ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230579

Enhancing Text Generation with Cooperative Training

Tong Wu[**;a;*], **Hao Wang**[b;**], **Zhongshen Zeng**[b], **Wei Wang**[a], **Hai-Tao Zheng**[a,c;***] **and Jiaxing Zhang**[***;b]

[a]Shezhen International Graduate School, Tsinghua Universiy
[b]International Digital Economy Academy
[c]Pengcheng Laboratory
ORCiD ID: Tong Wu[**;] https://orcid.org/0009-0003-3154-1213

Abstract. Recently, there has been a surge in the use of generated data to enhance the performance of downstream models, largely due to the advancements in pre-trained language models. However, most prevailing methods trained generative and discriminative models in isolation, which left them unable to adapt to changes in each other. These approaches lead to generative models that are prone to deviating from the true data distribution and providing limited benefits to discriminative models. While some works have proposed jointly training generative and discriminative language models, their methods remain challenging due to the non-differentiable nature of discrete data. To overcome these issues, we introduce a *self-consistent learning* framework in the text field that involves training a discriminator and generator cooperatively in a closed-loop manner until a scoring consensus is reached. By learning directly from selected samples, our framework are able to mitigate training instabilities such as mode collapse and non-convergence. Extensive experiments on four downstream benchmarks, including AFQMC, CHIP-STS, QQP, and MRPC, demonstrate the efficacy of the proposed framework.

1 Introduction

The advance of Pre-trained Language Models (PLMs) like GPT-3 [1] and LLaMA [2] has substantially improved the performance of deep neural networks across a variety of Natural Language Processing (NLP) tasks. Various language models, based on the Transformer [3] architecture, have been proposed, leading to state-of-the-art (SOTA) performance on the fundamental discrimination tasks. These models are first trained with self-supervised training objectives (e.g., predicting masked tokens according to surrounding tokens) on massive unlabeled text data, then fine-tuned on annotated data to adapt to downstream tasks of interest. However, annotated data is usually limited to a wide range of downstream tasks, which results in overfitting and a lack of generalization to unseen data.

One straightforward way to deal with this data scarcity problem is data augmentation , and incorporating generative models to perform data augmentation has been widely adopted recently . Despite its popularity, the generated text can easily deviate from the real data distribution without exploiting any of the signals passed back from the discrimination task. In previous studies, generative data augmentation and discrimination have been well studied as separate problems, but it is less clear how these two can be leveraged in one framework and how their performances can be improved simultaneously.

Generative Adversarial Networks (GANs) [4] are good attempts to couple generative and discriminative models in an adversarial manner, where a two-player minimax game between learners is carefully crafted. GANs have achieved tremendous success in domains such as image generation , and related studies have also shown their effectiveness in semi-supervised learning. However, in the text field, GANs are difficult to train, most training objectives work well for only one model, either the discriminator or the generator, so rarely both learners can be optimal at the same time. This essentially arises from the adversarial nature of GANs, that during the process, optimizing one learner can easily destroy the learning ability of the other, making GANs fail to converge.

Another limitation of simultaneously optimizing the generator and the discriminator comes from the discrete nature of text in NLP, as no gradient propagation can be done from discriminators to generators. One theoretically sound attempt is to use reinforcement learning (RL), but the sparsity and the high variance of the rewards in NLP make the training particularly unstable [5].

To address these shortcomings, we novelly introduce a self-consistent learning framework based on one generator and one discriminator: the generator and the discriminator are alternately trained by way of cooperation instead of competition, and the selected samples are used as the medium to pass the feedback signal from the discriminator. Specifically, in each round of training, the samples generated by the generator are synthetically labeled by the discriminator, and then only part of them would be selected based on dynamic thresholds and used for the training of the discriminator and the generator in the next round. Several benefits can be discovered from this cooperative training process. First, a closed-loop form of cooperation can be established so that we can get the optimal generator and discriminator at the same time. Second, this framework helps improve the generation quality while ensuring the domain specificity of generator, which in turn contributes to training. Third, a steady stream of diverse synthetic samples can be added to the training in each round and lead to continuous improvement of the performance of all learners. Finally, we can start the training with only domain-related corpus and obtain strong results, while these data can be easily sampled with little cost or supervision. Also, the performance on labeled datasets can be further boosted based on the strong baselines. As an example to demonstrate the effectiveness of our framework in the text field, we examine it on four downstream text generation benchmarks, includ-

[*] The work was done as an intern at IDEA.
[**] These authors contributed equally to this work.
[***] Corresponding Authors. Email: zheng.haitao@sz.tsinghua.edu.cn, zhangjiaxing@idea.edu.cn

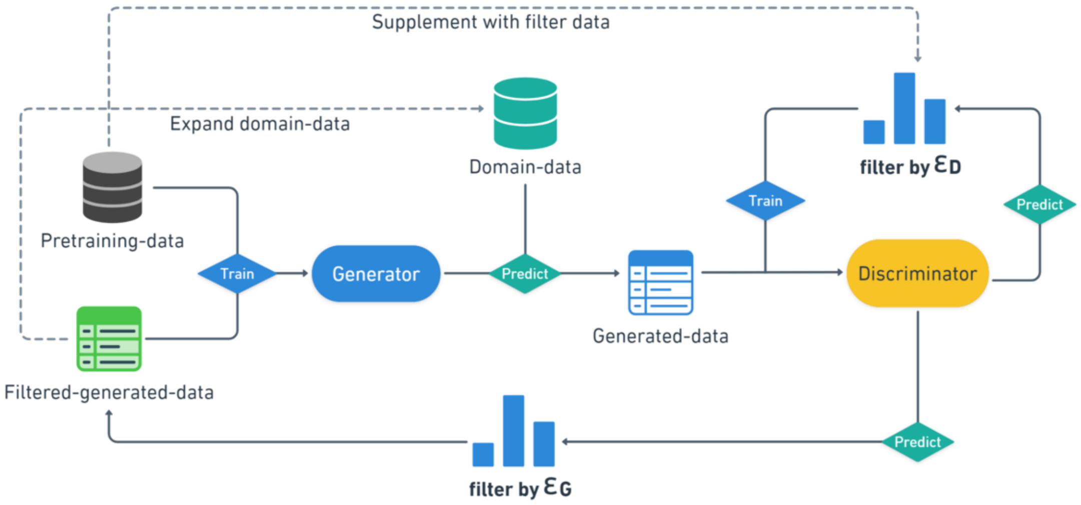

Figure 1: Overview of the flow chart for the SCL framework.

ing AFQMC, CHIP-STS, QQP, and MRPC. The experiments show that our method significantly improves over standalone state-of-the-art discriminative models on zero-shot and full-data settings.

Our contributions are summarized as follows,

• We propose a self-consistent learning framework in the text field that incorporates the generator and the discriminator, in which both achieve remarkable performance gains simultaneously.

• We propose a dynamic selection mechanism such that cooperation between the generator and the discriminator drives the convergence to reach their scoring consensus.

• Experimental results show that the generator in our framework can continuously adjust its generation samples based on the performance of downstream tasks, while the discriminator can outperform the strong baselines.

2 Related Works

To alleviate the lack of annotated data in supervised learning in NLP, semi-supervised learning (SSL) has been a popular line of research . The sources of the unlabeled data required by SSL are either collected from the domains or generated by generative language models. Then NLU models can learn from the unlabeled data by pseudo-labeling [6] and consistent regularization [7]. However, collecting unlabeled data comes at a cost(though smaller than labeling data), and the total amount is limited. Even with generative models, there is no guarantee of the quality of the generated samples, because the model cannot tune the generating results based on the performance of the downstream tasks. In contrast, our method usually includes a continuously updated generative model, which dynamically adjusts its generation according to the performance of downstream tasks.

GANs can be used as data enhancer to complement the lack of data for downstream tasks. Unlike conventional GANs in continuous domains, sequential GANs for discrete outputs are usually trained with reinforcement learning methods [8]. But they usually suffer from high variance, partly due to the non-stationarity nature of their reward distribution. Whereas work based on cooperative training has opened the way for more efficient methods. CoT [9] explicitly estimates and optimizes JS divergence through a joint maximization framework, ConcreteGAN [10] employs an autoencoder to learn implicit data manifold thus providing learning objective for adversarial training in a continuous space. However, their approach is still to back propagate through the gradient signal. More similar to our work is RML-GAN [11], which uses a discriminator combined with a generative strategy to output real text samples for the task at hand. But they require complex and time-consuming Monte Carlo tree search, whereas we utilize a dynamic selection mechanism, and the training objective of the discriminator is exactly the same as that of the downstream task.

3 Methodology

3.1 cooperative or adversarial

Following the principle of self-consistency outlined in [12], a closed-loop training needs to be built between the generator and the discriminator, either cooperatively or adversarially. GANs are typical examples of adversarial learning, but training GANs remains quite unstable. Let us consider an extreme case to show the possible instability: the discriminator can perfectly distinguish real data and fake data generated by the generator, and the generator can fully reproduce the real data distribution. Then the discriminator has only a 50% probability of selecting all samples that are generated by the generator. Therefore, any further updates to the generator parameters based on the feedback from the discriminator deviate the generator from the optimum. Neither the generator nor the discriminator can likely be optimal [13]. In practice, a very delicate balance needs to be maintained between the discriminator and the generator to keep the training stable. In terms of cooperatively closed-loop learning, as discussed below, it does not suffer from instability: the generator and the discriminator usually enhance each other.

3.2 Self-consistent Learning Framework

In this section, we introduce our self-consistent learning (**SCL**) framework.

As shown in Figure 1, our framework, similar to the GANs, consists of a generator and a discriminator model. However, contrasting to the GANs, these two parts in our framework work cooperatively to enhance each other. Specifically, for any given class k, the generator $\mathcal{G}$ now become a conditional generator that takes in an input sentence s_k^a and generate an output sentence s_k^b. The discriminator $\mathcal{D}$ is then responsible for discriminating the sentence using a dynamic threshold $\epsilon_{\mathcal{D}}$. The discriminated sentence is used as positive or negative data for that specific class to continue the training process. Once the new discriminator is trained, the sentence is discriminated again by the new discriminator with a different dynamic threshold $\epsilon_{\mathcal{G}}$. This time only the positive data is passed to the generator as the training data for the new round. In this way, a closed loop of cooperation is formed.

In the above closed-loop training, we propose a **selection mechanism** that uses dynamic thresholds to filter samples. This mechanism is empirically shown to play a critical role in closing the gap between the generator and the discriminator, and thus makes this cooperation loop a virtuous circle. Specifically, as shown in Equation 1, the output probability $p_{\mathcal{D}}(y = k|s_k^b)$ that the sentence $\{s_k^b\}$ belongs to class k is calculated from the embedding representation $\mathbf{h}$[1] of $\{s_k^b\}$,

$$p_{\mathcal{D}}(y = k|s_k^b) = \text{softmax}(\text{MLP}(\mathbf{h})) \tag{1}$$

where y represents the class label. Then, through the filtering function $\texttt{filter}_k^{(t)}(\cdot)$ in round t for the k-th class in Equation 2, we keep samples whose output probability is not less than threshold $\epsilon_{t,k}$, while other generated samples whose confidence is lower than threshold $\epsilon_{t,k}$ are discarded.

$$\texttt{filter}_k^{(t)}(s_k^b) \triangleq p_{\mathcal{D}}(k|s_k^b) \geq \epsilon_k^t \tag{2}$$

where $\epsilon_{t,k}$ represents the dynamic threshold for accepting $\{s_k^b\}$ as negative or positive samples in the t-th round. The generalized threshold function for ϵ_k^t is defined as,

$$\epsilon_k^t = f(t, \mathcal{L}_{t-1,k}, \epsilon_k^{t-1}) \tag{3}$$

where $\mathcal{L}_{t-1,k}$ and ϵ_k^{t-1} represent the discriminator loss and threshold for round $t-1$, respectively. $\mathcal{L}_{0,k}$ is set as 0 and $\epsilon_k^0 = \lambda$, where λ represents a hyperparameter.

Theorem 1 *At round t, given the previous round discriminator $\mathcal{D}_\phi^{t-1}$, the aim of the optimization of the generator $\mathcal{G}_\theta^t$, boils down to,*

$$\min_\theta \mathbb{D}_{KL}(p^k_{\mathcal{D}_\phi^{t-1}}(\cdot), p^k_{\mathcal{G}_\theta^t}(\cdot))$$

where $\mathbb{D}_{KL}$ is the standard KL divergence, $p^k_{\mathcal{G}_\theta^t}(\cdot)$ refers to the degree of confidence that the sentences generated by the generator belong to a given class k (we can either train the generator to express its confidence in the generated sentences or use a fixed third-party model to score them), and $p^k_{\mathcal{D}_\phi^{t-1}}(\cdot)$ the probability of being classified into class k given by the discriminator.

Theorem 1 shows that the generator at round t is encouraged to approximate the probability distribution given by the previous round discriminator. In particular, on the basis of a well-pretrained discriminator, the generated distribution of the generator can be guaranteed to be faithful to the real data distribution.

Proof. We use the previous round generator $\mathcal{G}_\theta^{t-1}$ to generate samples, and filter them using the previous round discriminator $\mathcal{D}_\phi^{t-1}$ with a threshold ϵ^{t-1}, then these samples are used for the training of the current round generator $\mathcal{G}_\theta^t$. Therefore, the optimization of $\mathcal{G}_\theta^t$ will tend to maximize the probability that the generated samples pass the discrimination for the fixed $\mathcal{D}_\phi^{t-1}$. For a given class k, we have

$$\max_\theta \mathbb{E}_{x\sim p^k_{\mathcal{G}_\theta^{t-1}}} p^k_{\mathcal{G}_\theta^t}(x) \quad \texttt{s.t.} \quad \texttt{filter}_k^{(t-1)}(x) = 1$$

where the definition of function $\texttt{filter}_k^{(t-1)}(\cdot)$ has been given in Equation 2.

The above objective is equivalent to sampling from the generator being optimized in round t and making these samples pass the discrimination in round $t-1$ as much as possible, which gives

$$\max_\theta \mathbb{E}_{x\sim p^k_{\mathcal{G}_\theta^t}} p^k_{\mathcal{D}_\phi^{t-1}}(x)$$

where $p^k_{\mathcal{D}_\phi^{t-1}}(x)$ is fixed.

A further transformation of the formula shows that

$$\begin{aligned}
&\max_\theta \mathbb{E}_{x\sim p^k_{\mathcal{G}_\theta^t}} p^k_{\mathcal{D}_\phi^{t-1}}(x) \\
&\overset{(i)}{\Rightarrow} \max_\theta \int \mathrm{d}\boldsymbol{\theta} \nabla_{\boldsymbol{\theta}} \mathbb{E}_{x\sim p^k_{\mathcal{G}_\theta^t}} p^k_{\mathcal{D}_\phi^{t-1}}(x) \\
&\overset{(ii)}{\Rightarrow} \max_\theta \int \mathrm{d}\boldsymbol{\theta} \mathbb{E}_{x\sim p^k_{\mathcal{G}_\theta^t}} \nabla_{\boldsymbol{\theta}} \log p^k_{\mathcal{G}_\theta^t}(x) p^k_{\mathcal{D}_\phi^{t-1}}(x) \\
&\overset{(iii)}{\Rightarrow} \max_\theta \int \mathrm{d}\boldsymbol{\theta} \nabla_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^N \{\log p^k_{\mathcal{G}_\theta^t}(x_i) p^k_{\mathcal{D}_\phi^{t-1}}(x_i) \\
&\quad - \log p^k_{\mathcal{D}_\phi^{t-1}}(x_i) p^k_{\mathcal{D}_\phi^{t-1}}(x_i)\} \\
&\overset{(iv)}{\Rightarrow} \min_\theta \mathbb{D}_{KL}(p^k_{\mathcal{D}_\phi^{t-1}}(\cdot), p^k_{\mathcal{G}_\theta^t}(\cdot))
\end{aligned}$$

where (i) uses the integral property that integrating the derivative of a function gives the original function along with a constant, (ii) takes advantage of the derivative property of the logarithmic function, (iii) approximates the expectation of the probability distribution $p^k_{\mathcal{G}_\theta^t}(\cdot)$ by using averaging on N samples sampling from $p^k_{\mathcal{G}_\theta^t}(\cdot)$, and adding a constant term $-\log p^k_{\mathcal{D}_\phi^{t-1}}(\cdot) p^k_{\mathcal{D}_\phi^{t-1}}(\cdot)$ with respect to θ under the summation would not change its derivative, and (iv) cancels out the integral and the derivative and uses the definition of KL divergence. The above concludes our proof.

Why Cooperative, Not Adversarial? (1) the generator is no longer a challenger to the discriminator that only provides negative data points to fool it, but now serves as a data augmenter to provide both positive and negative data points to enhance the discriminator; (2) the generator no longer updates its parameters through the policy gradients guided by the signals from the discriminator, but rather by utilizing the filtered data points to further improve its conditional generation quality. Note that by deliberately choosing the conditional generation paradigm along with the selection mechanism, we not only make the training more stable due to the different training goals, but also mitigate the mode collapse problem of GANs. Besides, by iterating through the loops, our framework achieves self-consistency by honing the domain specificity of the generator and increasing the domain data exposure of the discriminator.

3.3 Text Generation

We leverage the four text generation tasks ($i.e.$ $k = 2$) as an example to demonstrate the effectiveness of our method. At this time, corresponding to Equation 2, $k = 1/0$ represents the positive/negative class, and $\texttt{filter}_{1/0}^{(t)}$ represents the filter function in round t for the

[1] We follow [14] and use the embedding representation of CLS-token as the sentence representation $\mathbf{h}$.

positive/negative class respectively. First, let us introduce the formal definition of this task. Given two sentences $s^a = \{w_1^a, w_2^a, ..., w_{\ell_a}^a\}$ and $s^b = \{w_1^b, w_2^b, ..., w_{\ell_b}^b\}$, where w_i^a and w_j^b represent the i-th and j-th tokens in the sentences, and ℓ_a and ℓ_b indicate the length of s^a and s^b. The goal of this task is to learn a discriminator $\mathcal{D}$ to precisely predict the label $y = \mathcal{D}(s^a, s^b)$, where $y \in \mathcal{Y} = \{0, 1\}$ indicates whether the two sentences are similar.

In our task, $\mathcal{G}$ is trained to generate a similar sentence s^b from any given sentence s^a and $\mathcal{D}$ is trained to predict label y from any given sentence pair $\{s^a, s^b\}$. As demonstrated in Figure 1, there are mainly two training processes in the entire framework: fix $\mathcal{G}$ to train $\mathcal{D}$ and fix $\mathcal{D}$ to train $\mathcal{G}$. We introduce the two training procedures in detail with the t-th round training.

Training $\mathcal{D}$: We first randomly sample s_t^a from domain-related corpus C, and then input s_t^a to $\mathcal{G}^t$ to generate s_t^b. Next, we feed sentence pair $\{s_t^a, s_t^b\}$ into $\mathcal{D}^{t-1}$ to predict the label y_{t-1}, and filter $\{s_t^a, s_t^b, y_{t-1}\}$ using threshold $\epsilon_{\mathcal{D}}^{t-1}$. Finally, we train $\mathcal{D}^{t-1}$ on the selected data and pre-training data P to get an improved discriminator $\mathcal{D}^t$. Note that the filtered data have both positive and negative samples. The update process of $\mathcal{D}$ seeks to minimize the cross-entropy loss over all instances:

$$\mathcal{L}_{\mathcal{D}}(\boldsymbol{s}, \boldsymbol{y}) = \frac{1}{|\boldsymbol{s}|}\sum_{i=1}^{|\boldsymbol{s}|} -[y_i \cdot \log p_{\mathcal{D}}(y_i = 1|s_i^a, s_i^b) + (1 - y_i) \cdot \log(1 - p_{\mathcal{D}}(y_i = 1|s_i^a, s_i^b))] \tag{4}$$

Training $\mathcal{G}$: We feed the generated sentence pairs $\{s_t^a, s_t^b\}$ into $\mathcal{D}^t$ to predict new labels y_t, and then filter $\{s_t^a, s_t^b, y_t\}$ using threshold $\epsilon_{\mathcal{G}}^t$ and additional rules [2]. Note that the filtered data has only positive samples. For the filtered data, we supplement it with the pre-training data P to update $\mathcal{G}^t$ to $\mathcal{G}^{t+1}$ [3] We also take out s_t^b from the filtered data and add them to the domain-related corpus. The expanded domain corpus are used to sample conditional sentences in the next round of generation. The update procedure of $\mathcal{G}$ employs the negative log-likelihood function over all instances:

$$\mathcal{L}_{\mathcal{G}}(\boldsymbol{s^a}, \boldsymbol{s^b}) = -\frac{1}{|\boldsymbol{s^b}|}\sum_{t=1}^{|\boldsymbol{s^b}|} \log p_{\mathcal{G}}(s_t^b | s_{<t}^b, \boldsymbol{s^a})$$

For the selection mechanism, we adopt the form $\epsilon^t = m * t + \lambda$ after comparing the effects of different threshold functions through experiments according to Equation 3, where m is the increment of the threshold for each round, λ is the initial threshold, and ϵ^t is the threshold for rounds t.

In the process of training $\mathcal{G}$, since the sentences generated in each round are added to the domain-related corpus, the source of domain-specific data is thus monotonically expanding by iterating the self-consistent learning loop. The formalized process is shown in Algorithm 1.

Algorithm 1 Self-consistent Learning (**SCL**)

```
Require: Generator $\mathcal{G}$; Discriminator $\mathcal{D}$; Domain-Related Corpus $C$; Pre-training Data $P$.
Initialize $\mathcal{G}^0$ and $\mathcal{D}^0$ with pre-trained language models;
Warm-up $\mathcal{G}^0$ and $\mathcal{D}^0$ with pre-training data $P$ to get $\mathcal{G}^1$ and $\mathcal{D}^1$;

for each round $i \in [1, n]$ do
    if Two consecutive rounds of discriminator still improve then
        Generate similar sentences $s^b \sim p_{\mathcal{G}^i}(\cdot|s^a)$ from sampled sentences $s^a$ from $C$;
        Predict pseudo-labels $y^i \sim p_{\mathcal{D}^i}(\cdot|s^a, s^b)$;
        Use threshold $\epsilon_{\mathcal{D}}^i$ to select data on $\{s^a, s^b, y^i\}$ to train $\mathcal{D}^{i+1}$;
        Predict pseudo-labels $y^{i+1} \sim p_{\mathcal{D}^{i+1}}(\cdot|s^a, s^b)$;
        Use threshold $\epsilon_{\mathcal{G}}^i$ and additional rules to select data on $\{s^a, s^b, y^{i+1}\}$ to train $\mathcal{G}^{i+1}$;
    end if
end for
```

4 Experiments

4.1 Tasks Design

In our experiments, the pre-training datasets are used to warm up the discriminator and generator, and the domain-related corpus is a set of independent sentences. To avoid label leakage, none of the training datasets participate in the pre-training of the generator and discriminator. In other words, the datasets in pre-training and self-consistent training are two non-overlapped datasets.

Zero-Shot Baseline: In the zero-shot setting, we utilize the warm-up generator and employ the constructed prompts to directly generate samples without any specific learning towards the prediction targets. These samples are then filtered by the discriminator and used as training data for the next round of the generator.We utilize the best-performing Chinese model RoBERTa-wwm-ext-large [15] and English model ALBERT-xxlarge-v2 [16] as the base discriminators in our self-consistent learning framework.

Fine-Tune Baseline: In the fine-tuning setting, similar sentence pairs like $< s_a, s_b >$ are used as training data for the generator in the form of "s_a/s_b is similar to s_b/s_a". Here, "s_a/s_b is similar to" serves as the prompt, and s_b/s_a is the target that the generator will learn to predict. We compare our model with several strong baselines Chinese models MacBERT , StructBERT , RoFormer , XLNet, ELECTRA, ALBERT, RoBERTa and English models BERT, XLM-RoBERTa (XLM-R), XLNet, ELECTRA, ALBERT, RoBERTa.

4.2 Experiments Setup

4.2.1 Datasets

We conduct experiments on three Chinese datasets AFQMC (Financial) [17], CHIP-STS (Medical) [18], QQP-ZH (Common) [19] and an English dataset MRPC (News) [19]. More details about the datasets are given in supplementary material.

4.3 Zero-Shot Results

Table 1(a) shows how the F1 score of the discriminator varies with the number of self-consistent learning rounds on different datasets in the zero-shot task. According to Algorithm 1, the training is stopped when the discriminator no longer improves for two consecutive rounds. In addition, these four datasets are collected from different domains to further reflect the generality of our method in different domains.

The scores in the last line of Table 1(a) give the improvement of our discriminator in the last round relative to the first round. We can see that the F1 score gradually increases after each training round,

[2] The additional rules are used to exclude sentences which are too long, too short, or too similar according to the longest common substring algorithm.

[3] Note that the pre-training data P is used to warm up $\mathcal{G}$ and $\mathcal{D}$. Although pre-training data is not mandatory in subsequent training, we empirically found that including it when training $\mathcal{G}$ can prevent language degeneration and improve downstream performances.

Table 1: Results of Cooperative Training through Self-Consistent Learning.

(a) F1 Score of Discriminator in Zero-Shot Setting.

Round	AFQMC	CHIP-STS	QQP-ZH	MRPC
0	38.25	58.82	57.88	68.54
1	39.61	62.89	60.08	75.47
2	44.98	67.24	58.57	76.63
3	45.99	71.38	60.30	83.00
4	45.71	71.45	61.31	83.90
5	48.01	74.06	64.47	84.24
6	50.41	74.08	66.44	84.50
7	50.68	76.66	63.88	84.32
8	**51.36**	76.30	65.46	**84.61**
9	-	76.67	68.08	-
10	-	**77.42**	**70.51**	-
	+13.11	**+18.60**	**+12.63**	**+16.07**

(b) F1 Score of Discriminator in Fine-Tune Setting.

Method	AFQMC	CHIP-STS	QQP-ZH	MRPC
BERT*large*	-	-	-	82.51
XLM-R*base*	-	-	-	84.27
MacBERT*large*	61.11	85.94	72.94	-
StructBERT*large*	60.56	85.17	76.33	-
RoFormer*large*	64.19	84.16	76.56	-
XLNet*large*	50.31	82.97	64.96	79.51
ELECTRA*large*	54.59	84.97	71.81	89.64
ALBERT*large*	56.87	86.32	70.52	91.21
RoBERTa*large*	57.29	86.93	74.58	90.24
Self-Consistent	**66.59**	**88.39**	**78.43**	**92.78**

eventually reaching a 10+ absolute percentage (AP) improvement. We believe what drives the improvement of the discriminator is the self-consistency, which it acquires with the generator step by step during the loop.

To verify that the generator also improves after self-consistent training, we adopt Perplexity and Bertscore to measure the language fluency and the semantic similarity (i.e. domain specificity) respectively. For different generators in different rounds, we first select s^a in similar sentence pairs from the same test set as the original sentences input, and generate similar sentences s^b with greedy search. The reason for not using other sampling methods is to ensure reproducibility. Given the generated sentences, we introduce an additional GPT2 [4] model to calculate the perplexity of generated similar sentences, and use a third-party library [5] to calculate the bertscore between the original and generated similar sentences. The results are shown in Table 2.

Table 2: Zero-Shot Performance of Generator in Zero-Shot Setting.

	AFQMC	CHIP-STS	QQP-ZH	MRPC
Perplexity ↓ *-first round*	10.13	6.86	12.94	28.71
Perplexity ↓ *-last round*	8.43	5.97	12.27	17.56
Bertscore ↑ *-first round*	0.79	0.84	0.87	0.94
Bertscore ↑ *-last round*	0.80	0.85	0.89	0.97

We can see that the perplexity / bertscore of the last round in Table 2 has decreased / improved compared to the first round. Note that a lower perplexity indicates a more fluent sentence, while a higher bertscore indicates a more similar sentence. It suggests that after self-consistent training, the generator is gradually improved in language fluency and semantic similarity (i.e. domain specificity). The reason why the improvement of the generator is not as obvious as that of the discriminator is that the size of the generator is several times that of the discriminator, and the total number of training samples is limited. In supplementary material, the generated samples of the generator in different rounds are given to show the changes in the generation.

4.4 Fine-Tune Results

Our method not only works well in the zero-shot case, but also achieves good results in the full-data case. For the sake of a fair comparison, we reproduce several strong baselines on the four training sets, and their performances on the test sets are shown in Table 1(b).

Our approach uses the best-performing model on a single test set as the base discriminator for self-consistent learning. The bold scores in the last line of Table 1(b) show that our method outperforms the strong baselines (shaded in gray) by 1 to 2 AP on all four test datasets, indicating the potential of self-consistent learning to further improve the model performance.

4.5 Evaluating Self-consistency

In this section, we evaluate the consistency between the generator and the discriminator as the learning loop unfolds. We follow the same method used in Section 4.3 and use greedy search to generate similar sentences on the same test set. Then we take the confidence of the discriminator $R_{\mathcal{D}}$ as the score of the discriminator, which is calculated for the original sentences s^a and the generated similar sentences s^b according to Equation 5.

$$R_{\mathcal{D}} = p_{\mathcal{D}}(y^+|s^a, s^b) \tag{5}$$

where y^+ represents a positive label.

For the generator, using its own perplexity as a criterion for determining the similarity between sentences s^a and s^b is not always effective. Perplexity primarily reflects the generator's ability to fit similar data pairs, but it falls short in mitigating the impact of noise pairs. Therefore, to quantify this similarity, we introduce a third-party static model SimCSE [6] to get the embedding representation $\mathbf{a}, \mathbf{b}$ of sentences s^a, s^b. The cosine similarity $R_{\mathcal{G}}$ between $\mathbf{a}$ and $\mathbf{b}$ is then calculated according to Equation 6 to approximate the score of the generator.

$$\mathbf{a}, \mathbf{b} = \text{Encoder}(s^a), \text{Encoder}(s^b)$$
$$R_{\mathcal{G}} = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|_2 * \|\mathbf{b}\|_2} \tag{6}$$

where $\mathbf{a}$ and $\mathbf{b}$ both represent the embedding representation at the $[CLS]$ position. Note that the original sentence s^a remains un-

[4] Wenzhong-GPT2-110M for Chinese data, and GPT2-base for English data.
[5] https://pypi.org/project/bert-score/

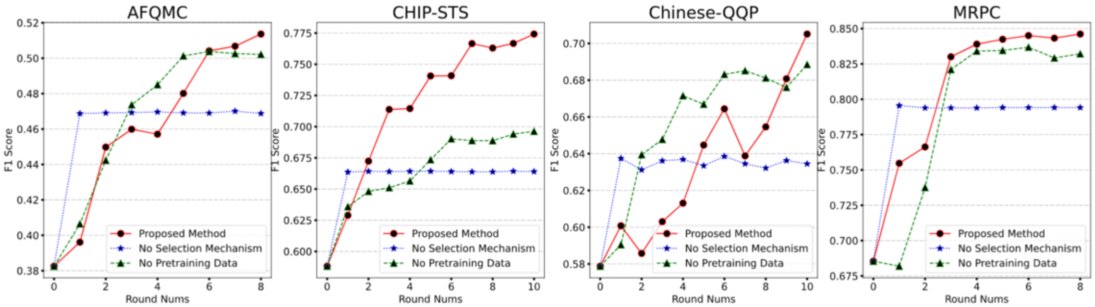

Figure 2: Results of ablation experiments on pre-training data and selection mechanism of Zero-Shot. Results of the proposed method, without pre-training data, and without the selection mechanism are given in red, green, and blue, respectively.

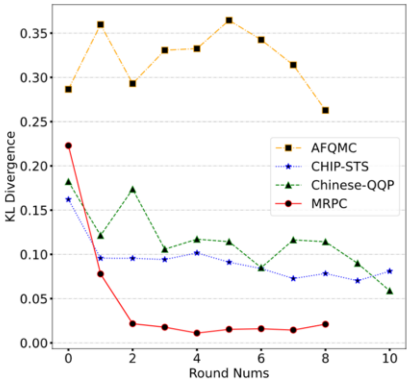

Figure 3: The KL Divergence between the score distributions of Discriminator and Generator in Zero-Shot.

changed in each round, while the generated sentence s^b changes.

Finally, for the trained discriminator and generator in each round t, we can obtain two score distributions $\mathbf{R}_{\mathcal{D}}^{\mathbf{t}}$ and $\mathbf{R}_{\mathcal{G}}^{\mathbf{t}}$ correspondingly. According to Theorem 1, we draw the curves of KL divergence between $\mathbf{R}_{\mathcal{D}}^{\mathbf{t}}$ and $\mathbf{R}_{\mathcal{G}}^{\mathbf{t}}$ in each round for the four datasets: AFQMC, CHIP-STS, QQP-ZH, and MRPC. As illustrated in Figure 3, all the curves show a clear downward trend, indicating that the distance between the two score distributions decreases with the increase in the number of training rounds until a score consensus is reached.

4.6 *Effect of Pre-training Data and Selection Mechanism*

We perform ablation experiments on the pre-training data and the selection mechanism in the zero-shot case. As described in Section 4.1, the pre-training data is used to pre-train the generator and discriminator, completely independent of the experimental datasets in self-consistent training.

To explore the influence of pre-training data on self-consistent training, we no longer add it in each round when training the discriminator, and only the generated data is used. But when the generator is trained, pre-training data is still retained to prevent language degeneration and lack of expressive diversity of the generation. The result of removing pre-training data is shown as the green curves in Figure 2. With all other training parameters being the same, after the same number of training rounds, the discriminator is slightly worse compared to the original method (red curves in Figure 2). However, the green curves maintain an upward trend and are very close to the red curves in all datasets except CHIP-STS. This shows that the generated data plays a key role in continuously improving the discriminator, while the pre-training data has a limited role.

In order to explore the effect of the selection mechanism on training the discriminator, we remove the selection mechanism when training the discriminator, while the training of the generator remains unchanged. The blue curves in Figure 2 depict the performance of the discriminator in each round after removing the selection mechanism. Compared to the original method (red curves), the discriminator only improves in the first round after removing the selection mechanism, which demonstrates the importance of the selection mechanism on the discriminator for the convergence of the self-consistent learning framework.

4.7 *Experiments on Different Threshold Functions*

To compare the effect of different threshold functions on the final result, we use four type of functions, including oscillatory function (cosine), constant function and monotonically increasing functions (quadratic and linear). For the fairness of comparison, we keep the maxima and minima the same for all functions(except for the constant threshold), and the values are given in supplementary material.

The best results and the second-best results are **bold** and underlined, respectively. As can be seen from the Table 3, in the zero-shot setting, the chosen linear function outperforms the other functions, and all the threshold functions show an averaging 10+ AP improvement relative to the baseline. Therefore, the self-consistent learning framework makes it easy to choose a certain threshold function and perform well, and the results are not so sensitive to the choice of the functions. A more detailed figure of the effect of different threshold functions on the results is shown in supplementary material.

Table 3: F1 Score of Different Threshold Functions in Zero-Shot.

	AFQMC	CHIP-STS	QQP-ZH	MRPC	AVG
Baseline	38.25	58.82	57.88	68.54	55.87
Cosine	47.38	74.26	64.39	83.48	67.38
Constant	47.06	74.15	68.67	84.11	68.50
Quadratic	**51.75**	73.09	**70.85**	83.48	69.79
Linear	51.36	**77.42**	70.51	**84.61**	**70.98**

Table 4 shows the effects of different threshold functions in the fine-tune experiment. It can be seen that all functions have a $1 \sim 2$

[6] We use SimCSE-BERT-base to calculate scores on Chinese datasets and sup-SimCSE-BERT-base-uncased on English datasets.

Table 4: F1 Score of Different Threshold Functions in Fine-Tune.

	AFQMC	CHIP-STS	QQP-ZH	MRPC	AVG
Baseline	64.19	86.93	76.56	91.21	79.72
Cosine	66.43	88.01	77.33	92.63	81.10
Constant	66.57	88.15	78.45	92.51	81.42
Quadratic	66.37	87.76	**79.26**	92.75	81.54
Linear	**66.59**	**88.39**	78.43	**92.78**	**81.55**

AP increase relative to the baseline, and the chosen linear function achieves the best performance on all datasets except QQP-ZH.

4.8 *Contrast Experiments with Adversarial Training*

We further demonstrate the superiority of the cooperative approach by comparing the results with adversarial experiments. All experimental settings independent of the training method remain the same in the adversarial training.

During the experiments, the generator is no longer trained using the samples filtered by the discriminator, but the rewards passed by the discriminator assist the training. All generated samples are treated as negative samples when training the discriminator.

Specifically, $\mathcal{G}$ takes the prompt ' "s^a" is similar to " ' and the first M tokens of s^b as input to get M sentence pairs $< s^a, s^b_m >$, where m is from 1 to M. Note that we repeat the process of generating sentences N times to reduce the negative impact caused by the large variance of the rewards.[7] The sentence pair is formalized as

$$< s^a, s^b_m >= \mathcal{G}_\theta(s^b_m | s^b_{<m}, \boldsymbol{s^a}; N)$$

Once the $M * N$ sentence pairs are generated, they are passed as input to the $\mathcal{D}$ to obtain the probability score Q^n_m for each of them. We take the average of Q^n_m over N as the reward $\bar{Q}_m$ corresponding to the m-th token. If the sentence length of s^b is greater than M, the rewards of the remaining tokens are all the same as those of the M-th token. Taking the m-th token as an example, the rewards $\bar{Q}_m$ can be formalized as

$$\bar{Q}^{\mathcal{G}_\theta}_{\mathcal{D}_\phi}(m) = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} \mathcal{D}_\phi(g^n_m) & m \le M \\ \bar{Q}(M) & m > M \end{cases}$$

where g^n_m is the n-th sentence pair with length m.

Therefore, the objective function for training the generator $\mathcal{G}$ is,

$$\mathcal{L}_\mathcal{G}(\boldsymbol{s^a}, \boldsymbol{s^b}) = -\frac{1}{|\boldsymbol{s^b}|}\sum_{t=1}^{|\boldsymbol{s^b}|} \log(p_\mathcal{G}(s^b_t | s^b_{<t}, \boldsymbol{s^a}) * \bar{Q}_t)$$

The loss function of training the discriminator remains the same as Equation 4, but differing from cooperative training, the generated samples are regarded as negative samples to the discriminator, and the training target for the discriminator can be given by

$$\min_\phi -\mathbb{E}_{X\sim p_{\text{data}}}\left[\log \mathcal{D}_\phi(X)\right] - \mathbb{E}_{X\sim p_{\mathcal{G}_\theta}}\left[\log\left(1-\mathcal{D}_\phi(X)\right)\right]$$

The results of zero-shot and fine-tune on the four datasets are shown in Tables 5 and 6.

As can be seen from Table 5, in the zero-shot setting, training in an adversarial manner does not give any improvement over the baseline. Because the initial discriminator in the zero-shot setting is very weak in distinguishing positive and negative samples, it is reasonable to believe that if all generated samples are considered negative samples from the very beginning, it is difficult for the discriminator to know how to distinguish positive samples. As a result, the F1 scores on both AFQMC and CHIP-STS datasets end up being 0, while the scores on the QQP-ZH and MRPC datasets fluctuate intensively with the number of rounds, which further validates the instability of the adversarial training in the zero-shot setting.

Table 5: F1 Score of Adversarially Trained Discriminator in Zero-Shot Setting.

Round	AFQMC	CHIP-STS	QQP-ZH	MRPC
0	38.25	58.82	57.88	68.54
1	0.0	8.73	21.71	4.19
2	0.02	7.13	49.30	7.06
3	0.0	0.29	42.94	5.32
4	0.0	1.09	41.13	0.0
5	0.0	0.10	43.10	1.72
6	0.0	0.39	34.30	67.38
7	0.0	0.20	42.62	48.31
8	0.0	0.20	34.95	37.97
9	-	0.20	41.81	-
10	-	0.20	40.00	-

Table 6: F1 score of the Discriminator in Fine-Tune Setting.

	AFQMC	CHIP-STS	QQP-ZH	MRPC	AVG
Baseline	64.19	86.93	76.56	91.21	79.72
Adversarial	58.37	80.46	77.93	92.18	77.24
Cooperative (OUR METHOD)	**66.59**	**88.39**	**78.43**	**92.78**	**81.55**

For the fine-tune experiments, Table 6 shows that training in an adversarial manner can slightly improve the performance on the QQP-ZH and MRPC datasets, but is still worse than the cooperative training. On the AFQMC and CHIP-STS dataset, adversarial training makes it even worse relative to the baseline. It is worth noting that the whole process of adversarial training is so unstable and it is easy to collapse after a few training rounds.

5 Conclusion

In this paper, we propose a self-consistent learning framework in the text field to enable cooperative training of the generator and the discriminator. During the training process, the generator and the discriminator continuously enhance each other until reaching a score consensus. This framework can utilize both limited labeled data and large-scale unlabeled domain-related corpus. Experimental results on four Chinese / English datasets demonstrate that as a form of closed-loop training, our proposed framework can outperforms the strong baselines with continuously improved generators and discriminators.

6 Acknowledgements

This research is supported by National Natural Science Foundation of China (Grant No.62276154), Research Center for Computer Network (Shenzhen) Ministry of Education, Beijing Academy of Artificial Intelligence (BAAI), the Natural Science Foundation of Guangdong Province (Grant No. 2023A1515012914), Basic Research Fund of Shenzhen City (Grant No. JCYJ20210324120012033

[7] In practice, we take $M = 5, N = 5$ for ease of calculation.

and JSGG20210802154402007), the Major Key Project of PCL for Experiments and Applications (PCL2021A06), and Overseas Cooperation Research Fund of Tsinghua Shenzhen International Graduate School (HW2021008).

References

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[2] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[4] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[5] Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. Language gans falling short. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[6] Amin Banitalebi-Dehkordi and Yong Zhang. Repaint: Improving the generalization of down-stream visual tasks by generating multiple instances of training examples. In *32nd British Machine Vision Conference 2021, BMVC 2021, Online, November 22-25, 2021*, page 122. BMVA Press, 2021.

[7] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in neural information processing systems*, 33:596–608, 2020.

[8] Qingyang Wu, Lei Li, and Zhou Yu. Textgail: Generative adversarial imitation learning for text generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14067–14075, 2021.

[9] Sidi Lu, Lantao Yu, Siyuan Feng, Yaoming Zhu, and Weinan Zhang. CoT: Cooperative training for generative modeling of discrete data. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4164–4172. PMLR, 09–15 Jun 2019.

[10] Yanghoon Kim, Seungpil Won, Seunghyun Yoon, and Kyomin Jung. Collaborative training of gans in continuous and discrete spaces for text generation. *IEEE Access*, 8:226515–226523, 2020.

[11] Sylvain Lamprier, Thomas Scialom, Antoine Chaffin, Vincent Claveau, Ewa Kijak, Jacopo Staiano, and Benjamin Piwowarski. Generative cooperative networks for natural language generation. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 11891–11905. PMLR, 17–23 Jul 2022.

[12] Yi Ma, Doris Tsao, and Heung-Yeung Shum. On the principles of parsimony and self-consistency for the emergence of intelligence. *Frontiers of Information Technology & Electronic Engineering*, pages 1–26, 2022.

[13] Martín Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[14] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics.

[15] Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. Revisiting pre-trained models for Chinese natural language processing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 657–668, Online, 2020. Association for Computational Linguistics.

[16] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[17] Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, Yin Tian, Qianqian Dong, Weitang Liu, Bo Shi, Yiming Cui, Junyi Li, Jun Zeng, Rongzhao Wang, Weijian Xie, Yanting Li, Yina Patterson, Zuoyu Tian, Yiwen Zhang, He Zhou, Shaoweihua Liu, Zhe Zhao, Qipeng Zhao, Cong Yue, Xinrui Zhang, Zhengliang Yang, Kyle Richardson, and Zhenzhong Lan. CLUE: A Chinese language understanding evaluation benchmark. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4762–4772, Barcelona, Spain (Online), 2020. International Committee on Computational Linguistics.

[18] Ningyu Zhang, Mosha Chen, Zhen Bi, Xiaozhuan Liang, Lei Li, Xin Shang, Kangping Yin, Chuanqi Tan, Jian Xu, Fei Huang, Luo Si, Yuan Ni, Guotong Xie, Zhifang Sui, Baobao Chang, Hui Zong, Zheng Yuan, Linfeng Li, Jun Yan, Hongying Zan, Kunli Zhang, Buzhou Tang, and Qingcai Chen. CBLUE: A Chinese biomedical language understanding evaluation benchmark. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7888–7915, Dublin, Ireland, 2022. Association for Computational Linguistics.

[19] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.