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

Anonymous ACL submission

Abstract

Contrastive learning is emerging as a powerful self-supervised technique for extracting knowledge from unlabeled image and text 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. 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 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. Our experiments find that the best results are obtained when the maximum traceable distance is at a certain range, demonstrating that there is an optimal range of historical information for a negative sample queue. We evaluate the proposed unsupervised MoCoSE on the semantic text similarity (STS) task and obtain an average Spearman's correlation of 77.27%. Source code is available here.

1 Introduction

In recent years, unsupervised learning has been brought to the fore in deep learning due to its ability to leverage large-scale unlabeled data. In computer vision, various unsupervised contrastive learning models have emerged, narrowing down the gap between supervised and unsupervised learning. Most contrastive models require negative samples to avoid model collapsing, i.e., to prevent the model from converging to a constant solve so that all samples are mapped to one point in the feature space.

There are several ways to obtain negative samples in contrastive learning. In computer vision, SimCLR series from Chen(Chen et al., 2020) and MoCo series from He(He et al., 2020) is known for using negative samples and get the leading performance in the contrastive learning. SimCLR uses different data augmentation (e.g., rotation, masking, etc.) on the same image to construct positive samples, and negative samples are from the rest of images in the same batch. MoCo goes a step further by randomly select the data in entire unlabeled training set to stack up a first-in-first-out negative sample queue.

Recently in natural language processing, contrastive learning has been widely used in the task of sentence embedding. The current state-of-the-art unsupervised method is SimCSE(Gao et al., 2021). Similar to image contrastive learning, the core idea of SimCSE is to make similar sentences in the embedding space closer while keeping dissimilar away from each other. SimCSE uses dropout mask as augmentation to construct positive text sample pairs, and negative samples are picked from the rest of sentences in the same batch. The dropout mask adopted from the standard Transformer makes good use of the minimal form of data augmentation brought by the dropout mechanism. Dropout results in a minimal difference without changing the semantics, reducing the negative noise introduced by data augmentation. However, the negative samples in SimCSE are selected from the same training batch with a limited batch size. Further experiments show that SimCSE does not obtain improvement as the batch size increases, which arouses our interest in using the negative sample queue.

We design a text contrastive learning model consisting of a two-branch structure and a negative sample queue, namely MoCoSE (**Mo**mentum

Contrastive **S**entence **E**mbedding with negative sample queue). We also introduce the idea of asymmetric structure from BYOL(Grill et al., 2020) by adding a prediction layer to the upper branch (i.e., the online branch). The lower branch (i.e., the target branch) is updated with exponential moving average (EMA) method during training. We set a negative sample queue and update it using the output of target branch. Unlike the negative queue in MoCo, we set an initialization process with a much smaller negative queue, and then filling the entire queue through training process, and then update normally. For text data augments, we test methods mentioned in ConSERT(Yan et al., 2021), including token shuffle, cut off, dropout and FGSM (Fast Gradient Sign Method).

Using the proposed MoCoSE model, we design a series of experiments to explore the contrastive learning for sentence embedding. We test the influence of prediction layer in the online branch with different mapping dimensions. We also test the effect of different text augmentation algorithms in MoCoSE. Result shows that FGSM can significantly bring the improvement, while token drop hurts the results substantially. In order to test how much text contrastive learning benefit from historical information of the model, we proposed a maximum traceable distance metric. The metric calculates how many update steps before the negative samples in the queue are pushed in, and thus measures the historical information contained in the negative sample queue. We find that the best results can be achieved when the maximum traceable distance is within a certain range, which means there is an optimal interval for the length of negative sample queue in text contrastive learning model.

Our main contributions are as follows:

1.We build a new text contrastive learning model. The model combines several advantages of the image contrastive learning framework, using asymmetric branching, EMA, and negative queues. These structures enable the model to achieve better results.

2.We evaluate the role of negative queues length and the historical information that queue contains in text contrastive learning. We define a metric named maximum traceable distance, and use it to assist in analyzing the effect of negative queue size, and also can be used to compute optimal negative queue length for a given batch size.

3.We study the influence of different text augmentation in text contrastive learning. Including token shuffle, cut off, token dropout, feature dropout, and FGSM. We carry out extensive experiments on the choice of specific optimal parameters for each augmentation method and verify that for text comparison learning, using FGSM and dropout as data augmentation can bring the most benefit.

2 Related Work

Contrastive Learning in CV

Contrast learning is a trending and effective unsupervised learning framework that was first applied to the computer vision(Hadsell et al., 2006). The core idea is to make the features of images within the same category closer and the features in different categories farther apart. Most of the current work are using two-branch structure(Chen et al., 2021). While influential works like SimCLR and MoCo using positive and negative sample pairs, BYOL(Grill et al., 2020)and SimSiam(Chen and He, 2021) can achieve the same great results with only positive samples. BYOL finds that by adding a prediction layer to the online branch to form an asymmetric structure and using momentum moving average to update the target branch, can train the model using only positive samples and avoid model collapsing. SimSiam explores the possibility of asymmetric structures likewise. Therefore, our work introduces this asymmetric idea to the text contrastive learning to prevent model collapse. In addition to the asymmetric structure and the EMA mechanism to avoid model collapse, some works consider merging the constraint into the loss function, like Barlow Twins(Zbontar et al., 2021), W-MSE(Ermolov et al., 2021), and ProtoNCE(Li et al., 2021).

Contrastive Learning in NLP

Since BERT(Devlin et al., 2018) redefined state-of-the-art in NLP, leveraging the BERT model to obtain better sentence representation has become a common task in NLP. A straightforward way to get sentence embedding is by the $[CLS]$ token due to the Next Sentence Prediction task of BERT. But the $[CLS]$ embedding is non-smooth anisotropic in semantic space, which is not conducive to STS tasks, this is known as the representation degradation problem(Gao et al., 2019). BERT-Flow(Li et al., 2020) and BERT-whitening(Su et al., 2021) solve the degradation problem by post-processing the output of BERT. SimCSE proposes supervised and unsupervised contrastive learning method to alleviate this problem.

Data augmentation is crucial for contrastive learning. In CLEAR(Wu et al., 2020), word and phrase deletion, phrase order switching, synonym substitution is served as augmentation. CERT(Fang and Xie, 2020) mainly using back-and-forth translation, and CLINE(Wang et al., 2021) proposed synonym substitution as positive samples and antonym substitution as negative samples, and then minimize the triplet loss between positive, negative cases as well as the original text. ConSERT(Yan et al., 2021) uses adversarial attack, token shuffling, cutoff, and dropout as data augmentation. CLAE(Ho and Nvasconcelos, 2020) also introduces Fast Gradient Sign Method, an adversarial attack method, as text data augmentation. Several of these augmentations are also introduced in our work. The purpose of data augmentation is to create enough distinguishable positive and negative samples to allow contrastive loss to learn the nature of same data after different changes. Works like (Mitrovic et al., 2020) points out that longer negative sample queues do not always give the best performance. This also interests us how the negative queue length affects the text contrastive learning.

3 Method

Figure 1 depicts the architecture of proposed MoCoSE. In the embedding layer, two versions of the sentence embedding are generated through data augmentation ($dropout = 0.1 + fgsm = 5e-9$). The resulting two slightly different embeddings then go through the online and target branch to obtain the query and key vectors respectively. The structure of encoder, pooler and projection of online and target branch is identical. We add prediction layer to the online branch to make asymmetry between online and target branch. The pooler, projection and prediction layers are all composed of several fully connected layers.

Finally, model calculates contrastive loss between query with keys and queue to update of the online branch, where keys served as positive samples with respect to the query vector, while the queue served as negative samples to the query. The target branch truncates the gradient and is updated with the EMA mechanism. The queue is a first-in-first-out collection of negative samples with size K which means it sequentially stores the keys vectors generated from the last few training steps.

The PyTorch style pseudo-code for training MoCoSE with the negative sample queue is shown in Algorithm 1 in Appendix A.2.

Data Augmentation Compared to SimCSE, we consider some additional data augmentation mechanisms mentioned in ConSERT, but experiments show that only adversarial attacks and dropout have improved the results. We use FGSM(Goodfellow et al., 2015) (Fast Gradient Sign Method) as adversarial attack. In a white-box environment, 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 resulting 'perturbation' is added to the original input to obtain the sample under the FGSM attack. The FGSM is expressed as follows:

$$x' = x + \varepsilon \cdot sign\left(\nabla_x \mathcal{L}\left(x, \theta\right)\right) \tag{1}$$

Where x is the input to the embedding layer, θ is the online branch of the model, and $\mathcal{L}(\cdot)$ is the contrastive loss computed by the query, keys and negative sample queue. ∇_x is the gradient computed through the network for input x, $sign()$ is the sign function, and ε is the perturbation parameter.

EMA and Asymmetric Branches Our model uses EMA mechanism to update the target branch. Formally, denoting the parameters of online and target branch as θ_o and θ_t, EMA decay weight as η, we update θ_t by:

$$\theta_t \leftarrow \eta\theta_t + (1-\eta)\theta_o \tag{2}$$

Experiments demonstrate that not using EMA leads to model collapsing, which means the model did not converge during training and did not obtain good results. We also add prediction layer to the online branch to make two branches asymmetric to further prevent model collapse. For more experiment details about symmetric model structure without EMA mechanism, please refer to Appendix A.1.

Negative Sample Queue Theoretically, if the negative samples are removed, the model will simply map all representations to the same point, thus satisfying the goal of narrowing the distance between positive pairs. This means the model will soon converge to a trivial solution, causing a model collapse problem. Therefore, with the use of double branching, we add a negative sample queue to

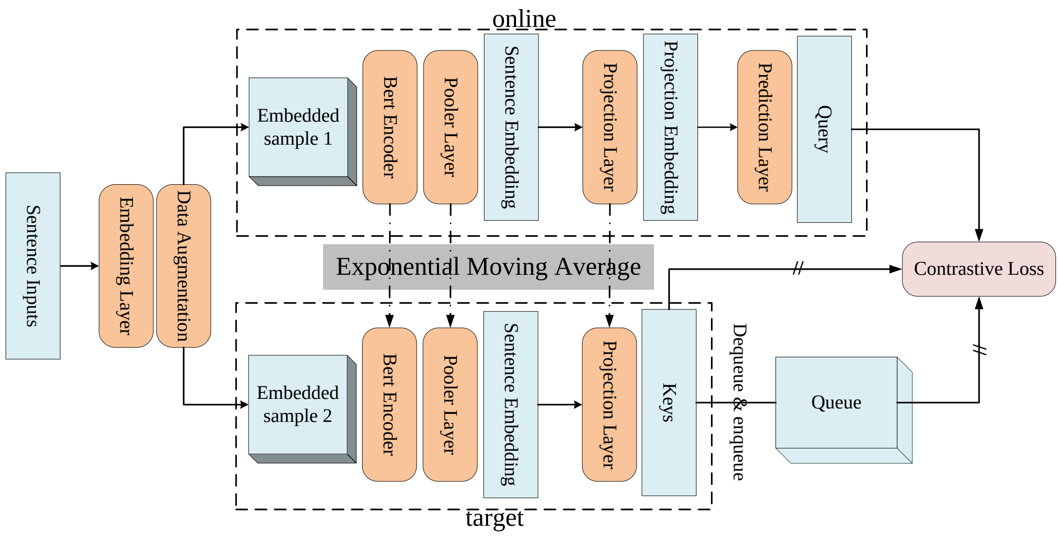

Figure 1: The model structure of MoCoSE. The embedding layer consists of a Bert embedding layer with additional data augmentation. The pooler, projection, and predictor layers all keep the same dimensions with the encoder layer. The MoCoSE minimizes contrastive loss between query, queue and keys (i.e.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

increase the negative sample number and increase the performance of the model.

Contrastive Loss Similar to MoCo, we also use InfoNCE(Oord et al., 2018) as contrastive loss, as shown in eq.(3).

$$\mathcal{L} = -\log \frac{exp\,(q \cdot k/\tau)}{exp\,(q \cdot k/\tau) + \sum_l exp\,(q \cdot l/\tau)} \quad (3)$$

Where, q refers to the query vectors obtained by the online branch; k refers to the key vectors obtained by the target branch; and l is the negative samples in the queue; τ is the temperature parameter.

4 Experiments

4.1 Settings

We train with a randomly selected corpus of 1 million sentences from the English Wikipedia, and we conduct experiments on seven standard semantic text similarity (STS) tasks, including STS 2012—2016(Agirre et al., 2012, 2013, 2014, 2015, 2016), STSBenchmark(Cer et al., 2017) and SICK-Relatedness(Wijnholds and Moortgat, 2021). The SentEval toolbox is used to evaluate our model, and we use the Spearman's correlation to measure the performance. We start our training by loading pre-trained Bert checkpoints[1] and use the $[CLS]$ token embedding of the model output as the sentence embedding. In addition to the semantic similarity task, we also evaluate on seven transfer learning tasks to test the generalization performance of the model.

Training Details We train our MoCoSE model using NVIDIA RTX3090 GPUs. We use Python 3.8 and PyTorch version v1.8. We use Transformers 4.4.2(Wolf et al., 2020) and Datasets 1.8.0(Lhoest et al., 2021) from Huggingface.

The learning rate of MoCoSE-BERT-base is set to 3e-5, and for MoCoSE-BERT-large is 1e-5. With a weight decay of 1e-6, the batch size of the base model is 64, and the batch size of the large model is 32. We validate the model every 100 step and train for one epoch. The EMA decay weight η is incremented from 0.85 to 1.0 by the cosine function.

4.2 Main Results

We compare the proposed MoCoSE with several commonly used methods and the current state-of-the-art contrastive learning method on the text semantic similarity (STS) task, including average GloVe embeddings(Pennington et al., 2014), average BERT or RoBERTa embeddings, BERT-flow, BERT-whitening, ISBERT(Zhang et al., 2020), DeCLUTR(Giorgi et al., 2021), CT-BERT(Carlsson et al., 2021) and SimCSE.

As shown in Table 1, the average Spearman's correlation of our best model is 77.27%, outperforming unsupervised SimCSE with BERT-base. Our model outperforms SimCSE significantly on

[1] https://huggingface.co/models

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	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.

STS2012, STS2015, and STS-B, and SimCSE perform better on the STS2013 task. Our MoCoSE-BERT-large model outperforms SimCSE-BERT-Large by about 0.7 on average, mainly on STS12, STS13, and STS14 tasks, and maintains a similar level on other tasks.

Furthermore, we also evaluate the performance of MoCoSE on the seven transfer tasks provided by SentEval. As shown in Table 2, MoCoSE-BERT-base outperforms most of the previous unsupervised method, and is on par with SimCSE-BERT-base.

5 Empirical Study

We build the MoCoSE with common and effective structures from image contrastive learning, such as the negative queue, initialization of the queue, data augmentation of text, etc. Therefore, we need to measure how much influence each of them brings. Thus, we set up the following ablation experiments.

5.1 EMA Decay Weight

We use EMA to update the model parameters for the target branch and find that EMA decay weight affects the performance of the model. The EMA decay weight affects the update process of the model, which further affects the vectors involved in the contrastive learning process. Therefore, we set different values of EMA decay weight and train the model with other hyperparameters held constant. As shown in Table 3 and Appendix A.4, the best result is obtained when the decay weight of EMA is set to 0.85. Compared to the choice of EMA decay

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

weight in CV (generally larger than 0.99), the value of 0.85 in our model is smaller, which means that the model is updated faster. We speculate that this is because the NLP model is more sensitive in the fine-tuning phase and the model weights change more after each step of the gradient, so a faster update speed is needed.

5.2 Projection and Prediction

Several papers have shown (e.g. Section F.1 in BYOL(Grill et al., 2020)) that the structure of projection and prediction layers in a contrastive learning framework affects the performance of the model. We combine the structure of projection and prediction with different configurations and train them with the same hyperparameters. As shown in Table 4, the best results are obtained when the projection is 1 layer and the prediction has 2 layers. The experiments also show that the removal of projection layers degrades the performance of the model.

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.

5.3 Data Augmentation

We investigate the effect of some widely-used data augmentation methods (token shuffle, cut off, dropout, and adversarial attack) on the model performance. As shown in Table 5, the experiments show that cut off and token shuffle do not improve, even slightly hurt the model's performance. Only the adversarial attack (we use FGSM) has slight improvement on the model. Therefore, in our experiments, we added FGSM as a data augmentation of our model in addition to dropout. Please refer to Appendix A.6 for more FGSM parameters results.

Augmentation Methods	Avg.
Dropout only	76.76
+ FGSM	**77.04**
+ Position_shuffle (True)	73.80
+ Token drop (prob=0.1)	41.32
+ Feature drop (prob=0.01)	76.33
+ Feature drop (prob=0.1)	71.62

Table 5: The effect of different data augmentation methods.

We speculate that the reason token cut off is detrimental to the model results is that the cut off perturbs too much the vector formed by the sentences passing through the embedding layer. Removing one word from the text may have a significant impact on the semantics. We tried two parameters 0.1 and 0.01 for the feature cut off, and with these two parameters, the results of using the feature cut off is at most the same as without using feature the cut off, so we discard the feature cut off method. More results can be found in Appendix A.5.

The token shuffle is slightly, but not significantly, detrimental to the results of the model. This may be due to that BERT is not sensitive to the position of token. We did not add token shuffle to the final data augmentation.

Among the data augmentation methods, only FGSM together with dropout improves the results, which may due to the adversarial attack slightly enhances the difference between the two samples and therefore enables the model to learn a better representation in more difficult contrastive samples.

5.4 Predictor Mapping Dimension

The predictor maps the representation to a feature space of a certain dimension. We investigate the effect of the predictor mapping dimension on the model performance. Table 6.a shows that the predictor mapping dimension can seriously impair the performance of the model when it is small, and when the dimension rises to a suitable range or larger, it no longer has a significant impact on the model. This may be related to the intrinsic dimension of the representation, which leads to the loss of semantic information in the representation when the predictor dimension is smaller than the intrinsic dimension of the feature, compromising the model performance. We keep the dimension of the predictor consistent with the encoder in our experiments.

More results can be found in Appendix A.7.

Dim	Avg.
256	73.91
512	76.07
768	**77.04**
1024	77.02
2048	77.03

(a)

Size	Avg.
32	73.86
64	**77.25**
128	76.78
256	76.62

(b)

Table 6: (a) Impact of prediction dimension on model performance. (b) Impact of batch size on the model with fixed queue size.

5.5 Batch Size

With a fixed queue size, we investigated the effect of batch size on model performance, the results in Table 6.b, and the model achieves the best performance when the batch size is 64. Surprisingly the model performance does not improve with increasing batch size, which contradicts the general experience in image contrastive learning. This is one of our motivations for further exploring the effect of the number of negative samples on the model.

5.6 Size of Negative Sample Queue

The queue length determines the number of negative samples, which direct influence performance of the model. Thus, we study in detail on how the length of the negative sample queue affect the model. We first test the initialization of negative sample queue with different initial size, and not surprisingly to find the impact on the final performance. We suppose this may be due to the random interference introduced to the training by filling the initial negative sample queue. This interference causes a degradation of the model's performance when the initial negative sample queue becomes longer. To reduce the drawbacks carried out by this randomness, we changed the way the negative queue is initialized. We initialize a smaller negative queue, then fill the queue to its set length in the first few updates, and then update normally. According to experiments, the model achieves the highest results when the negative queue size set to 512 and the smaller initial queue size set to 128.

According to the experiments of MoCo, the increase of queue length improves the model performance. However, as shown in Table 7, increasing the queue length with a fixed batch size decreases our model performance, which is not consistent with the observation in MoCo. We speculate that this may be due to that NLP models updating faster, and thus larger queue lengths store too much outdated feature information, which is detrimental to the performance of the model. Combined with the observed effect of batch size, we further conjecture that the effect of the negative sample queue on model performance is controlled by the model history information contained in the negative sample in the queue. See Appendix A.8 and A.9 for more results of the effect of randomization size and queue length.

Queue Size	Initial Size	Avg.
128	1	76.40
	32	75.92
	64	76.16
	128	76.87
256	1	76.19
	64	76.34
	128	76.39
	256	75.81
512	1	75.38
	128	**77.30**
	256	76.94
	512	76.29

Queue Size	Initial Size	Avg.
1024	1	76.63
	128	54.15
	256	76.20
	512	76.57
	1024	76.45
4096	1	50.17
	128	49.13
	1024	50.42
	2048	38.74
	4096	45.80

Table 7: Impact of queue length on model performance with fixed batch size.

In our experiments, we found that simply increasing the batch size does not improve the model performance, while adding a negative queue can give us better results. We speculate that the negative queue contains not only a larger number of negative samples, but also contains information about the history of the model, which makes harder negative samples, thus improving the performance of the model. To test this hypothesis, we train a new model with the same structure as our model, but with different ways of updating the negative sample queue.

We propose two comparison models. The first model maintains a queue of sentence samples, which is also updated at each training step using a first-in-first-out approach. At each step, the current target network is used to generate the latest sentence embedding to fill the negative sample queue, and then the model is updated using the same loss function. The comparison model uses the current target model to obtain the negative sample queue, thus reducing the historical information in the queue. Another comparison model uses

Corr.	**normal**	latest	oldest
Avg.	**77.30**	76.65	76.04

Table 8: Impact of changing the update strategy of the queue on the model with fixed batch size and queue length.

samples from older queue as negative samples. It maintains a negative sample queue of length 1024, but use only the 512 negative samples queued first, thus using older negative samples for contrastive learning.

The results of these two comparison models are shown in the Table 8, and they both reduce the model performance. So 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.

5.7 Maximum Traceable Distance Metric

In order to explore more secrets of negative queue, we define the Maximum Traceable Distance Metric as eq.4.

$$d_{trace} = \frac{1}{1-\eta} + \frac{queue\_size}{batch\_size} \tag{4}$$

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. The first term of the formula represents the traceable distance between target and online branch due to the EMA update mechanism. The second term represents the traceable distance between the negative samples in the queue and the current target branch due to the queue's first-in-first-out mechanism. The longer traceable distance, the wider the temporal range of the historical information contained in the queue. We obtained different value of traceable distance by jointly adjust the decay weight, queue size, and batch size. As shown in Figure 2 and Figure 3, the best result of BERT base is obtained with d_{trace} is set around 14.67. The best result of Bert large shows the similar phenomenon, see Appendix A.10 for details. This further demonstrates that in text contrastive learning, the historical information used should be not too old and not too new, and the appropriate traceable distance between branches is also important. Some derivations about eq.4 can be found in Appendix A.11.

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

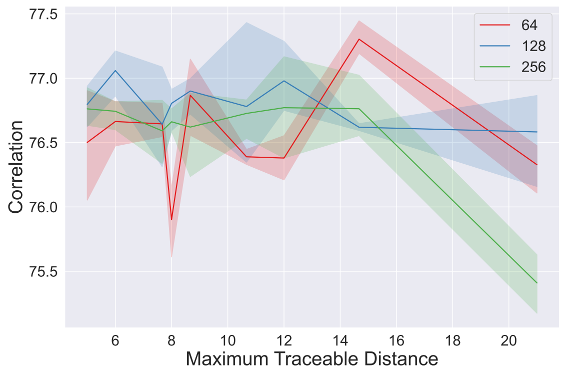

Figure 3: The batch size does not invalidate the traceable distance. The traceable distance needs to be maintained within a reasonable range even for different batch sizes. This explains why increasing the batch size only does not improve the performance, because increasing the batch size only can cause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

6 Conclusion

In this work, we propose MoCoSE, a new negative sample queue based text contrastive learning framework that surpasses the current SOTA model. We further delve into the application of the negative sample queue to text contrastive learning and propose maximum traceable distance metric to explain the relation between the queue size and model performance. We also investigate the application of multiple text augmentation methods in our proposed contrastive learning model.

In addition, we observe that the performance of negative queue in MoCoSE is quite different from the performance of different image constrative learning models (e.g., MoCo, MoCoV3), and therefore, further experiments are needed to investigate in depth why negative queue mechanisms between modalities exhibit such differences, which will be our future work.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91.

Eneko Agirre, Carmen Banea, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, volume 1, pages 385–393.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, volume 1, pages 32–43.

Fredrik Carlsson, Magnus Sahlgren, Evangelia Gogoulou, Amaru Cuba Gyllensten, and Erik Ylipää Hellqvist. 2021. Semantic re-tuning with contrastive tension. In *ICLR 2021: The Ninth International Conference on Learning Representations*.

Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14.

Pengguang Chen, Shu Liu, and Jiaya Jia. 2021. Jigsaw clustering for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11526–11535.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Xinlei Chen and Kaiming He. 2021. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina N. Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186.

Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. 2021. Whitening for self-supervised representation learning. In *ICML 2021: 38th International Conference on Machine Learning*, pages 3015–3024.

Hongchao Fang and Pengtao Xie. 2020. Cert: Contrastive self-supervised learning for language understanding. *arXiv preprint arXiv:2005.12766*.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Representation degeneration problem in training natural language generation models. *arXiv preprint arXiv:1907.12009*.

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

John Giorgi, Osvald Nitski, Bo Wang, and Gary Bader. 2021. DeCLUTR: Deep contrastive learning for unsupervised textual representations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 879–895, Online. Association for Computational Linguistics.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *ICLR 2015 : International Conference on Learning Representations 2015*.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. 2020. Bootstrap your own latent: A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 21271–21284.

R. Hadsell, S. Chopra, and Y. LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738.

Chih-Hui Ho and Nuno Nvasconcelos. 2020. Contrastive learning with adversarial examples. In *Advances in Neural Information Processing Systems*, volume 33, pages 17081–17093.

Quentin Lhoest, Albert Villanova del Moral, Patrick von Platen, Thomas Wolf, Yacine Jernite, Abhishek Thakur, Lewis Tunstall, Suraj Patil, Mariama Drame, Julien Chaumond, Julien Plu, Joe Davison, Simon Brandeis, Victor Sanh, Teven Le Scao, Kevin Canwen Xu, Nicolas Patry, Steven Liu, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Nathan Raw, Sylvain Lesage, Anton Lozhkov, Matthew Carrigan, Théo Matussière, Leandro von Werra, Lysandre Debut, Stas Bekman, and Clément Delangue. 2021. huggingface/datasets: 1.13.2.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9119–9130.

Junnan Li, Pan Zhou, Caiming Xiong, and Steven Hoi. 2021. Prototypical contrastive learning of unsupervised representations. In *ICLR 2021: The Ninth International Conference on Learning Representations*.

Jovana Mitrovic, Brian McWilliams, and Melanie Rey. 2020. Less can be more in contrastive learning. *"I Can't Believe It's Not Better!" NeurIPS 2020 workshop*.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Jianlin Su, Jiarun Cao, Weijie Liu, and Yangyiwen Ou. 2021. Whitening sentence representations for better semantics and faster retrieval. *arXiv preprint arXiv:2103.15316*.

Dong Wang, Ning Ding, Piji Li, and Haitao Zheng. 2021. Cline: Contrastive learning with semantic negative examples for natural language understanding. In *ACL 2021: 59th annual meeting of the Association for Computational Linguistics*, pages 2332–2342.

Gijs Wijnholds and Michael Moortgat. 2021. Sick-nl: A dataset for dutch natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1474–1479.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Madian Khabsa, Fei Sun, and Hao Ma. 2020. Clear: Contrastive learning for sentence representation. *arXiv preprint arXiv:2012.15466*.

Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. Consert: A contrastive framework for self-supervised sentence representation transfer. In *ACL 2021: 59th annual meeting of the Association for Computational Linguistics*, pages 5065–5075.

Jure Zbontar, Li Jing, Ishan Misra, yann lecun, and Stephane Deny. 2021. Barlow twins: Self-supervised learning via redundancy reduction. In *ICML 2021: 38th International Conference on Machine Learning*, pages 12310–12320.

Yan Zhang, Ruidan He, Zuozhu Liu, Kwan Hui Lim, and Lidong Bing. 2020. An unsupervised sentence embedding method by mutual information maximization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1601–1610.

A Appendix

A.1 Symmetric Two-branch Structure

We remove the online branch predictor and set the EMA decay weight to 0, i.e., make the structure and weights of the two branches identical. As shown in Figure 4, it is clear that the model is collapsing at this point. And we find that the model always works best at the very beginning, i.e., training instead hurts the performance of the model. In addition, as the training proceeds, the correlation coefficient of the model approaches 0, i.e., the prediction results have no correlation with the actual labeling. At this point, it is clear that a collapse of the model is observed. We observed such a result for several runs, so we adopted a strategy of double branching with different structures plus EMA momentum updates in our design. Subsequent experiments demonstrated that this allowed the model to avoid from collapsing.

Figure 4: Experiment on a symmetric two-branch structure with EMA decay weight set to 0.

Figure 5: Experiment after adding predictor on the online branch with EMA decay weight set to 0.

We add predictor to the online branch and set the EMA decay weight to 0. We find that the model also appears to collapse and has a dramatic oscillation in the late stage of training, as shown in Figure 5.

A.2 Pseudo-Code for Training MoCoSE

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

A.3 Distribution of Singular Values

Similar to SimCSE, we plot the distribution of singular values of MoCoSE sentence embeddings with SimCSE and Bert for comparison. As illustrated in Figure 6, our method 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.

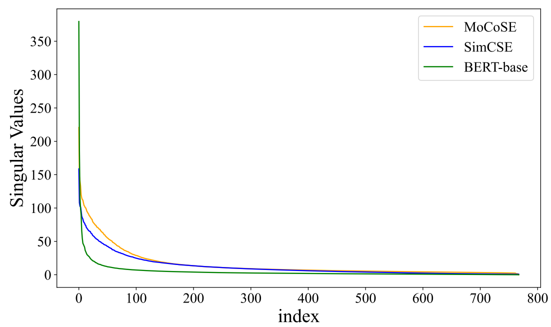

Figure 6: Singular value distributions of sentence embedding matrix from sentences in STS-B.

A.4 Experiment Details of EMA Hyperparameters

The details of the impact caused by the EMA parameter are shown in the Figure 7. We perform this experiment with all parameters held constant except for the EMA decay weight.

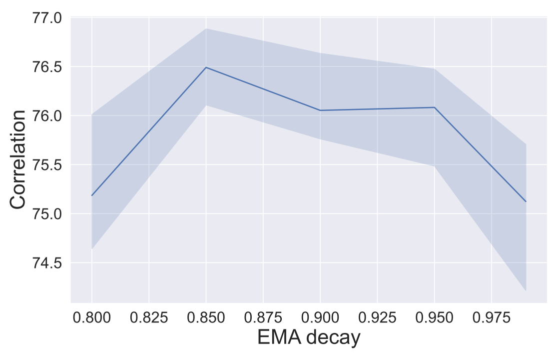

Figure 7: Effect of EMA decay weight on model performance.

Algorithm 1: Momentum **Co**ntrastive **S**entence **E**mbedding

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

A.5 Details of Different Data Augmentations

We use only dropout as a baseline for the results of data augmentations. Then, we combine dropout with other data augmentation methods and study their effects on model performance. The results are shown in Figure 8.

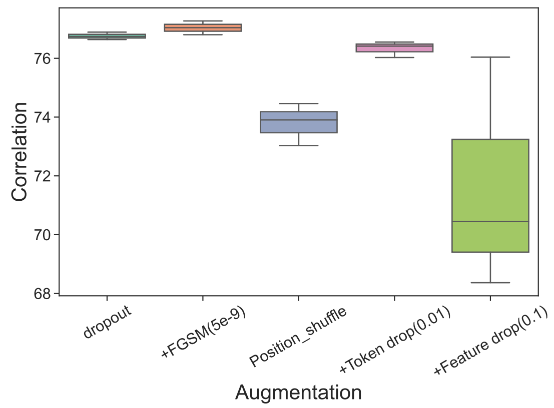

Figure 8: Impact of four additional data enhancements with dropout combinations on the model.

A.6 Experiment Details of FGSM

We test the effect of the intensity of FGSM on the model performance. We keep the other hyper-parameters fixed, vary the FGSM parameters (1e-9, 5e-9, 1e-8, 5e-8). As seen in Table 9, the average results of the model are optimal when the FGSM parameter is 5e-9.

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.

A.7 Dimension of Sentence Embedding

In both BERT-whitening (Su et al., 2021) and MoCo (He et al., 2020), it is mentioned that the dimension of embedding can have some impact on the performance of the model. Therefore, we also changed the dimension of sentence embedding in MoCoSE and trained the model several times to observe the impact of the embedding dimension. Because of the queue structure of MoCoSE, we need to keep the dimension of negative examples consistent while changing the dimension of sentence embedding. As shown in the Figure 9, when the dimension of Embedding is low, this causes considerable damage to the performance of the model; while when the dimension rises to certain range, the performance of the model stays steady.

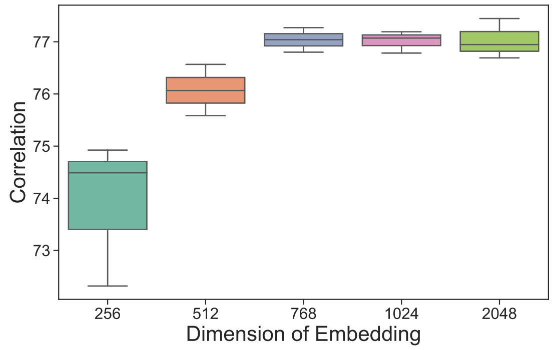

Figure 9: Impact of dimensions of the sentence embedding.

A.8 Details of Random Initial Queue Size

We test the influence of random initialization size of the negative queue on the model performance when queue length and batch size are fixed. As seen in Figure 10, random initialization does have some impact on the model performance.

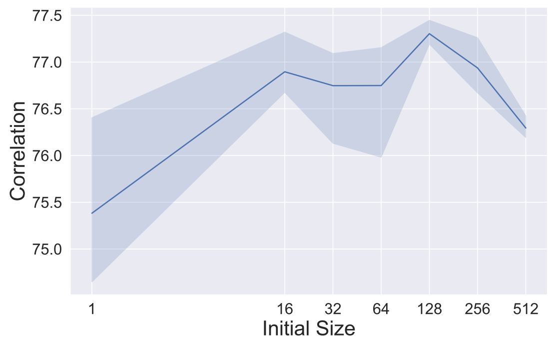

Figure 10: The effect of the initial queue size on the model results when the queue length is 512 and the batch size is 64.

A.9 Queue Size and Initial Size

We explored the effect of different combinations of initial queue sizes and queue length on the model performance. The detailed experiment results are shown in Figure 11. It can be found that model performance rely deeply on initialization queue size. Yet, too large queue size will make the model extremely unstable. This is quite different from the observation of negative sample queue in image contrastive learning.

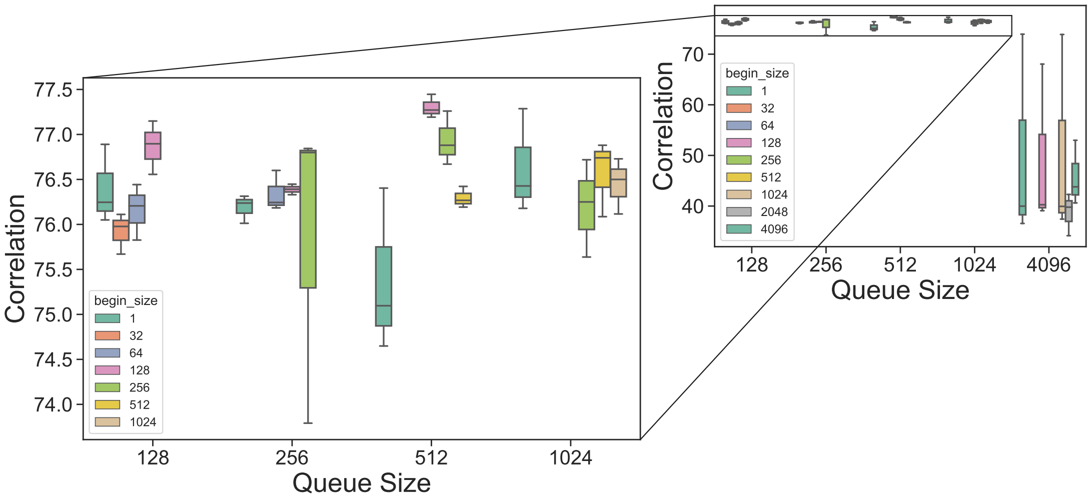

Figure 11: The impact of different initial negative sample queue sizes for different initial sizes on model performance. (left):Zoomed view. (right):Overview with different negative queue size. Results of different initial size under same queue size.

A.10 Maximum Traceable Distance in Bert-large

We also train mocose with different batch size and queue size on Bert-large. As shown in Figure 12, we observe the best model performance in MoCoSE-BERT-large within the appropriate Maximum Traceable Distance range (around 22). Once again, this suggests that even on BERT-large, the longer queue sizes do not improve the model performance indefinitely. Which also implies that the history information contained in the negative sample queue needs to be kept within a certain range on BERT-large as well.

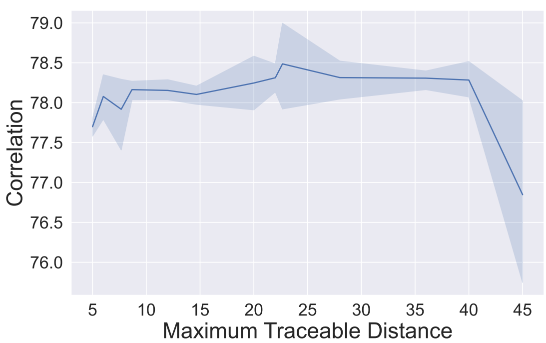

Figure 12: The relationship between mtd and correlation of MoCoSE-BERT-large. It can be seen that even at large model, peaks occur within a certain mtd range.

A.11 Proof of Maximum Traceable Distance

Here, we prove the first term of the formula for Maximum Traceable Distance. Due to the EMA update mechanism, the weight of target branch is a weighted sum of the online weight in update history. The first term of Maximum Traceable Distance calculate the weighted sum of the historical update steps given a certain EMA decay weight η. From the principle of EMA mechanism, we can get the following equation.

$$\mathcal{S}_n = \sum_{i=0}^{k} (1-\eta) \cdot \eta^i \cdot (i+1) \tag{5}$$

$\mathcal{S}_n$ represents the update steps between online and target branch due to the EMA mechanism. Since EMA represents the weighted sum, we need to ask for $\mathcal{S}_n$ to get the weighted sum.

We can calculate $\mathcal{S}_n$ as:

$$\mathcal{S}_n = (-1) * \eta^{k+1} * (k+1) - \frac{(1-\eta^{k+1})}{(\eta-1)} \tag{6}$$

As k tends to infinity, the limit for $\mathcal{S}_n$ can be calculated as following:

$$\lim_{k\to\infty} \mathcal{S}_n = \lim_{k\to\infty} \left[(-1) * \eta^{k+1} * (k+1) - \frac{(1-\eta^{k+1})}{(\eta-1)} \right] \tag{7}$$

It is obvious to see that the limit of the equation 7 consists of two parts, so we calculate the limit of these two parts first.

$$\lim_{k\to\infty} (-1) * \eta^{k+1} * (k+1) \overset{\eta \leq 1}{=} 0 \tag{8}$$

The limit of the first part can be calculated as 0.

Next, we calculate the limit of the second part.

$$\lim_{k\to\infty} \frac{\left(1-\eta^{k+1}\right)}{(\eta-1)} \overset{\eta\leq 1}{=} \frac{1}{1-\eta} \tag{9}$$

We calculate the limit of the second part as $\frac{1}{1-\eta}$. Since the limits of both parts exist, we can obtain the limit of $\mathcal{S}_n$ by the law of limit operations.

$$\begin{aligned}\lim_{k\to\infty} \mathcal{S}_n &= \lim_{k\to\infty} \left[(-1)*\eta^{k+1}*(k+1) - \frac{\left(1-\eta^{k+1}\right)}{(\eta-1)}\right] \\ &= \lim_{k\to\infty} (-1)*\eta^{k+1}*(k+1) - \lim_{k\to\infty} \frac{\left(1-\eta^{k+1}\right)}{(\eta-1)} \\ &= \frac{1}{1-\eta}\end{aligned} \tag{10}$$