Pre-training Language Model as a Multi-perspective Course Learner

Beiduo Chen[§‡*], **Shaohan Huang**[‡†], **Zihan Zhang**[‡], **Wu Guo**[§], **Zhenhua Ling**[§], **Haizhen Huang**[‡], **Furu Wei**[‡], **Weiwei Deng**[‡] **and Qi Zhang**[‡]

[§] National Engineering Research Center of Speech and Language Information Processing, University of Science and Technology of China, Hefei, China

[‡] Microsoft Corporation, Beijing, China

beiduo@mail.ustc.edu.cn, {guowu,zhling}@ustc.edu.cn,
{shaohanh,zihzha,hhuang,fuwei,dedeng,qizhang}@microsoft.com

Abstract

ELECTRA (Clark et al., 2020), the generator-discriminator pre-training framework, has achieved impressive semantic construction capability among various downstream tasks. Despite the convincing performance, ELECTRA still faces the challenges of monotonous training and deficient interaction. Generator with only masked language modeling (MLM) leads to biased learning and label imbalance for discriminator, decreasing learning efficiency; no explicit feedback loop from discriminator to generator results in the chasm between these two components, underutilizing the course learning. In this study, a multi-perspective course learning (MCL) method is proposed to fetch a many degrees and visual angles for sample-efficient pre-training, and to fully leverage the relationship between generator and discriminator. Concretely, three self-supervision courses are designed to alleviate inherent flaws of MLM and balance the label in a multi-perspective way. Besides, two self-correction courses are proposed to bridge the chasm between the two encoders by creating a "correction notebook" for secondary-supervision. Moreover, a course soups trial is conducted to solve the "tug-of-war" dynamics problem of MCL, evolving a stronger pre-trained model. Experimental results show that our method significantly improves ELECTRA's average performance by 2.8% and 3.2% absolute points respectively on GLUE and SQuAD 2.0 benchmarks, and overshadows recent advanced ELECTRA-style models under the same settings. The pre-trained MCL model is available at https://huggingface.co/McmanusChen/MCL-base.

1 Introduction

Language models pre-training (Radford et al., 2018, 2019; Devlin et al., 2019; Liu et al., 2019) has shown great success in endowing machines with the ability to understand and process various downstream NLP tasks. A wide range of pre-training strategies have been proposed, among which the most prevailing object is the masked language modeling (MLM) (Devlin et al., 2019). Such autoencoding language modeling objects (Vincent et al., 2008) typically first randomly corrupt a certain percentage of training corpus with masked tokens, and then encourage encoders to restore the original corpus. To reduce the randomness of pre-training and produce a sample-efficient method, ELECTRA-style frameworks (Clark et al., 2020) leverage an Transformer-based (Vaswani et al., 2017) generator training with MLM to build challenging ennoising sentences for a discriminator in the similar structure to carry out the denoising procedure.

Typically in the ELECTRA-style training, the generator first constructs its semantic representations through MLM training and cloze these masked sentences with pseudo words; in the meanwhile, the discriminator inherits the information from the former and distinguish the originality of every token, which is like a step-by-step course learning. However, only MLM-based generator training may lead to monotonous learning of data, which conduces to the incomprehensive generation and imbalanced label of corrupted sentences for the discriminator (Hao et al., 2021). Besides, interactivity between the two encoders stops abruptly except the sharing of embedding layers (Xu et al., 2020; Meng et al., 2021), since there is no direct feedback loop from discriminator to generator.

To enhance the efficiency of training data and to adequately utilize the relationship of generator and discriminator, in this work, we propose a sample-efficient method named multi-perspective course learning (MCL). In the first phase of MCL, to fetch a many degrees and visual angles to impel initial semantic construction, three self-supervision courses are designed, including cloze test, word rearrangement and slot detection. These courses instruct

[*] Contribution during internship at Microsoft.

[†] Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2023, pages 114–128
July 9-14, 2023

language models to deconstruct and dissect the exact same corpus from multi perspectives under the ELECTRA-style framework. In the second phase, two self-correction courses are tasked to refine both generator and discriminator. A confusion matrix regarding to discriminator's recognition of each sentence is analyzed and applied to the construction of revision corpora. Secondary learning is carried out for the two components in response to the deficiencies in the previous course learning. At last, the model mines the same batch of data from multiple perspectives, and implement progressive semantic learning through the self-correction courses.

Experiments on the most widely accepted benchmarks GLUE (Wang et al., 2019) and SQuAD 2.0 (Rajpurkar et al., 2018) demonstrate the effectiveness of the proposed MCL. Compared with previous advanced systems, MCL achieved a robust advantage across various downstream tasks. Abundant ablation studies confirm that multi-perspective courses encourage models to learn the data in a sample-efficient way. Besides, a course soups trial is conducted to further interpret and dissect the core of multi-perspective learning, providing a novel approach to enhance the pre-training efficiency and performance.

2 Preliminary

In this work, we built our system based on the ELECTRA-style framework. Thus, the framework of ELECTRA is reviewed. Unlike BERT (Devlin et al., 2019), which uses only one transformer encoder trained with MLM, ELECTRA is trained with two transformer encoders: a generator G and a discriminator D. G is trained with MLM and used to generate ambiguous tokens to replace masked tokens in the input sequence. Then the modified input sequence is fed to D, which needs to determine if a corresponding token is either an original token or a token replaced by the generator.

Generator Training Formally, given an input sequence $X = [x_1, x_2, ..., x_n]$, a mask operation is conducted to randomly replace its tokens with `[MASK]` at the position set $\boldsymbol{r}$.[1] And the masked sentence $X^{\text{mask}} = [x_1, x_2, ..., \texttt{[MASK]}_i, ..., x_n]$ is fed into the generator to produce the contextualized representations $\{\boldsymbol{h}_i\}_{i=1}^{n}$. G is trained via the following loss $\mathcal{L}_{\text{MLM}}$ to predict the original tokens from the vocabulary V at the masked positions:

$$p_{\text{MLM}}(x_t|\boldsymbol{h}_i) = \frac{\exp(\boldsymbol{x}_t^\top \boldsymbol{h}_i)}{\sum_{t'=1}^{|V|} \exp(\boldsymbol{x}_{t'}^\top \boldsymbol{h}_i)}, \quad (1)$$

$$\mathcal{L}_{\text{MLM}} = \mathbb{E}\left(- \sum_{i \in \boldsymbol{r}} \log p_{\text{MLM}}\left(x_i \middle| \boldsymbol{h}_i\right) \right), \quad (2)$$

where $\{\boldsymbol{x}_t\}_{t=1}^{|V|}$ are the embeddings of tokens that are replaced by `[MASK]`. Masked language modeling only conducts on the masked positions.

Discriminator Training. G tends to predict the original identities of the masked-out tokens and thus X^{rtd} is created by replacing the masked-out tokens with generator samples:

$$x_i^{\text{rtd}} \sim p_{\text{MLM}}\left(x|\boldsymbol{h}_i\right), \text{ if } i \in \boldsymbol{r}\ ; \quad x_i^{\text{rtd}} = x_i, \text{ else.} \quad (3)$$

D is trained to distinguish whether the tokens in X^{rtd} have been replaced by G via the replaced token detection (RTD) loss $\mathcal{L}_{\text{RTD}}$:

$$p_{\text{RTD}}(x_i^{\text{rtd}} = x_i \big| \boldsymbol{h}_i) = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{h}_i)}{(1 + \exp(\boldsymbol{w}^\top \boldsymbol{h}_i))}, \quad (4)$$

$$\mathcal{L}_{\text{RTD}} = \mathbb{E}\Bigg(- \sum_{x_i^{\text{rtd}} = x_i} \log p_{\text{RTD}}\left(x_i^{\text{rtd}} = x_i \middle| \boldsymbol{h}_i\right) - \sum_{x_i^{\text{rtd}} \neq x_i} \log\left(1 - p_{\text{RTD}}\left(x_i^{\text{rtd}} = x_i \middle| \boldsymbol{h}_i\right)\right) \Bigg), \quad (5)$$

where $\boldsymbol{w}$ is a learnable weight vector. This optimization is conducted on all tokens.

The overall pre-training objective is defined as:

$$\mathcal{L}_{\text{ELECTRA}} = \mathcal{L}_{\text{MLM}} + \lambda \mathcal{L}_{\text{RTD}}. \quad (6)$$

where λ (typically 50) is a hyperparameter used to balance the training pace of G and D. Only D is fine-tuned on downstream tasks after pre-training.

3 Challenges

Biased Learning Though the ELECTRA training method is simple and effective, treating corpora from a single perspective could cause biased learning. As for the progress of MLM and RTD, there exists an inherent flaw that G might predict appropriate but not original token on the `[MASK]` position, and such appropriate expression still needs to be judged as substitution by D. For example, if the original sequence "Alan buys an apple" is

[1] Typically the proportion is set as 15%, which means 15% of the tokens are masked out for each sentence.

masked as "Alan [MASK] an apple", there are too many candidate words like "eats, peels, cuts" that can replace [MASK] to form a harmonious context. Thus, to request D to continue to distinguish replaced tokens is a difficult even awkward task. Treating the same piece of data in a single way reduces the effectiveness of training. As for the distribution of generated corpus from G, the label-imbalance may gradually emerge with the MLM training of G, which could disturb the RTD training of D. As the semantic construction of G thrives with the pre-training, the pseudo token on [MASK] becomes more reasonable and even matches the original word. Thus, the proportion of replaced tokens in the training sentences of D collapses, which interferes with the binary classification task seriously. In this work, three self-supervision courses are tasked to train the model in a multi-perspective way, improving the learning efficiency of data and balancing the distribution of labels.

Deficient Interaction The core of self-supervision training is to ingeniously design and construct labeled data from original corpora. Evolving from random masking as BERT does, ELECTRA provides more realistic corpora by generator samples, encouraging G and D to compete with each other. However, there is no explicit feedback loop from D to G, resulting that the pre-training of G is practically dominated by MLM as before. To bridge the chasm between these two components, in this work, we take advantage of discriminant results from D to create a "correction notebook" for both G and D, and propose two self-correction courses to provide secondary-supervision. Revision-training fully leverages the relationship and characteristics of the two encoders to increase the quality of training.

4 Methodology

In this section, we will start by formulating three self-supervision courses which encourage models to treat data in a multi-perspective way. Then two self-correction courses are elaborated, deriving from the course-like relationship between G and D. These various courses are weaved into the entirety of the multi-perspective course learning method.

4.1 Self-supervision Course

The essentiality of large-scale data pre-training, undoubtedly is to excogitate a way to take full advantage of the massive rude corpora. ELECTRA has

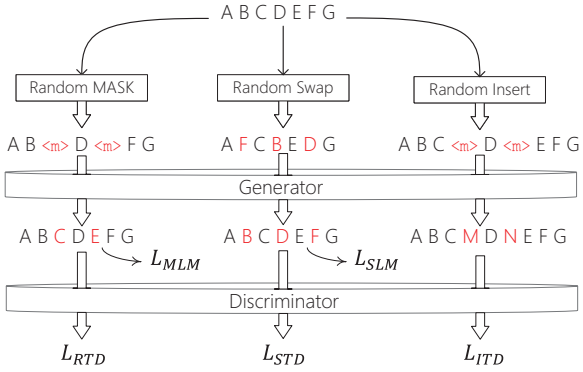

Figure 1: The overall structure of the self-supervision courses. <m> denotes [MASK]. A capital letter stands for a token and letters in red indicate operated positions.

provided an applicable paradigm for models to construct semantic representations through ennoising and denosing. Based on this framework, we extend the perspective that models look at sequences and propose three binary classification tasks in order to improve training efficiency, alleviate biased learning, and balance label distributions.

4.1.1 Replaced Token Detection

On account of the compelling performance of pre-training language models with masked language modeling, we retain the replaced token detection task from ELECTRA. Following the previous symbol settings, given an original input sequence $X = [x_1, x_2, ..., x_n]$, we first mask out it into $X^{\text{mask}} = [x_1, x_2, ..., \texttt{[MASK]}_i, ..., x_n]$, which is then fed into G to get the filling-out sequence $X^{\text{rtd}} = [x_1, x_2, ..., x_i^{\text{rtd}}, ..., x_n]$ by generator samples. Finally, D is tasked to figure out which token is original or replaced. As illustrated in the Section 2, G and D are trained with $\mathcal{L}_{\text{MLM}}$ and $\mathcal{L}_{\text{RTD}}$ respectively. MLM endows G with fundamental contextual semantic construction by cloze test, and RTD is a higher level course for D to let the model drill down into context for seeking out dissonance in the pseudo sequence X^{rtd}.

4.1.2 Swapped Token Detection

Intuitively, recombination tasks contribute to sequence-related learning. As mentioned in Section 3, information absence at the [MASK] position will cause the unreliability of generated pseudo words. Whether the filled sample is appropriate or not, biased learning occurs interfering training of D. Thus, to reserve primeval message for precise prediction, without slashing the degree of task difficulty, we present swapped token detection (STD)

course to sharpen the model's structure perception capability through a word rearrangement task.

For an input sentence $X = [x_1, x_2, ..., x_n]$, a random position set $\boldsymbol{s}$ is chosen for swapping operation.[2] Precisely, tokens at the chosen position are extracted, reordered and filled back into the sentence. G is required to restore the swapped sentence X^{swap} to X, and the adjacent D is tasked to discriminate which token is swapped in X^{std} by generator samples. Note the contextualized representations from G as $\{\boldsymbol{h}_i\}_{i=1}^{n}$, the training process of swapped language modeling (SLM) is formulated below:

$$p_{\text{SLM}}(x_s|\boldsymbol{h}_i) = \frac{\exp(\boldsymbol{x}_s^\top \boldsymbol{h}_i)}{\sum_{s'=1}^{|V|} \exp(\boldsymbol{x}_{s'}^\top \boldsymbol{h}_i)}, \tag{7}$$

$$\mathcal{L}_{\text{SLM}} = \mathbb{E}\left(-\sum_{i \in \boldsymbol{s}} \log p_{\text{SLM}}\left(x_i \middle| \boldsymbol{h}_i\right) \right), \tag{8}$$

where $\{\boldsymbol{x}_s\}_{s=1}^{|V|}$ are the embeddings of tokens at the swapped positions. Note that the vocabulary V is still the same across all courses, because it helps the generation of G in a consistent and natural environment, even the correct answer is lying in the pending sequence during SLM. SLM is only conducted on tokens at the swapped positions.

SLM brings G to making reasonable even original predictions on the swapped positions, taking the attention of training from guessing of a single word to comprehensively understanding structure and logic of the whole sequence. The swapped token detection (STD) course of D is naturally formed as a deja vu binary classification. X^{std} is created by replacing the swapped positions with generator samples:

$$x_i^{\text{std}} \sim p_{\text{SLM}}\left(x|\boldsymbol{h}_i\right), \text{ if } i \in \boldsymbol{s}\ ; \quad x_i^{\text{std}} = x_i, \text{ else.} \tag{9}$$

D is trained to distinguish whether the tokens in X^{std} is original or not via the swapped token detection (RTD) loss:

$$p_{\text{STD}}(x_i^{\text{std}} = x_i \big| \boldsymbol{h}_i) = \text{sigmoid}(\boldsymbol{w}_s^T \boldsymbol{h}_i), \tag{10}$$

$$\mathcal{L}_{\text{STD}} = \mathbb{E}\Bigg(-\sum_{x_i^{\text{std}} = x_i} \log p_{\text{STD}}\left(x_i^{\text{std}} = x_i \big| \boldsymbol{h}_i\right) - \sum_{x_i^{\text{std}} \neq x_i} \log\left(1 - p_{\text{STD}}\left(x_i^{\text{std}} = x_i \big| \boldsymbol{h}_i\right)\right) \Bigg). \tag{11}$$

[2] In the absence of special instructions, the proportion is set as 15%.

where $\boldsymbol{w}_s$ is an independent trainable parameter from $\boldsymbol{w}$ since each of courses uses its own binary classification head.

4.1.3 Inserted Token Detection

With the pace of pre-training with MLM and SLM, G is inevitably able to produce much more harmonious sequences for the consummation of semantic learning. In the meanwhile, the label distribution of corrupted sentences provided by G becomes magically imbalanced, since almost all tokens exactly match the words in the original sentence. Thus, training of D faces serious interference and lack of efficiency. The propensity of the training labels leads to the propensity of D's judgment.

To alleviate the issue of label-imbalance, and to seek another perspective of treating data, we propose the inserted token detection (ITD) course. For a given sentence $X = [x_1, x_2, ..., x_n]$, `[MASK]` is randomly inserted into the sequence at the inserted position set $\boldsymbol{i}$. The extended sentence X^{in} contains several illusory vacancies waiting for the prediction of G. Subsequently, D has to figure out which token should not be presented in the generated sentence X^{itd} with the training of the following loss:

$$p_{\text{ITD}}(x_i^{\text{itd}} = x_i^{in} \big| \boldsymbol{h}_i) = \text{sigmoid}(\boldsymbol{w}_i^T \boldsymbol{h}_i), \tag{12}$$

$$\mathcal{L}_{\text{ITD}} = \mathbb{E}\Bigg(-\sum_{x_i^{\text{itd}} = x_i^{in}} \log p_{\text{ITD}}\left(x_i^{\text{itd}} = x_i^{in} \big| \boldsymbol{h}_i\right) - \sum_{x_i^{\text{itd}} \neq x_i^{in}} \log\left(1 - p_{\text{ITD}}\left(x_i^{\text{itd}} = x_i^{in} \big| \boldsymbol{h}_i\right)\right) \Bigg). \tag{13}$$

On the one hand, the ratio of real and inserted words is fixed, solving the label-imbalance to some extent. On the other hand, training on void locations tones up the generation capability of models.

The overall structure of the proposed self-supervision courses is presented in Figure 1. All courses are jointly conducted within the same data and computing steps.

4.2 Self-correction Course

According to the above self-supervision courses, a competition mechanism between G and D seems to shape up. Facing the same piece of data, G tries to reform the sequence in many ways, while D yearns to figure out all the jugglery caused previously. However, the shared embedding layer of these two encoders becomes the only bridge of

Predict\Label	original	replaced
original	✓ pos_1	✗ pos_2
replaced	✗ pos_3	✓ pos_4

Table 1: The confusion matrix of output tokens from D. ✓ denotes that D makes a correct judgment, conversely ✗ presents the situation of wrong discrimination.

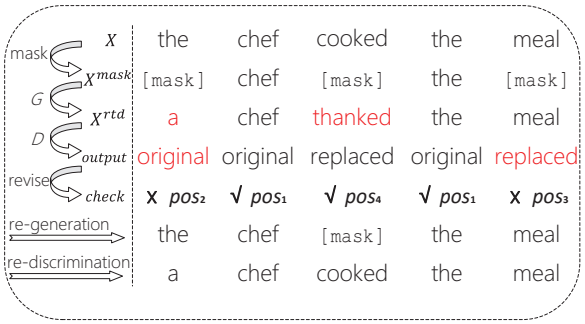

Figure 2: An example for self-correction course of RTD.

communication, which is apparently insufficient. To strengthen the link between the two components, and to provide more supervisory information on pre-training, we conduct an intimate dissection of the relationship between G and D.

Take the procedure of RTD for example. For each token x_i^{rtd} in the corrupted sentence X^{rtd}, whereafter fed into D, we identify and document its label by comparing with the token x_i at the corresponding position in X. After the discrimination process of D, this token is binary classified as original or replaced. As shown in Table 1, there exist four situations of distinguish results for x_i. pos_1: where G predicts the correct answer on the `[MASK]` position and D successfully makes a good judgment, no additional operation needs to be conducted for this kind of token. pos_2: where G fills an alternative to replace the original token and D inaccurately views it as original, it means G produces an appropriate expression to form a harmonious context as mentioned in Section 3, which makes it difficult for D to distinguish. pos_3: where D makes a clumsy mistake of incorrectly annotating an original token as replaced. pos_4: where G fills in an impertinent token at the `[MASK]` position and D easily figures it out.

To sum it up, on the one hand, G needs to regenerate tokens at pos_4, since the initial alternatives are inappropriate and unchallenging for D. As shown in Figure 2, too much `[MASK]` are placed at important locations rich in information, leading to the erratic generation "thanked". Considering that other `[MASK]` in the same sequence may interfere with the generation of tokens at pos_4, we restore other `[MASK]` to the original tokens for convenience of the re-generation process. On the other hand, D is expected to re-discriminate tokens at pos_2 and pos_3. When there exist tokens at pos_4 in a sequence, these inappropriate tokens may seriously disturb decisions of D on other tokens, leading to the consequence of pos_2 and pos_3. Take the sentence in Figure 2 for example, serious distraction "thanked" makes D falsely judges "meal" as replaced. So we replace tokens at pos_4 in X^{rtd} to original tokens to alleviate this kind of interference, and conduct the re-discrimination training on D at pos_2 and pos_3.

By sorting out and analyzing errors, an "correction notebook" for G and D is built, guiding the re-generation and re-discrimination training. Note that it's not just redoing the problem, however, we redesign the context for each kind of issue. Thus, $\mathcal{L}_{\text{re-MLM}}$ and $\mathcal{L}_{\text{re-RTD}}$ is designed as the learning objective for self-correction course of RTD. Likewise, $\mathcal{L}_{\text{re-SLM}}$ and $\mathcal{L}_{\text{re-STD}}$ presents the training loss self-correction course of STD.[3] Cause there are no original tokens at the inserted `[MASK]` positions, no revision is conducted for ITD. Two proposed self-correction courses bridge the chasm between G and D through introspection and melioration, and provide a sample-efficient secondary-supervision for the same piece of data.

Finally, G and D are co-trained with three self-supervision courses as well as two self-correction courses. The proposed MCL dissects one same sequence profoundly and comprehensively, without incurring any additional inference or memory costs.

5 Experiments

5.1 Setup

Pre-training Settings We implement the experiments on two settings: *base* and *tiny*. *Base* is the standard training configuration of $\text{BERT}_{\text{Base}}$ (Devlin et al., 2019). The model is pre-trained on English Wikipedia and BookCorpus (Zhu et al., 2015), containing 16 GB of text with 256 million samples. We set the maximum length of the input sequence

[3] The equation is not listed since the form is consistent with the previous text.

Model	GLUE Single Task								
	MNLI -m/-mm	**QQP** Acc	**QNLI** Acc	**SST-2** Acc	**CoLA** MCC	**RTE** Acc	**MRPC** Acc	**STS-B** PCC	**AVG**
Base Setting: BERT Base Size, Wikipedia + Book Corpus									
BERT (Devlin et al., 2019)	84.5/-	91.3	91.7	93.2	58.9	68.6	87.3	89.5	83.1
XLNet (Yang et al., 2019)	85.8/85.4	-	-	92.7	-	-	-	-	-
RoBERTa (Liu et al., 2019)	85.8/85.5	91.3	92.0	93.7	60.1	68.2	87.3	88.5	83.3
DeBERTa (He et al., 2021)	86.3/86.2	-	-	-	-	-	-	-	-
TUPE (Ke et al., 2021)	86.2/86.2	91.3	92.2	93.3	63.6	73.6	89.9	89.2	84.9
MC-BERT (Xu et al., 2020)	85.7/85.2	89.7	91.3	92.3	62.1	75.0	86.0	88.0	83.7
ELECTRA (Clark et al., 2020)	86.9/86.7	91.9	92.6	93.6	66.2	75.1	88.2	89.7	85.5
+HP_{Loss}+Focal (Hao et al., 2021)	87.0/86.9	91.7	92.7	92.6	66.7	81.3	90.7	91.0	86.7
CoCo-LM (Meng et al., 2021)	**88.5**/88.3	92.0	93.1	93.2	63.9	**84.8**	91.4	90.3	87.2
MCL	**88.5/88.5**	**92.2**	**93.4**	**94.1**	**70.8**	84.0	**91.6**	**91.3**	**88.3**
Tiny Setting: A quarter of training flops for ablation study, Wikipedia + Book Corpus									
ELECTRA(*reimplement*)	85.80/85.77	91.63	92.03	92.70	65.49	74.80	87.47	89.02	84.97
+STD	86.97/86.97	92.07	92.63	93.30	70.25	82.30	91.27	90.72	87.38
+ITD	87.37/87.33	91.87	92.53	93.40	68.45	81.37	90.87	90.52	87.08
Self-supervision	87.27/87.33	91.97	92.93	93.03	67.86	82.20	90.27	90.81	87.07
+ re-RTD	87.57/87.50	92.07	92.67	92.97	69.80	83.27	91.60	90.71	87.57
+ re-STD	87.80/87.77	91.97	92.93	93.33	**71.25**	82.80	91.67	**90.95**	**87.83**
MCL	**87.90/87.83**	**92.13**	**93.00**	**93.47**	68.81	**83.03**	**91.67**	90.93	87.64

Table 2: All evaluation results on GLUE datasets for comparison. Acc, MCC, PCC denote accuracy, Matthews correlation, and Spearman correlation respectively. Reported results are medians over five random seeds.

to 512, and the learning rates are 5e-4. Training lasts 125K steps with a 2048 batch size. We use the same corpus as with CoCo-LM (Meng et al., 2021) and 64K cased SentencePiece vocabulary (Kudo and Richardson, 2018). The details of the hyper-parameter of pre-training is listed in Appendix A. *Tiny* conducts the ablation experiments on the same corpora with the same configuration as the *base* setting, except that the batch size is 512.

Model Architecture The layout of our model architecture maintains the same as (Meng et al., 2021) both on *base* and *tiny* settings. D consists of 12-layer Transformer, 768 hidden size, plus T5 relative position encoding (Raffel et al., 2020). G is a shallow 4-layer Transformer with the same hidden size and position encoding. After pre-training, we discard G and use D in the same way as BERT, with a classification layer for downstream tasks.

Downstream Tasks To verify the effectiveness of the proposed methods, we conduct evaluation experiments on various downstream tasks. We evaluate on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019) and Stanford Question Answering 2.0 (SQuAD 2.0) dataset (Rajpurkar et al., 2018). As for the evaluation metrics of GLUE tasks, we adopt Spearman correlation for STS, Matthews correlation for CoLA, and accuracy for the other. For SQuAD 2.0, in which some questions are unanswerable by the passage, the standard evaluation metrics of Exact-Match (EM) and F1 scores are adopted. More details of the GLUE and SQuAD 2.0 benchmark are listed in Appendix B.

Baselines Various pre-trained models are listed and compared in the *base* setting. All numbers are from reported results in recent research. When multiple papers report different scores for the same method, we use the highest of them for comparison.

Implementation Details Our implementation builds upon the open-source implementation from fairseq (Ott et al., 2019). With 128 A100 (40 GB Memory), one pre-training run takes about 24 hours in *base* setting. The fine-tuning costs are the same with BERT plus relative positive encodings. More details of fine-tuning are listed in Appendix C.

5.2 Evaluation Results

We first pre-trained our model with the proposed MCL method, and then fine-tuned it with training sets of 8 single tasks in the GLUE benchmark. We conducted a hyperparameter search for all downstream tasks, and report the aver-

Model	SQuAD 2.0	
	EM	F1
Base Setting		
BERT (Devlin et al., 2019)	73.7	76.3
XLNet (Yang et al., 2019)	78.5	81.3
RoBERTa (Liu et al., 2019)	77.7	80.5
DeBERTa (He et al., 2021)	79.3	82.5
ELECTRA (Clark et al., 2020)	79.7	82.6
+HP_{Loss}+Focal (Hao et al., 2021)	82.7	85.4
CoCo-LM (Meng et al., 2021)	82.4	85.2
MCL	**82.9**	**85.9**
Tiny Setting for ablation study		
ELECTRA(*reimplement*)	79.37	81.31
+STD	81.73	84.55
+ITD	81.43	84.20
Self-supervision	81.87	84.85
+ re-RTD	81.70	84.48
+ re-STD	81.81	84.71
MCL	**82.04**	**84.93**

Table 3: All evaluation results on SQuAD 2.0 datasets.

age scores among 5 random seeds. Results are elaborated in the top half of Table 2. The proposed MCL evidently enhances ELECTRA and achieves at least 1.1% absolute overall improvements against state-of-art pre-trained language models on the GLUE benchmark under the *base* setting. For the most widely reported task MNLI, our model achieves 88.5/88.6 points on the matched/mismatched (m/mm) set, which obtains 1.6/1.8 absolute improvements against ELECTRA. Take a broader look at all GLUE single tasks, MCL overshadows all previous models, except RTE tasks, where CoCo-LM takes a narrow lead.

We also evaluated the proposed MCL on the SQuAD 2.0 datasets, which is an important reading comprehension dataset that requires the machine to extract the answer span given a document along with a question. The results of Exact-Match (EM) and F1 score (F1) are displayed in the top half of Table 3. Consistently, our model significantly improves the ELECTRA baseline and achieves a banner score compared with other same-size models. Specifically, under the *base* setting, the proposed MCL improves the absolute performance over ELECTRA by 3.2 points (EM) and 3.3 points (F1). Also, our model outperforms all other previous models with an overt margin.

The compelling results demonstrate the effectiveness of the proposed MCL. With the equal amount of training corpus, plus slight computing cost of forward propagation, MCL tremendously advanced ELECTRA baseline, showing its property of sample-efficiency. In other words, multi-perspective course learning gives the model a deeper and more comprehensive insight into the underlying meaning of corpora, which provides more valuable information for the pre-training process.

5.3 Ablation Study

In order to dive into the role of each component in the proposed MCL, an ablation study is conducted under the *tiny* setting. Both the GLUE and SQuAD 2.0 datasets are utilized for evaluation, and the ablation results are listed in the bottom half of Table 2 and Table 3. Bolstered by several curve graphs regarding with loss and accuracy during pre-training, every course is discussed below. [4]

RTD The most basic component, also represents the ELCETRA itself. Its performance would be employed as the baseline to compare with other additions. Not only the scores, but also the curves would be taken for important reference.

STD This course is tasked to help the model to obtain better structure perception capability through a more harmonious contextual understanding. STD improves ELECTRA on all tasks in GLUE and SQuAD 2.0 datasets. It is worth noting that the scores on CoLA task surprisingly stand out amongst the crowd. The Corpus of Linguistic Acceptability (CoLA) is used to predict whether an English sentence is linguistically acceptable or not. Apparently, pre-training on word rearrangement indeed lifts the global intellection of models, which makes it focus more on structure and logic rather than word prediction. Even the best CoLA result of 71.25 comes from the re-STD course, which further embodies the effectiveness of STD.

ITD This course is tasked to alleviate label-imbalance. As shown in Figure 5, replace rate reflects the prediction accuracy of G. Accompanied by MLM and SLM, G predicts more correct words on the `[MASK]` positions, causing the "replaced" labels to become scarce for the training of D. By adding inserted `[MASK]`, the replace rate has a fixed lower limit corresponding to the inserted proportion, leading to a balanced distribution of labels. Besides, ITD shows great improvements over

[4] Due to the constraints of space, curve graphs of pre-training and concrete instructions are replaced at Appendix D

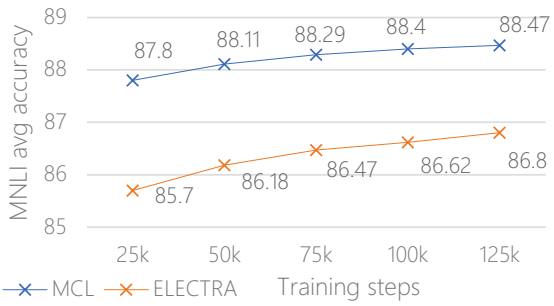

Figure 3: MNLI Comparison of pre-training efficiency.

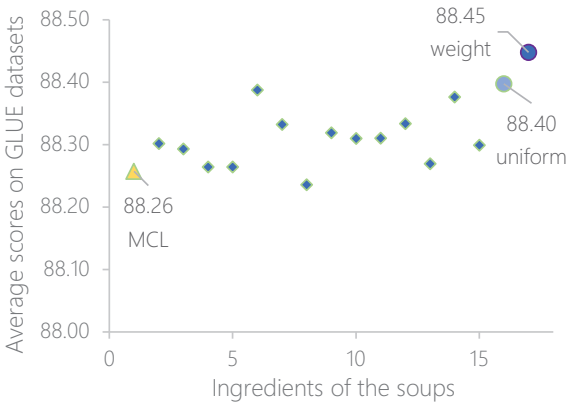

Figure 4: Average GLUE results of the course soups.

ELECTRA, especially on SST-2 datasets. The Stanford Sentiment Treebank (SST-2) provides a dataset for sentiment classification that needs to determine whether the sentiment of a sentence extracted from movie reviews is positive or negative. Predicting for the illusory `[MASK]` makes the model focus more on content comprehension, which may helpful for sentiment classification.

Self-correction Course Revision always acts as a difficult assignment, because of the challenge to reverse stereotypes. As shown in Figure 5, losses of G and D during self-correction training generally exceed that during self-supervision training, demonstrating the difficulties. However, the replace accuracy of re-RTD course goes higher than the baseline, certifying the effectiveness. Despite that self-correction training outperforms other components on all downstream tasks, the phenomena of "tug-of-war" dynamics is worth exploring. Scores listed in the last three rows of Table 2 almost touch each other, and optimal results of every single task do not always appear under the same model. It means multi-perspective courses may interfere with each other in attempts to pull parameters in different directions, which seems even more apparent under the self-correction course where secondary-samples are well designed for bootstrapping. To alleviate this situation and further improve the effectiveness of training, we found a feasible solution elaborated in Section 5.5.

5.4 Sample Efficiency Comparison

To demonstrate the proposed MCL is sample-efficient, we conduct a comparative trial between MCL and ELECTRA. As shown in Figure 3, the prevalent task MNLI is chosen for evaluation. For every 25K steps of pre-training, we reserved the model and fine-tuned it with the same configuration mentioned in Section 5.1. Obviously, MCL preponderates over ELECTRA baseline on every training node, which obtains 87.8 points at 25K steps, demonstrating its enormous learning efficiency even on small pieces of corpora.

5.5 Course Soups Trial

Inspired by model soups (Wortsman et al., 2022), which averages many models in a hyperparameter sweep during fine-tuning, we find similarities and bring this idea to MCL in a task sweep during pre-training. Different courses lead the model to lie in a different low error basin, and co-training multiple courses may create the "tug-of-war" dynamics. To solve the training conflicts, and to further improve the learning efficiency of models in the later pre-training stage, we conduct a "course soups" trial.

For ingredients in soups, we arrange all combinations of 4 losses in self-correction courses, training them into 14 single models while retaining the structure of self-supervision courses. Then all ingredients are merged through uniform and weighted integration. Results lies in Figure 4. Optimal results obtained by weight soups, which improves the average GLUE score by 0.19 absolute points against our best model MCL. The results show that the course soups suggests a effective way to guide the later training of the model by separating multiple objectives and combining them at last. More details scores are listed in Appendix E.

6 Conclusion

This paper proposes the multi-perspective course learning method, containing three self-supervision courses to improve learning efficiency and balance label distributions, as well as two self-correction courses to create a "correction notebook" for revision training. Besides, the course soups method is

designed to figure out a novel approach for efficient pre-training. Experiments show that our method significantly improves ELECTRA's performance and overshadows multiple advanced models under same settings, verifying the effectiveness of MCL.

Limitations

Although the proposed method has shown great performance to alleviate the issues of biased learning and deficient interaction, which are common problems among ELECTRA-style pre-training models, we should realize that the proposed method still can be further improved. For example, the inherent flaw of RTD mentioned in Section 3 could only be relieved rather than solved. More about mission design regarding with this issue is worth studying. Besides, although the results show great performance, more efforts are required to explore the hidden impact of each course, which will help the application of the proposed model in the future.

References

Daniel Cer, Mona 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, Vancouver, Canada. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing, IWP@IJCNLP 2005, Jeju Island, Korea, October 2005, 2005*. Asian Federation of Natural Language Processing.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL@ACL 2007 Workshop on Textual Entailment and Paraphrasing, Prague, Czech Republic, June 28-29, 2007*, pages 1–9. Association for Computational Linguistics.

Yaru Hao, Li Dong, Hangbo Bao, Ke Xu, and Furu Wei. 2021. Learning to sample replacements for ELECTRA pre-training. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 4495–4506. Association for Computational Linguistics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: decoding-enhanced bert with disentangled attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. First Quora dataset release: Question pairs.

Guolin Ke, Di He, and Tie-Yan Liu. 2021. Rethinking positional encoding in language pre-training. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*, pages 66–71. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Yu Meng, Chenyan Xiong, Payal Bajaj, Saurabh Tiwary, Paul Bennett, Jiawei Han, and Xia Song. 2021. COCO-LM: correcting and contrasting text sequences for language model pretraining. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 23102–23114.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Demonstrations*, pages 48–53. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 784–789. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1631–1642. ACL.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *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.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*, pages 1096–1103. ACM.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. 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.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Trans. Assoc. Comput. Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR.

Zhenhui Xu, Linyuan Gong, Guolin Ke, Di He, Shuxin Zheng, Liwei Wang, Jiang Bian, and Tie-Yan Liu. 2020. MC-BERT: efficient language pre-training via a meta controller. *CoRR*, abs/2006.05744.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 19–27. IEEE Computer Society.

A Hyperparameters for Pre-training

As shown in Table 4, we present the hyperparameters used for pre-training MCL on the *base* setting. We follow the optimization hyperparameters of CoCo-LM (Meng et al., 2021) for comparisons. Note that all losses conducted on D are multiplied by λ (set as 50), which is a hyperparameter used to balance the training pace of G and D.

Layers	12
Hidden size	768
FFN inner hidden size	3072
Attention heads	12
Attention head size	64
Max relative position	128
Training steps	125K
Batch size	2048
Adam ϵ (Kingma and Ba, 2015)	1e-6
Adam β	(0.9, 0.98)
Learning rate	5e-4
Learning rate schedule	Linear
Warmup steps	10K
Gradient clipping	2.0
Dropout (Hinton et al., 2012)	0.1
Weight decay	0.01

Table 4: Hyperparameters for pre-training.

B Details of Downstream Tasks

GLUE contains a wide range of tasks covering textual entailment: RTE (Giampiccolo et al., 2007) and MNLI (Williams et al., 2018), question-answer entailment: QNLI (Rajpurkar et al., 2016), paraphrase: MRPC (Dolan and Brockett, 2005), question paraphrase: QQP (Iyer et al., 2017), textual similarity: STS (Cer et al., 2017), sentiment: SST (Socher et al., 2013), and linguistic acceptability CoLA (Warstadt et al., 2019).

Natural Language Inference involves reading a pair of sentences and judging the relationship between their meanings, such as entailment, neutral and contradiction. We evaluate on three diverse datasets, including Multi-Genre Natural Language Inference (MNLI), Question Natural Language Inference (QNLI) and Recognizing Textual Entailment (RTE).

Semantic similarity tasks aim to predict whether two sentences are semantically equivalent or not. The challenge lies in recognizing rephrasing of concepts, understanding negation, and handling syntactic ambiguity. Three datasets are used, including Microsoft Paraphrase corpus (MRPC), Quora Question Pairs (QQP) dataset and Semantic Textual Similarity benchmark (STS-B).

Dataset	**#Train/#Dev/#Test**
Single-Sentence Classification	
CoLA (Acceptability)	8.5k/1k/1k
SST-2 (Sentiment)	67k/872/1.8k
Pairwise Text Classification	
MNLI (NLI)	393k/20k/20k
RTE (NLI)	2.5k/276/3k
QNLI (NLI)	105k/5.5k/5.5k
WNLI (NLI)	634/71/146
QQP (Paraphrase)	364k/40k/391k
MRPC (Paraphrase)	3.7k/408/1.7k
Text Similarity	
STS-B (Similarity)	7k/1.5k/1.4k

Table 5: Summary of the GLUE benchmark.

Classification The Corpus of Linguistic Acceptability (CoLA) is used to predict whether an English sentence is linguistically acceptable or not. The Stanford Sentiment Treebank (SST-2) provides a dataset for sentiment classification that needs to determine whether the sentiment of a sentence extracted from movie reviews is positive or negative.

As a widely used MRC benchmark dataset, SQuAD 2.0 is a reading comprehension dataset that requires the machine to extract the answer span given a document along with a question. We select the v2.0 version to keep the focus on the performance of pure span extraction performance. Two official metrics are used to evaluate the model performance: Exact Match (EM) and a softer metric F1 score, which measures the average overlap between the prediction and ground truth answer at the token level.

The summary of the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019) is shown in Table 5.

C Hyperparameters for Fine-tuning

Table 6 presents the hyperparameters used for fine-tuning over SQuAD v2.0 (Rajpurkar et al., 2018), and the GLUE benchmark (Wang et al., 2019) following CoCo-LM for fair comparison. On the development sets, the hyperparameters are searched based on the average performance of five runs.

Parameters	GLUE Small Tasks	GLUE Large Tasks	SQuAD 2.0
Max Epochs	{2, 3, 5, 10}	{2, 3, 5}	{2, 3}
Peak Learning Rate	{2e-5, 3e-5, 4e-5, 5e-5}	{1e-5, 2e-5, 3e-5, 4e-5}	{2e-5, 3e-5, 4e-5, 5e-5}
Batch Size	{16, 32}	32	{16, 32}
Learning Rate Decay	Linear	Linear	Linear
Warm-Up Proportion	{6%, 10%}	6%	{6%, 10%}
Sequence Length	512	512	512
Adam ϵ	1e-6	1e-6	1e-6
Adam (β_1, β_2)	(0.9, 0.98)	(0.9, 0.98)	(0.9, 0.98)
Clip Norm	-	-	-
Dropout	0.1	0.1	0.1
Weight Decay	0.01	0.01	0.01

Table 6: Hyperparameter ranges searched for fine-tuning on GLUE and SQuAD 2.0. GLUE small tasks include CoLA, RTE, MRPC and STS-B. GLUE large tasks include MNLI, QQP, QNLI and SST-2.

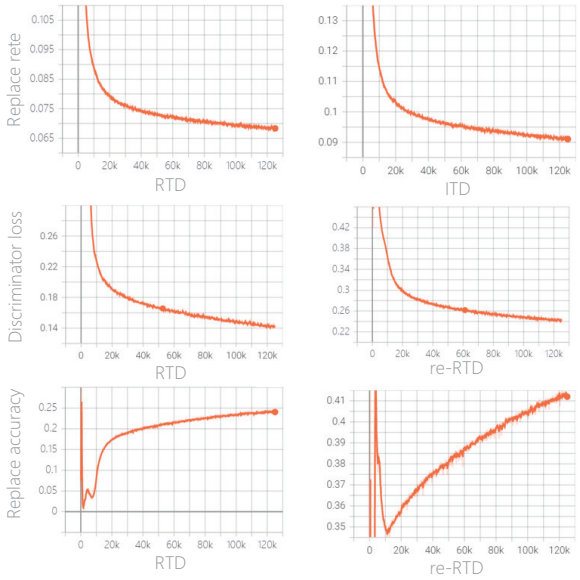

Figure 5: Curves of replace rate, pre-training loss and replace accuracy for ablation study. All figures are drawn by Tensorboard.

D Curves for Ablation Study

As shown in Figure 5, three metrics are selected to evaluate the quality of pre-training.

Replace Rate This metric represents the ratio of replaced tokens in the corrupted sentence X^{rtd}. The less of this ratio, the better of G's pre-training, and the more uneven of the label distributions for D. From the first row in the diagram, we can see that the lower bound of the replace rate with ITD apparently exceeds that with RTD, demonstrating that ITD indeed alleviates the issue of label-imbalance.

Loss and Replace Accuracy Training loss reflects the pre-training assessment. One of the self-correction courses, re-RTD, holds a higher loss against that with RTD, showing the difficulty of the revision training. Replace accuracy denotes the prediction accuracy on replaced tokens. As shown in the third line of Figure 5, re-RTD achieves better replace accuracy by a significant margin compared with RTD, demonstrating the effectiveness of the self-correction course.

E Detailed Scores of Course Soups

Table 7 lists all scores on the GLUE benchmark of the course soups trial. It seems that optimal results on single tasks are randomly distributed in the ingredients of soups. Through uniform and weighted integration, the best model emerges with the uppermost average GLUE scores.

Model	GLUE Single Task								
	MNLI -m/-mm	**QQP** Acc	**QNLI** Acc	**SST-2** Acc	**CoLA** MCC	**RTE** Acc	**MRPC** Acc	**STS-B** PCC	**AVG**
Base Setting: BERT Base Size, Wikipedia + Book Corpus									
MCL	88.47/88.47	92.23	93.37	94.13	70.76	84.00	91.57	91.32	88.26
$\mathcal{L}_{\text{re-MLM}}$	**88.53**/88.50	92.23	93.40	94.33	70.53	84.00	92.00	91.18	88.30
$\mathcal{L}_{\text{re-RTD}}$	88.47/88.43	92.23	93.37	94.13	70.77	83.63	**92.40**	91.20	88.29
$\mathcal{L}_{\text{re-SLM}}$	88.43/88.43	92.23	93.43	94.27	70.53	83.77	92.00	91.29	88.26
$\mathcal{L}_{\text{re-STD}}$	88.43/88.43	92.23	93.43	94.27	70.53	83.77	92.00	91.29	88.26
$\mathcal{L}_{\text{re-MLM+RTD}}$	88.43/88.33	92.20	93.43	94.27	70.88	83.77	92.17	91.30	88.31
$\mathcal{L}_{\text{re-MLM+SLM}}$	88.50/88.43	92.17	93.47	94.07	71.12	83.40	**92.40**	91.24	88.31
$\mathcal{L}_{\text{re-MLM+STD}}$	88.43/88.43	92.23	93.37	94.27	71.09	83.77	92.17	91.24	88.33
$\mathcal{L}_{\text{re-RTD+SLM}}$	88.47/88.43	**92.27**	93.43	94.23	70.84	83.27	92.23	91.25	88.27
$\mathcal{L}_{\text{re-RTD+STD}}$	88.50/88.50	92.23	93.40	94.30	71.00	84.03	92.17	91.25	88.38
$\mathcal{L}_{\text{re-SLM+STD}}$	88.50/88.47	**92.27**	93.57	94.13	70.53	83.63	92.23	**91.36**	88.30
$\mathcal{L}_{\text{re-SLM+RTD+STD}}$	88.47/88.43	**92.27**	93.57	94.07	**71.61**	83.77	92.10	91.21	88.39
$\mathcal{L}_{\text{re-MLM+SLM+STD}}$	88.47/**88.53**	**92.27**	93.40	94.40	70.79	84.00	91.90	91.24	88.33
$\mathcal{L}_{\text{re-MLM+RTD+STD}}$	88.50/88.47	92.23	93.43	94.07	70.80	83.53	91.93	91.15	88.24
$\mathcal{L}_{\text{re-MLM+RTD+SLM}}$	88.43/88.40	92.23	**93.60**	94.30	70.75	**84.13**	91.77	91.25	88.32
uniform soups	**88.53**/88.43	92.23	93.43	**94.53**	71.40	83.53	92.23	91.24	88.40
weight soups	88.47/88.43	92.20	93.57	94.13	**71.61**	84.00	**92.40**	91.22	**88.45**

Table 7: All evaluation results on GLUE datasets for the course soups trial. Acc, MCC, PCC denote accuracy, Matthews correlation, and Spearman correlation respectively. Reported results are medians over five random seeds.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Left blank.

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Left blank.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☒ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
No response.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

Left blank.

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Left blank.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Left blank.

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Left blank.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Left blank.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.