Exploring the Impact of Negative Samples of Contrastive Learning: A Case Study of Sentence Embedding

Rui Cao *and* ***Yihao Wang*** *and* ***Yuxin Liang*** *and* ***Ling Gao*** *and* ***Jie Zheng*** *and* ***Jie Ren*** *and* ***Zheng Wang***
{ caorui, wangyihao, liangyuxin }@stumail.nwu.edu.cn, { gl, jzheng }@nwu.edu.cn, renjie@snnu.edu.cn, z.wang5@leeds.ac.uk

Background and Motivation

Contrastive learning is emerging as a powerful technique for extracting knowledge from unlabeled data. This technique requires a balanced mixture of two ingredients: positive (similar) and negative (dissimilar) samples. This is typically achieved by maintaining a queue of negative samples during training. Prior works in the area typically uses a fixed-length negative sample queue, but how the negative sample size affects the model performance remains unclear. The opaque impact of the number of negative samples on performance when employing contrastive learning aroused our in-depth exploration.

Can we apply MoCo-style contrastive structure on unsupervised textual tasks?

If could:

1. How to boost performance?
2. How to pervent model from clasping?
3. How much negative sample do textual task needs?

This paper presents a momentum contrastive learning model with negative sample queue for sentence embedding, namely MoCoSE. We add the prediction layer to the online branch to make the model asymmetric and together with EMA update mechanism of the target branch to prevent the model from collapsing. We define a maximum traceable distance metric, through which we learn to what extent the text contrastive learning benefits from the historical information of negative samples.

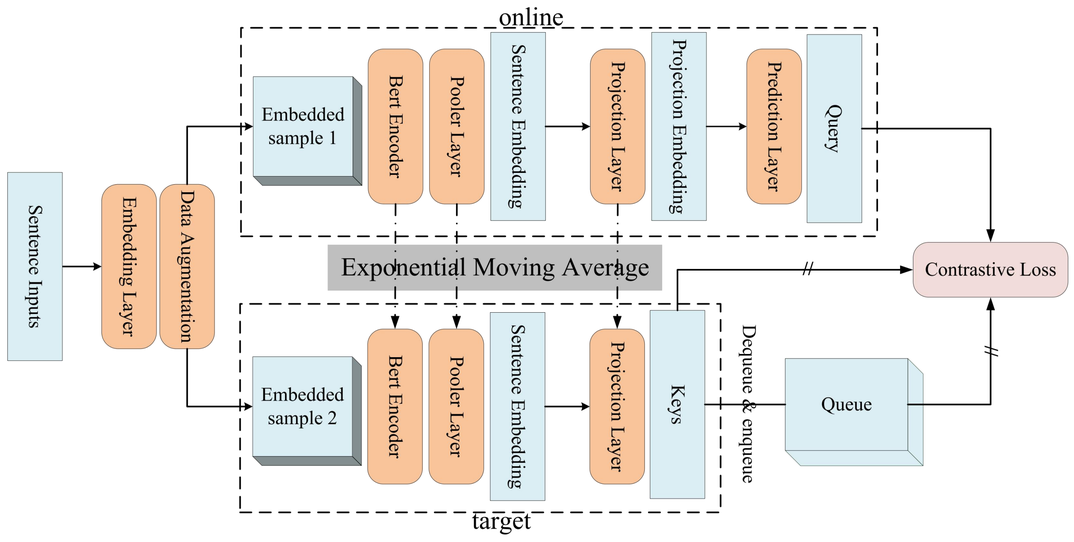


Fig.1 The model structure of MoCoSE. The embedding layer consists of a BERT embedding layer with additional data augmentation. The pooler, projection, and predictor layers all keep the same dimensions with the encoder layer. The MoCoSE minimizes contrastive loss between query, queue and keys (i.e. InfoNCE loss).

Overview of Our Approach

We adapted a MoCo-style model to study the effect of the structure in text contrastive learning. The structure of the model is shown in Fig. 1.

The sentences are passed through the embedding layer and data augmentation to generate two slightly different embeddings, and then the embeddings of query and key are obtained through the online and target branches, respectively. The structure of encoder, pooler and projection of online and target branch is identical. Similar to MoCo V2, a prediction layer composed of MLP is added to the upper branch to prevent model from collapsing. We use the same InfoNCE loss as MoCo.

Algorithm 1: Momentum Contrastive Sentence Embedding

Input:
$\mathcal{D}$: Training data set ;
$\mathcal{Q}$: Negative Sample Queue;
E_a : Embedding with random data augmentation;
θ_o, θ_t : weights of online branch and target branch;
Optimizer : Adam optimizer
K, K_s: Queue size, Queue size at initialisation;
η : ema decay ema and ema scheduling strategy;
τ Temperature parameters
Output: MoCoSE model θ_o

Initializing the queue $\mathcal{Q}$ with size K_s;
foreach $\mathcal{B} \in \mathcal{D}$ **do**
  $v_o, v_t \leftarrow E_a(\mathcal{B}), E_a(\mathcal{B})$ // Using data Augmentation to generate different views
  $z_o \leftarrow \theta_o(v_o)$ // (N, d), N is batch size, d is dimension of sentence embedding
  $z_t \leftarrow \theta_t(v_t)$
  $l_{z_o, z_t, \mathcal{Q}} \leftarrow -\log \frac{\exp(z_o \cdot z_t/\tau)}{\exp(z_o \cdot z_t/\tau) + \sum_{x \in \mathcal{Q}} \exp(z_o \cdot x/\tau)}$ // compute contrastive loss using $InFoNCE$
  optimizer$(l_{z_o, z_t, \mathcal{Q}}, \theta_o)$ // Update only the parameters of the online branch according to the loss gradient;
  $\theta_t \leftarrow \eta * \theta_t + (1-\eta) * \theta_o$ // Update the parameters of the target branch using EMA
  enqueue$(\mathcal{Q}, v_t)$ // Update the negative sample queue $\mathcal{Q}$
  dequeue$(\mathcal{Q})$
return θ_o

The PyTorch style pseudo-code for training MoCoSE with the negative sample queue is shown in Algorithm 1.

Experiment and Analysis

1. Main Results

We train with a corpus of 1 million sentences randomly selected from the English Wikipedia and experiment in seven standard semantic text similarity (STS) tasks using [CLS] token embeddings from the online encoder output as sentence embeddings.

Learning rate:	
MoCoSE-BERT-base	3e-5
MoCoSE-BERT-large	1e-5
Batch size:	
MoCoSE-BERT-base	64
MoCoSE-BERT-large	32
Validate per step	100
EMA decay weight	0.75~0.95
Weight decay	1e-6
Negative queue size	512

In addition to the semantic similarity task, we evaluate seven migration learning tasks. The proposed base model surpasses SimCSE in most of unsupervised tasks, and the large model obtains better results in most migration tasks.

Model	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
Unsupervised Models (Base)								
GloVe (avg.)	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
BERT (first-last avg.)	39.70	59.38	49.67	66.03	66.19	53.87	62.06	56.70
BERT-flow	58.40	67.10	60.85	75.16	71.22	68.66	64.47	66.55
BERT-whitening	57.83	66.90	60.90	75.08	71.31	68.24	63.73	66.28
IS-BERT	56.77	69.24	61.21	75.23	70.16	69.21	64.25	66.58
CT-BERT	61.63	76.80	68.47	77.50	76.48	74.31	69.19	72.05
RoBERTa (first-last avg.)	40.88	58.74	49.07	65.63	61.48	58.55	61.63	56.57
RoBERTa-whitening	46.99	63.24	57.23	71.36	68.99	61.36	62.91	61.73
DeCLUTR-RoBERT	52.41	75.19	65.52	77.12	78.63	72.41	68.62	69.99
SimCSE	68.40	**82.41**	74.38	80.91	78.56	76.85	72.23	76.25
MoCoSE	**71.48**	81.40	**74.47**	**83.45**	**78.99**	**78.68**	**72.44**	**77.27**
Unsupervised Models (Large)								
SimCSE-RoBERTa	72.86	83.99	75.62	**84.77**	**81.80**	**81.98**	71.26	78.90
SimCSE-BERT	70.88	84.16	76.43	84.50	79.76	79.26	**73.88**	78.41
MoCoSE-BERT	**74.50**	**84.54**	**77.32**	84.11	79.67	80.53	73.26	**79.13**

Table 1: Spearman correlation of MoCoSE on seven semantic text similarity tasks. We compared with the state-of-the-art method SimCSE. MoCoSE achieves the best results with both BERT-base and BERT-large pre-trained models.

Model	MR	CR	SUBJ	MPQA	SST	TREC	MRPC	Avg.
Unsupervised Model (Base)								
GloVe (avg.)	77.25	78.30	91.17	87.85	80.18	83.00	72.87	81.52
Skip-thought	76.50	80.10	93.60	87.10	82.00	92.20	73.00	83.50
Avg. BERT embeddings	78.66	86.25	94.37	88.66	84.40	**92.80**	69.54	84.94
BERT-[CLS]embedding	78.68	84.85	94.21	88.23	84.13	91.40	71.13	84.66
SimCSE-RoBERTa	81.04	**87.74**	93.28	86.94	**86.60**	84.60	73.68	84.84
SimCSE-BERT	**81.18**	86.46	94.45	88.88	85.50	89.80	74.43	**85.81**
MoCoSE-BERT	81.07	86.43	**94.76**	**89.70**	86.35	84.06	**75.86**	85.46
Unsupervised Model (Large)								
SimCSE-RoBERTa	82.74	87.87	93.66	88.22	**88.58**	**92.00**	69.68	86.11
MoCoSE-BERT	**83.71**	**89.07**	**95.58**	**90.26**	87.96	84.92	**76.81**	**86.90**

Table 2: Performance of MoCoSE on the seven transfer tasks. We compare the performance of MoCoSE and other models on the seven transfer tasks evaluated by SentEval, and MoCoSE remains at a comparable level with the SimCSE.

2. Abalation Study

Symmetric Two-branch Structure

Figure 5: Experiment on a symmetric two-branch structure with EMA decay weight set to 0.

Proj.	Pred.	Corr.	Proj.	Pred.	Corr.
	1	60.46		1	66.96
0	2	62.67	2	2	66.29
	3	63.62		3	61.57
	1	76.74		1	31.51
1	**2**	**76.89**	3	2	43.97
	3	76.24		3	39.13

Table 4: The impact of different combinations of projection and predictor on the model.

EMA Hyperparameters

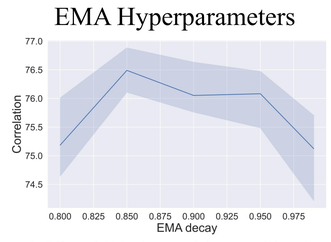


Figure 8: Effect of EMA decay weight on model performance.

EMA	0.5	0.8	**0.85**	0.9	0.95	0.99
Avg.	75.76	75.19	**76.49**	76.05	76.08	75.12

Table 3: Effect of EMA decay weight on model performance. The best results are obtained with the EMA decay weight at 0.85.

Compared to the choice of EMA decay weight in CV (generally as large as 0.99), the value of 0.85 in our model is smaller, which means that the model is updated faster.

Different Data Augmentations

Augmentation Methods	Avg.
Dropout only	76.76
+ **FGSM** (ε=5e-9)	**77.04**
+ Position_shuffle (True)	73.80
+ Token dropout (prob=0.1)	41.32
+ Feature dropout (prob=0.01)	76.33
+ Feature dropout (prob=0.1)	71.62
+ Typos	22.32
+ Synonym replace (roberta-base)	28.70
+ Paraphrasing (xlnet-base-cased)	60.45
+ Backtranslation (en->de->en)	69.35

Epsilon	1e-9	**5e-9**	1e-8	5e-8	No
Avg.	75.61	**76.64**	75.39	76.62	76.26

Table 9: Different parameters of FGSM in data augmentation affect the model results.

FGSM first calculates the derivative of model with respect to the input, and use a sign function to obtain its specific gradient direction. Then, after multiplying it by a step size, the result_x0002_ing ' perturbation' is added to the original input to obtain the sample under the FGSM attack.

We use the nlpaug toolkit* in our data augmentation experiments. All the parameter listing is default value given by official.

* https://github.com/makcedward/nlpaug

Predictor Mapping Dimension

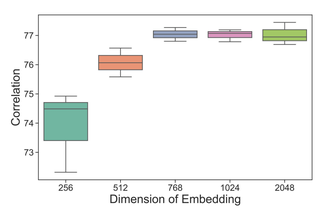


Figure 10: Impact of dimensions of the sentence embedding on the model with fixed queue size of 512.

When the dimension of embedding is low, this causes considerable damage to the performance of the model, while the dimension rises to certain range, the performance of the model stays steady.

Batch Size

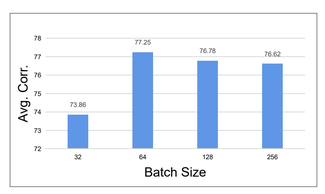


Table 6(b): Impact of batch size on the model with fixed queue size of 512.

The model performance does not improve with increasing batch size, which contradicts the general experience in image contrastive learning.

Distribution of Singular Values

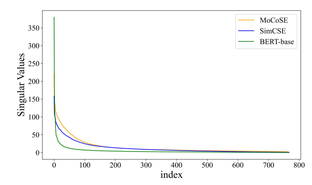


Figure 7: Singular value distributions of sentence embedding matrix from sentences in STS-B.

The model is able to alleviate the rapid decline of singular values compared to other methods, making the curve smoother, i.e., our model is able to make the sentence embedding more isotropic.

The rapid decrease of the singular value means that the distribution is wider in the first few directions (singular value represents the variance) and smaller in the later directions, the decrease of the singular value slows down means the variance of the distribution becomes more consistent and uniform in all directions.

3. The study of Negative Sample Queue

We test the size of negative sample queue to the model performance. With queue size longer than 1024, the results get unstable and worse. We suppose this may be due to the random interference introduced to the training by filling the initial negative sample queue.

Initial Size	Queue Size 128	256	512	1024	4096
w.o. init.	76.40	76.19	75.38	**76.63**	50.17
init. 1/4 queue	75.92	76.34	**77.30**	76.20	**50.42**
init. 1/2 queue	76.16	**76.39**	76.94	76.57	38.74
init. all (normal)	**76.87**	75.81	76.29	76.45	45.80

Table 7: Correlation performance of initializing different proportion of negative queue with different negative queue size.

Figure 12: The impact of different initial negative sample queue sizes for different initial sizes on model performance. (left):Zoomed view. (right):Overview with different negative queue size. Results of different initial size under same queue size.

Add random inital process to First-in-First-out queue

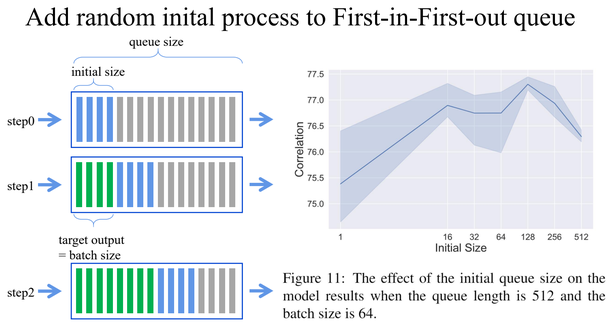
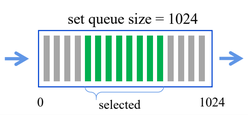


Figure 11: The effect of the initial queue size on the model results when the queue length is 512 and the batch size is 64.

We initialize a smaller negative queue, then fill the queue to its set length in the first few updates, and then update normally.



Corr.	0∼ 512	256∼ 768	512∼ 1024	Without 256∼768	All
Avg.	76.10	**77.02**	75.71	76.18	76.86

Table 8: The impact of negative samples at different locations in the queue on the model performance.

We find that the increase in queue length affects the model performance not only because of the increased number of negative samples, but more because it provides historical information within a certain range.

We define a Maximum Traceable Distance Metric to help explore the phenomenon.

$$d_{trace} = \frac{1}{1-\eta} + \frac{queue\_size}{batch\_size}$$

The η refers to the decay weight of EMA. The d_{trace} calculates the update steps between the current online branch and the oldest negative samples in the queue.

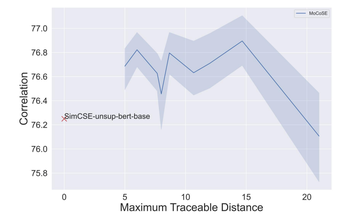


Figure 2: The relationship between traceable distance and model correlation.

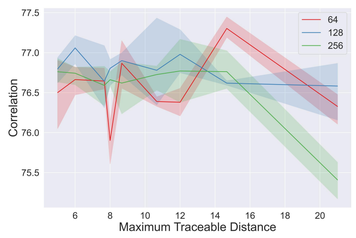


Figure 3: The batch size does not invalidate the traceable distance. The traceable distance needs to be maintained within a reasonable range even for different batch sizes. This explains why increasing the batch size only does not improve the performance, because increasing the batch size only can cause the distance changes into unsuitable regions.

Conclusion

In this study, we propose MoCoSE, it applies the MoCo-style contrastive learning model to the empirical study of sentence embedding. Our further experiments find that the model performance is related to the EMA update parameter, batch size, and the size of the negative sample queue during training.

We define the Maximum Traceable Distance (MTD) to measure the relationship between them, and found there's an 'age'-like concept in the negative sample queue for text contrastive learning.

Affiliation: State-Province Joint Engineering and Research Center of Advanced Networking and Intelligent Information Services, Northwest University, Xi'an, Shaanxi, China.