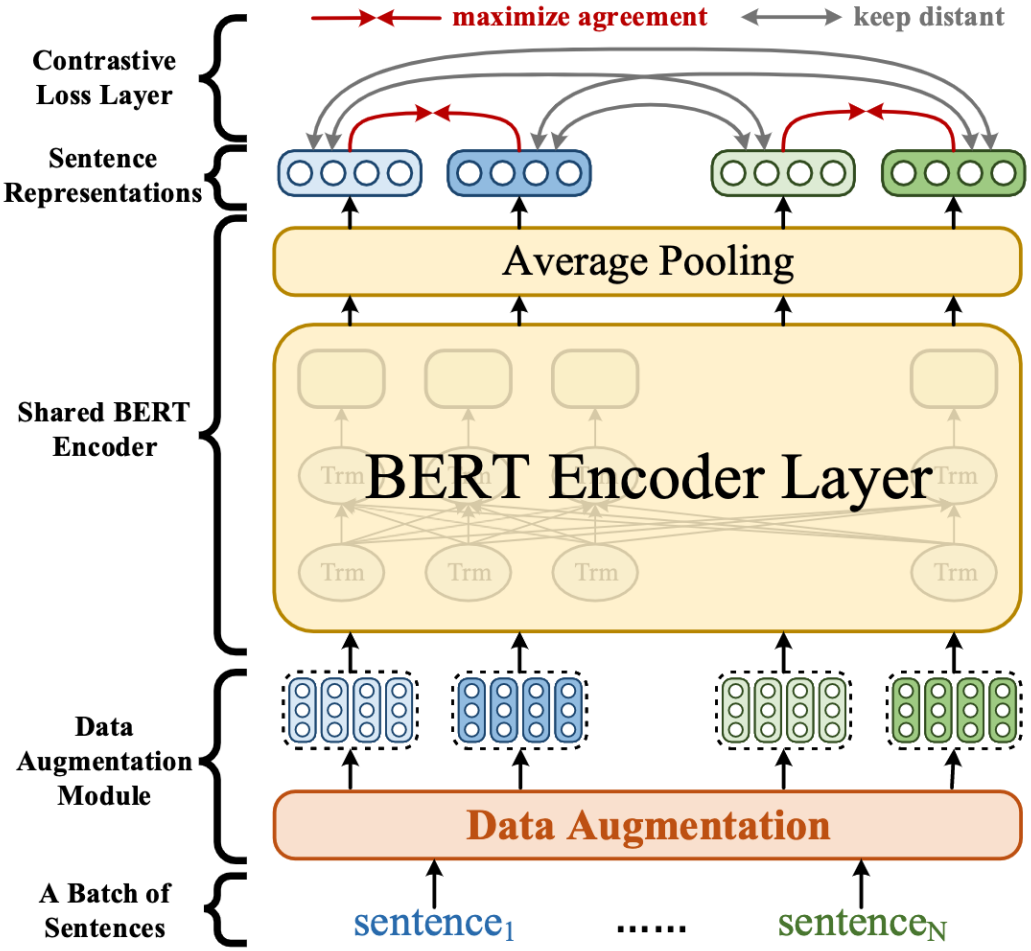


Figure 2: The general framework of our proposed approach.

1. A data augmentation module that generates different views of input samples at the token embedding layer
2. A BERT encoder computes SR for each input text
    - During training, use average pooling of the token embeddings at the last layer to obtain SR
    - During evaluation, use average pooling of the token embeddings at the last two layers to obtain SR
3. Constrastive loss layer on top of BERT encoder
    - Maximize agreement between one representation and its corresponding version (i.e. from augmentation)
    - Minimize agreement with other, $2(N-1)$, SR in the same batch

Following Chen et al. (2020a), we adopt the normalized temperature-scaled cross-entropy loss (NT-Xent) as the contrastive objective. During each training step, we randomly sample N texts from $\mathcal{D}$ to construct a mini-batch, resulting in $2N$ representations after augmentation. Each data point is trained to find out its counterpart among $2(N-1)$ in-batch negative samples:

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(r_i, r_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(r_i, r_k)/\tau)} \tag{1}$$

, where $\text{sim}(\cdot)$ indicates the cosine similarity function, τ controls the temperature and $\mathbb{1}$ is the indicator. Finally, we average all $2N$ in-batch classification losses to obtain the final contrastive loss $\mathcal{L}_{\text{con}}$.

Data Augmentation

- SimCLR: Image augmentation

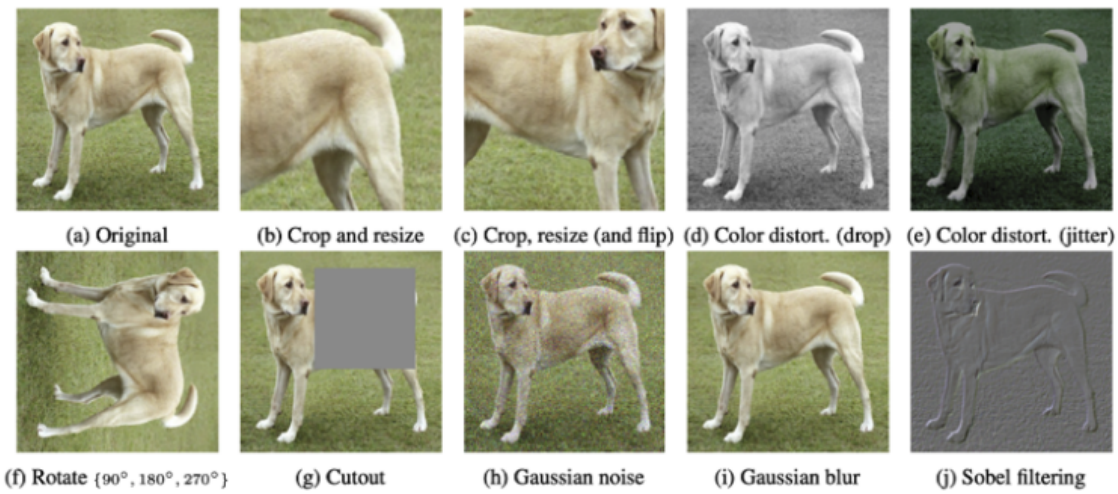


Figure 4. Illustrations of the studied data augmentation operators.

- ConSERT: Token-level augmentation

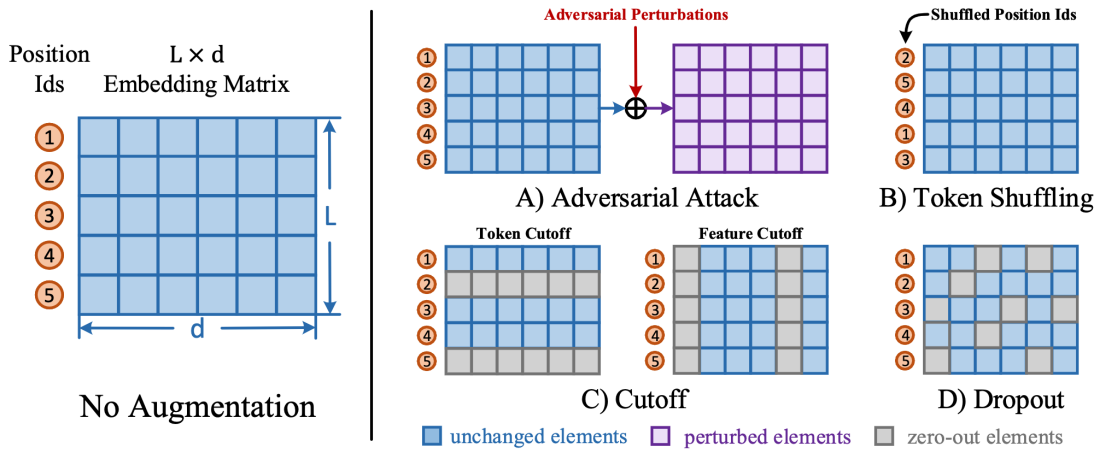


Figure 3: The four data augmentation strategies used in our experiments.

1. Adversarial Attack
    - Implement with Fast Gradient Value → directly uses the gradient to compute the perturbation → $\tilde{x} = x + \eta$ where $\eta = \nabla_x J_\theta(x, l)$
    - ONLY applicable when training with supervision since it relies on supervised loss
2. Token Shuffling
    - Randomly shuffle the order of the tokens in the input sequences
3. Cutoff
    - Idea proposed from Shen et. (2020)
    - Randomly erase some tokens (token-cutoff, $p = 0.15$) and features dimensions (feature-cutoff, $p = 0.2$)
4. Dropout
    - Randomly drop elements in the token embedding layer by a specific probability (0.2) and set their values to zero
    - Remove dropout layer in BERT encoder

Supervised Training

- In addition to above unsupervised transfer, ConSERT can incorporate supervised learning

- NLI Task (a sentence pair classification)
- Classification Objective

  $f = Concat(r_1, r_2, |r_1 - r_2|)$

  $L_{CE} = CrossEntropy(Wf + b, y)$ where r_1 and r_2 denote two SR

- Different Training Strategies

  - **Joint training (joint)** We jointly train the model with the supervised and unsupervised objectives $\mathcal{L}_{\text{joint}} = \mathcal{L}_{\text{ce}} + \alpha\mathcal{L}_{\text{con}}$ on NLI dataset. α is a hyper-parameter to balance two objectives.
  - **Supervised training then unsupervised transfer (sup-unsup)** We first train the model with $\mathcal{L}_{\text{ce}}$ on NLI dataset, then use $\mathcal{L}_{\text{con}}$ to fine-tune it on the target dataset.
  - **Joint training then unsupervised transfer (joint-unsup)** We first train the model with the $\mathcal{L}_{\text{joint}}$ on NLI dataset, then use $\mathcal{L}_{\text{con}}$ to fine-tune it on the target dataset.

Experiment

Baseline

▼ BERT-flow

- Proposes a flow-based approach that maps BERT embeddings to a standard Gaussian latent space, where embeddings are more suitable for comparison

▼ BERT-CT

- Dual-encoder with maximizing agreement with identical sentence and minimizing with random sample

4 METHOD

To counter the negative trend found in Section 3, where the lacking STS performance of the sentence representations in the final layers became apparent, we define a training objective meant to encourage the model to retain a semantically distinguishable sentence representation until the final layer. We name this method *Contrastive Tension* (CT), where two independent models, with identically initialized weights, are set to maximise the dot product between their sentence representations for identical sentences, and minimize the dot product for their sentence representations of differing sentences. Hence, the CT objective is defined as:

$$z = f_1(s_1)^T \cdot f_2(s_2)$$
$$\mathcal{L}(z, s_1, s_2) = \begin{cases} -log\ \sigma(z) & \text{if } s_1 = s_2 \\ -log\ \sigma(1-z) & \text{if } s_1 \neq s_2 \end{cases} \quad (1)$$

Where f_1 and f_2 are two independently parameterized models that given a sentence s produces a fixed size vector representation and where σ refers to the Logistic function.

▼ CLEVER

- Dual-encoder

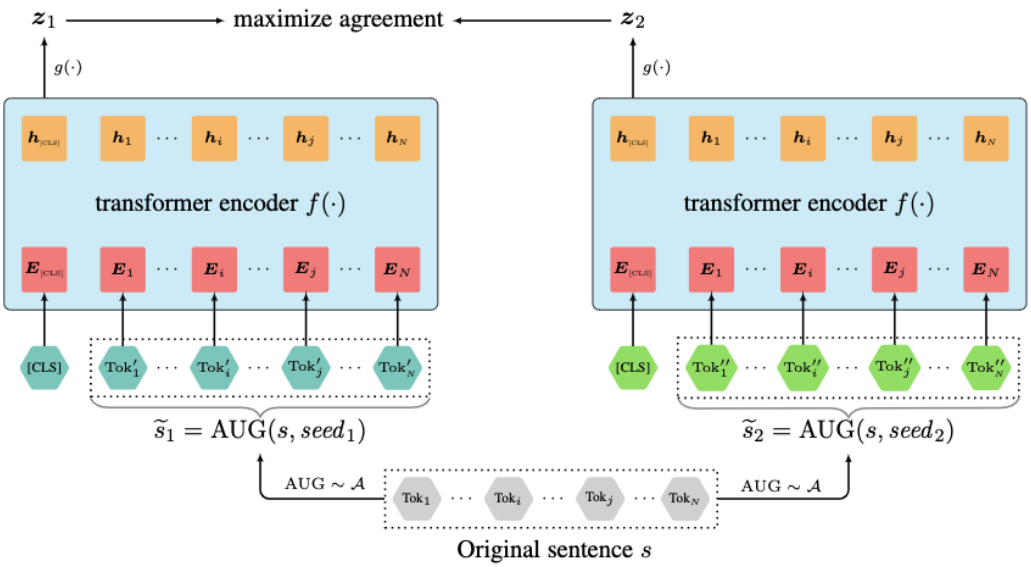
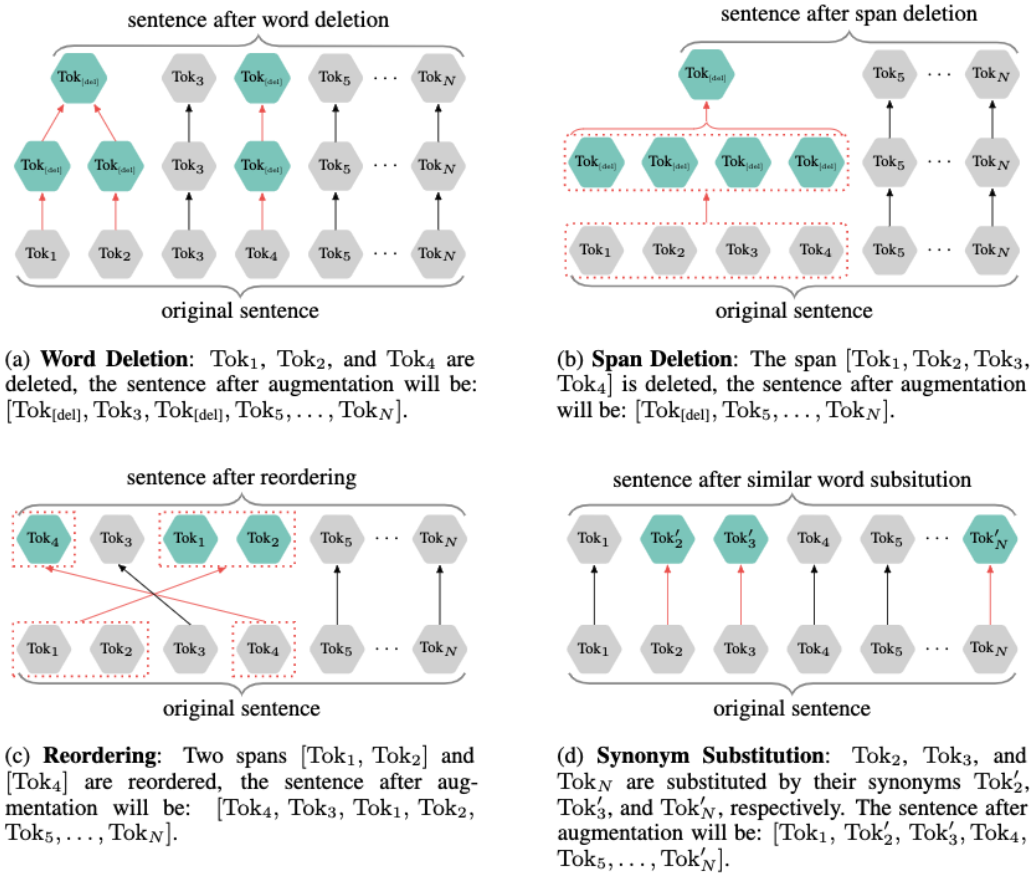


Figure 1: The proposed contrastive learning framework CLEAR.

- Various augmentation



(a) **Word Deletion**: Tok_1, Tok_2, and Tok_4 are deleted, the sentence after augmentation will be: $[\text{Tok}_{[\text{del}]}, \text{Tok}_3, \text{Tok}_{[\text{del}]}, \text{Tok}_5, \ldots, \text{Tok}_N]$.

(b) **Span Deletion**: The span $[\text{Tok}_1, \text{Tok}_2, \text{Tok}_3, \text{Tok}_4]$ is deleted, the sentence after augmentation will be: $[\text{Tok}_{[\text{del}]}, \text{Tok}_5, \ldots, \text{Tok}_N]$.

(c) **Reordering**: Two spans $[\text{Tok}_1, \text{Tok}_2]$ and $[\text{Tok}_4]$ are reordered, the sentence after augmentation will be: $[\text{Tok}_4, \text{Tok}_3, \text{Tok}_1, \text{Tok}_2, \text{Tok}_5, \ldots, \text{Tok}_N]$.

(d) **Synonym Substitution**: Tok_2, Tok_3, and Tok_N are substituted by their synonyms Tok'_2, Tok'_3, and Tok'_N, respectively. The sentence after augmentation will be: $[\text{Tok}_1, \text{Tok}'_2, \text{Tok}'_3, \text{Tok}_4, \text{Tok}_5, \ldots, \text{Tok}'_N]$.

Figure 2: Four sentence augmentation methods in proposed contrastive learning framework CLEAR.

Result

- Unsupervision

Method	STS12	STS13	STS14	STS15	STS16	STSb	SICK-R	Avg.
Unsupervised baselines								
Avg. GloVe embeddings†	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
$\text{BERT}_{\text{base}}$‡	35.20	59.53	49.37	63.39	62.73	48.18	58.60	53.86
$\text{BERT}_{\text{large}}$‡	33.06	57.64	47.95	55.83	62.42	49.66	53.87	51.49
$\text{CLEAR}_{\text{base}}$†	49.0	48.9	57.4	63.6	65.6	75.6	72.5	61.8
$\text{IS-BERT}_{\text{base}}$-NLI†	56.77	69.24	61.21	75.23	70.16	69.21	64.25	66.58
$\text{BERT}_{\text{base}}$-CT†	66.86	70.91	72.37	78.55	77.78	-	-	-
$\text{BERT}_{\text{large}}$-CT†	69.50	75.97	74.22	78.83	78.92	-	-	-
Using STS unlabeled texts								
$\text{BERT}_{\text{base}}$-flow†	63.48	72.14	68.42	73.77	75.37	70.72	63.11	69.57
$\text{BERT}_{\text{large}}$-flow†	65.20	73.39	69.42	74.92	**77.63**	72.26	62.50	70.76
$\text{ConSERT}_{\text{base}}$‡	64.64	78.49	69.07	79.72	75.95	73.97	67.31	72.74
$\text{ConSERT}_{\text{large}}$‡	**70.69**	**82.96**	**74.13**	**82.78**	76.66	**77.53**	**70.37**	**76.45**

Table 2: The performance comparison of ConSERT with other methods in an *unsupervised* setting. We report the spearman correlation $\rho \times 100$ on 7 STS datasets. Methods with † indicate that we directly report the scores from the corresponding paper, while methods with ‡ indicate our implementation.

    - $ConSERT_{large}$ outperforms some supervised models such InferSent, Universal Sentence and keeps comparable to $SBERT_{large}$
- Supervision

Method	STS12	STS13	STS14	STS15	STS16	STSb	SICK-R	Avg.
Using NLI supervision								
InferSent - GloVe†	52.86	66.75	62.15	72.77	66.87	68.03	65.65	65.01
Universal Sentence Encoder†	64.49	67.80	64.61	76.83	73.18	74.92	76.69	71.22
$SBERT_{base}$-NLI†	70.97	76.53	73.19	79.09	74.30	77.03	72.91	74.89
$SBERT_{large}$-NLI†	72.27	78.46	74.90	80.99	76.25	79.23	73.75	76.55
$SBERT_{base}$-NLI (re-impl.)‡	69.89	75.77	72.36	78.51	73.67	76.75	72.76	74.24
$SBERT_{large}$-NLI (re-impl.)‡	72.69	78.77	75.13	80.95	76.89	79.53	73.25	76.74
$BERT_{base}$-CT†	68.80	74.58	76.62	79.72	77.14	-	-	-
$BERT_{large}$-CT†	69.80	75.45	76.47	81.34	78.11	-	-	-
$ConSERT_{base}$ *joint*‡	70.53	79.96	74.85	81.45	76.72	78.82	77.53	77.12
$ConSERT_{large}$ *joint*‡	**73.26**	**82.36**	**77.73**	**83.84**	**78.75**	**81.54**	**78.64**	**79.44**
Using NLI supervision and STS unlabeled texts								
$BERT_{base}$-flow†	68.95	78.48	77.62	81.95	78.94	81.03	74.97	77.42
$BERT_{large}$-flow†	70.19	80.27	78.85	82.97	**80.57**	81.18	74.52	78.36
$ConSERT_{base}$ *sup-unsup*‡	73.51	84.86	77.44	83.11	77.98	81.80	74.29	79.00
$ConSERT_{large}$ *sup-unsup*‡	75.26	**86.01**	79.00	83.88	79.45	82.95	76.54	80.44
$ConSERT_{base}$ *joint-unsup*‡	74.07	83.93	77.05	83.66	78.76	81.36	76.77	79.37
$ConSERT_{large}$ *joint-unsup*‡	**77.47**	85.45	**79.41**	**85.59**	80.39	**83.42**	**77.26**	**81.28**

Table 3: The performance comparison of ConSERT with other methods in a *supervised* setting. We report the spearman correlation $\rho \times 100$ on 7 STS datasets. Methods with † indicate that we directly report the scores from the corresponding paper, while methods with ‡ indicate our implementation.

Qualitative Analysis

BERT Embedding Space

- Hypothesis: The *collapse* issue is mainly due to the *anisotropic* space that is sensitive to the token frequency
- Experiment: Mask the embeddings of several most frequent tokens

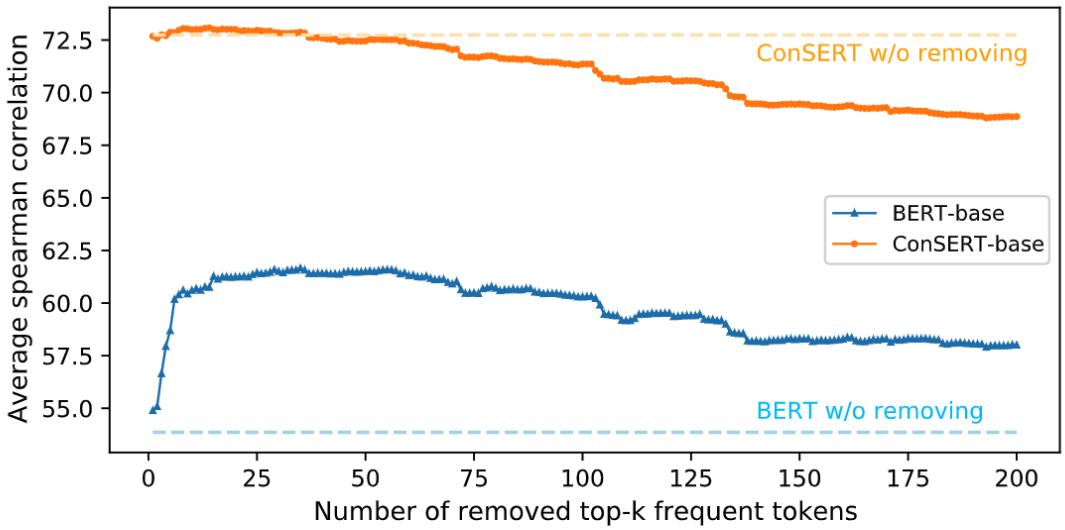


Figure 4: The average spearman correlation on STS tasks w.r.t. the number of removed top-k frequent tokens. Note that we also considered the [CLS] and [SEP] tokens and they are the 2 most frequent tokens. The frequency of each token is calculated through the test split of the STS Benchmark dataset.

- BERT

  Removing frequent tokens increases performance because the dominance of SR by high-frequent tokens vanishes

- ConSERT

  Removing frequent tokens does not increase much (0.3)

  This shows that ConSERT reshapes the BERT's original embedding space, reducing the influence of common tokens on SR

Effect of Augmentation Strategies

- Shuffle and Token Cutoff most effect
- None-None is better than BERT (63.84 > 53.86) → In-batch negative works !

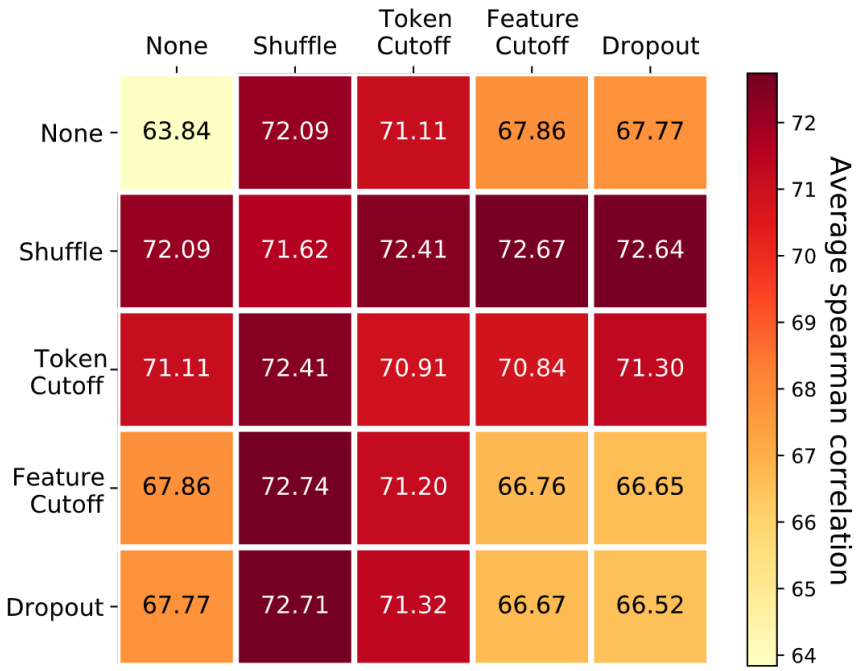


Figure 5: The performance visualization with different combinations of data augmentation strategies. The row indicates the 1st data augmentation strategy while the column indicates the 2nd data augmentation strategy.

Influence of Temperature

- The temperature τ controls the smoothness of the distribution
- SR ability is sensitive to the temperature

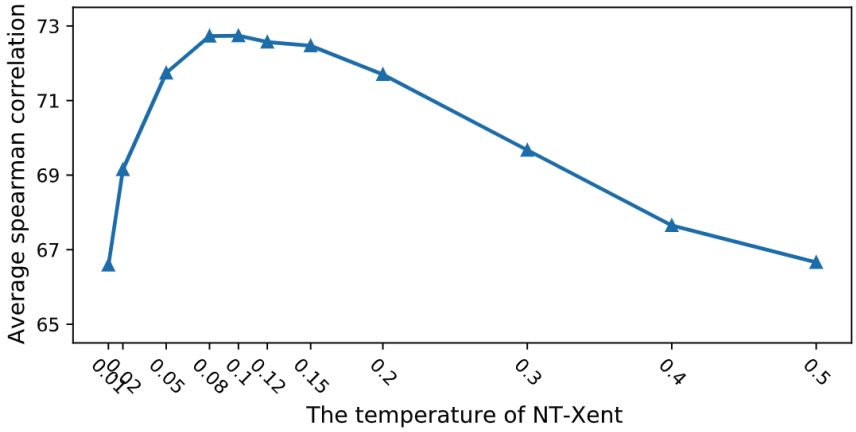


Figure 7: The influence of different temperatures in NT-Xent. The best performance is achieved when the temperature is set to 0.1.

Influence of Batch Size

- Large batch size is helpful as in SimCLR but not significant

Batch Size	16	48	96	192	288
Avg. Spearman	72.63	72.60	72.74	72.86	72.98
Number of Steps	6175	2459	1530	930	620

Table 4: The average spearman correlation as well as the training steps of our unsupervised approach with different batch sizes.

▼ SimCLR

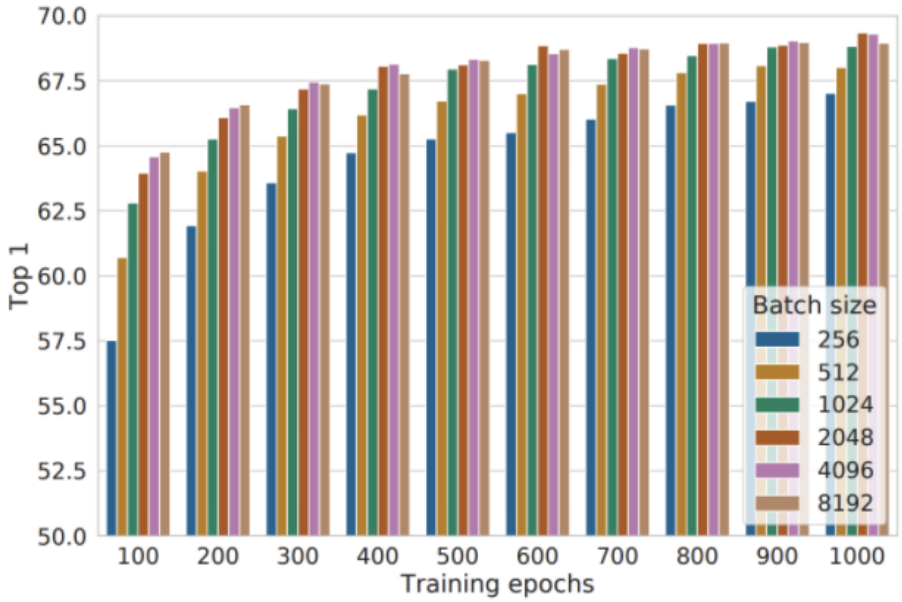


Figure 9. Linear evaluation models (ResNet-50) trained with different batch size and epochs.

Each bar is a single run from scratch.