COCO-LM: Correcting and Contrasting Text Sequences for Language Model Pretraining

Yu Meng [1] **Chenyan Xiong** [2] **Payal Bajaj** [2] **Saurabh Tiwary** [2] **Paul Bennett** [2] **Jiawei Han** [1] **Xia Song** [2]

Abstract

We present COCO-LM, a new self-supervised learning framework that pretrains Language Models by COrrecting challenging errors and COntrasting text sequences. COCO-LM employs an auxiliary language model to mask-and-predict tokens in original text sequences. It creates more challenging pretraining inputs, where noises are sampled based on their likelihood in the auxiliary language model. COCO-LM then pretrains with two tasks: The first task, corrective language modeling, learns to correct the auxiliary model's corruptions by recovering the original tokens. The second task, sequence contrastive learning, ensures that the language model generates sequence representations that are invariant to noises and transformations. In our experiments on the GLUE and SQuAD benchmarks, COCO-LM outperforms recent pretraining approaches in various pretraining settings and few-shot evaluations, with higher pretraining efficiency. Our analyses reveal that COCO-LM's advantages come from its challenging training signals, more contextualized token representations, and regularized sequence representations.

1. Introduction

Pretraining language models (PLMs) (Devlin et al., 2019; Radford et al., 2019; Raffel et al., 2019) have revolutionized the way AI systems process natural languages. By pretraining on large text corpora (Raffel et al., 2019; Brown et al., 2020) and scaling Transformers (Vaswani et al., 2017) to millions and billions of parameters (Devlin et al., 2019; Raffel et al., 2019), the state-of-the-art in many language related tasks has been refreshed at a historic speed in the past several years.

On the other hand, within the standard language model pretraining framework, it is observed that the empirical performance of PLMs on downstream tasks only improves linearly with the exponential growth of parameter size and pretraining cost (Kaplan et al., 2020). This is unsustainable as PLMs have reached trillions of parameters (Brown et al., 2020; Fedus et al., 2021).

Recent research has revealed some intrinsic limitations of existing pretraining frameworks that may result in this sublinear efficiency. One challenge is that pretraining with randomly altered texts (*e.g.*, randomly masked tokens) yields many non-informative signals no longer useful after a certain amount of pretraining (Roberts et al., 2020; Guu et al., 2020; Ye et al., 2020). Another one is that pretraining at token level does not explicitly learn language semantics at the sequence level and Transformers may not generalize to higher level semantics efficiently during pretraining (Li et al., 2020; Thakur et al., 2020).

In this paper, we aim to overcome these limitations with a new self-supervised learning framework, COCO-LM, that pretrains Language Models by COrrecting and COntrasting text sequences with more challenging noises. To construct more informative pretraining signal, COCO-LM leverages an auxiliary language model, similar to the generator in ELECTRA (Clark et al., 2020b), to corrupt text sequences by sampling more contextually plausible noises from its masked language modeling (MLM) probability. Different from the replaced token detection task in ELECTRA, COCO-LM revives a language modeling task, corrective language modeling (CLM), which pretrains the Transformer to not only detect the challenging noises in the corrupted texts, but also correct them via a multi-task setting.

To improve the learning of sequence level semantics, COCO-LM introduces a sequence level pretraining task, sequence contrastive learning (SCL), that uses contrastive learning to enforce the pretraining model to align the corrupted text sequence and its cropped original sequence close in the representation space, while away from other random sequences. This encourages the model to leverage more information from the entire sequence to produce sequence representations that are invariant to token-level alterations.

[1]University of Illinois at Urbana-Champaign {yumeng5, hanj}@illinois.edu [2]Microsoft {Chenyan.Xiong, Payal.Bajaj, satiwary, Paul.N.Bennett, xiaso}@microsoft.com.

COCO-LM significantly improves the generalization ability of language models on a variety of downstream tasks in GLUE (Wang et al., 2018) and SQuAD (Rajpurkar et al., 2016) benchmarks. It outperforms recent approaches (Clark et al., 2020b; He et al., 2020; Ke et al., 2020) by large margins (*e.g.*, about 1.6 and 1.42 points accuracy gain on MNLI and EM gain SQuAD 2.0 respectively with base model training). It is also more cost-effective and has better few-shot ability in downstream tasks. Our thorough analyses also reveal that the benefits of COCO-LM come from its challenging pretraining signals, more contextualized token representations, and regularized sequence representations.[1]

2. Related Work

Designing better pretraining tasks than standard language modeling (Bengio et al., 2003; Devlin et al., 2019) is an important research topic in language representation learning (Radford et al., 2019; Song et al., 2019). For example, XLNet proposes permutation language modeling that conducts MLM in an autoregressive manner (Yang et al., 2019); UniLM uses pseudo MLM to unify autoregressive and MLM tasks for both language representation and generation (Dong et al., 2019; Bao et al., 2020). Lewis et al. (2019) conduct a thorough study of these variants and show MLM is still among the most effective in many applications.

One way to make MLM more informative is to mask more informative positions/spans (Joshi et al., 2019; Song et al., 2019; Guu et al., 2020) or to automatically learn masking positions (Ye et al., 2020). By masking more informative tokens, the pretrained language models focus more on the semantics required to recover those tokens (*e.g.*, entities and attributes). This significantly boostes the generalization ability of the pretrained models in semantic-centric tasks, including more accurate question answering and more factually correct language generation (Guu et al., 2020; Roberts et al., 2020; Rosset et al., 2020).

Instead of optimizing mask positions, ELECTRA (Clark et al., 2020b) employs an auxiliary network to corrupt the input sequence with more challenging noises. It uses the MLM trained auxiliary model to replace tokens with samples from its MLM probability, and pretrains the main Transformer to detect the replaced tokens via binary classification. The two networks are pretrained jointly: The auxiliary model generates more and more challenging noises for the main Transformer to detect; the main Transformer so trained achieves strong performance in downstream tasks.

Despite its empirical advantage, there are concerns about whether ELECTRA's binary classification task misses some properties of language modeling. The ELECTRA authors explored a standard language model task on the corrupted

text sequence (All-Token LM), but observed performance degradation (Clark et al., 2020b). ELECTRIC (Clark et al., 2020a) proposes a language model task that contrasts the original tokens from noises sampled from a cloze model. Although it underperforms ELECTRA on GLUE, ELECTRIC still pretrains a language model which can be used in tasks like scoring the feasibility of a generated text sequence. MC-BERT uses a multiple-choice task which selects original tokens from plausible alternatives and performs on par with ELECTRA (Xu et al., 2020). COCO-LM also leverages an auxiliary network to generate pretraining inputs but uses two new pretraining tasks, one of which is a language modeling task.

Another frontier in pretraining research is to incorporate sentence level signals, for example, next sentence prediction (Devlin et al., 2019), sentence ordering (Lan et al., 2019), and previous sentence prediction (Wang et al., 2019). However, RoBERTa found the next sentence prediction task not beneficial and only uses token level MLM task (Liu et al., 2019). The benefits of sentence level pretraining task are usually observed on some specific tasks (Chi et al., 2020; Lewis et al., 2020) such as modeling long-form texts (Ravula et al., 2020) and grounded question answering (Guu et al., 2020).

Recent successes of contrastive learning with language are mainly achieved in the fine-tuning stage. Gunel et al. (2020) conducts supervised contrasting learning on GLUE and improves few-shot accuracy in fine-tuning. Xiong et al. (2020) uses contrastive learning in dense text retrieval, using relevant query-document labels to construct contrast pairs. CERT (Fang & Xie, 2020) conducts continuous training from BERT using contrastive pairs generated from back-translation (Pham et al., 2020) but underperforms RoBERTa.

3. Method

In this section, we first recap ELECTRA-Style language model pretraining and then present COCO-LM.

3.1. Preliminary

In the *masked language modeling* (MLM) (Devlin et al., 2019) task, the pretraining model, often a Transformer (Vaswani et al., 2017), takes an input sequence $X^{\text{orig}} = [x_1, \ldots, x_i, \ldots, x_n]$ with some tokens randomly replaced by [MASK] symbols and learns to predict the original tokens:

$$[x_1, \ldots, \text{[MASK]}, \ldots, x_n] \xrightarrow{\text{Transformer}} \boldsymbol{H}^{\text{mlm}} \xrightarrow{\text{LM Head}} X^{\text{mlm}},$$

where $\boldsymbol{H}^{\text{mlm}} = [\boldsymbol{h}_1^{\text{mlm}}, \ldots, \boldsymbol{h}_i^{\text{mlm}}, \ldots, \boldsymbol{h}_n^{\text{mlm}}]$ is the contextualized representation of the input sequence, $X^{\text{mlm}} = [x_1^{\text{mlm}}, \ldots, x_i^{\text{mlm}}, \ldots, x_n^{\text{mlm}}]$ is the sequence with masked positions filled with MLM predicted tokens, and LM Head is

---

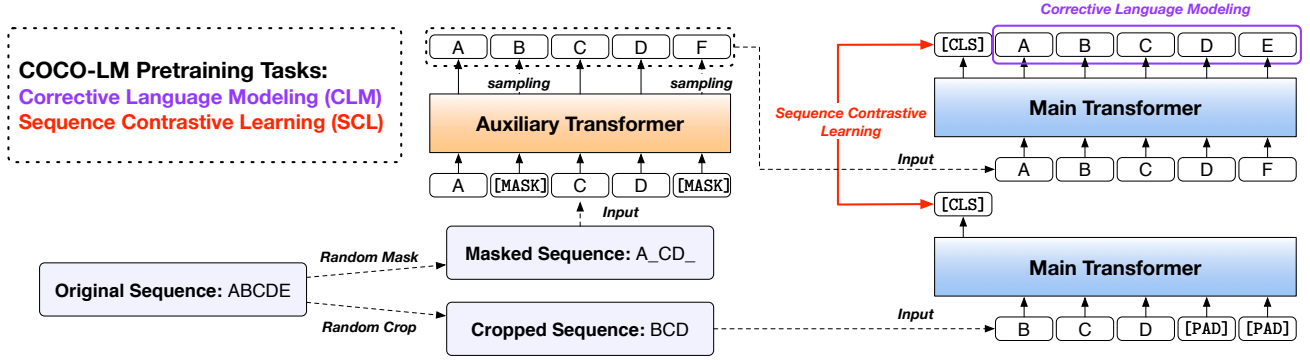[1]We plan to open-source our code and pretrained models.

Figure 1. The overview of COCO-LM. The auxiliary Transformer is pretrained by MLM. We sample output tokens from its LM probability to construct a corrupted sequence, which is used as the pretraining input of the main Transformer for Corrective Language Modeling. The corrupted sequence also forms a positive sequence pair with the cropped original sequence in Sequence Contrastive Learning.

a classification layer that learns to predict the original token from the vocabulary V with the following probability:

$$p_{\mathrm{LM}}(x_i|\boldsymbol{h}_i) = \frac{\exp(\boldsymbol{x}_i^\top \boldsymbol{h}_i)}{\sum_{x_j \in V} \exp(\boldsymbol{x}_j^\top \boldsymbol{h}_i)}.$$

The token embeddings $\boldsymbol{x}_j$ are parameters shared in Transformer input and the LM Head output layer. The Transformer is trained via the cross entropy loss between X^{mlm} and X^{orig} on the masked positions (Devlin et al., 2019).

The randomly chosen masks do not always provide the best pretraining signals: Many masked tokens are trivial common words that may not push the Transformer to capture meaningful language semantics (Guu et al., 2020), while some might be too hard or have many false negatives (Xu et al., 2020). Pretraining on those masks is not guaranteed to elevate the language model's generalization ability.

Clark et al. (2020b) developed a new framework, *ELECTRA*, which instead of working on the masked sequences directly, first leverages an auxiliary MLM model ("generator") to infer a sequence X^{mlm} as the pretraining input for the main network ("discriminator"). The latter learns to detect which tokens are replaced via a binary classification task called "replaced token detection":

$$X^{\mathrm{orig}} \xrightarrow{\text{Aux. Transformer}} X^{\mathrm{mlm}} \xrightarrow{\text{Main Transformer}} \boldsymbol{H}^{\mathrm{disc}} \xrightarrow{\text{Sigmoid}} Y.$$

The main Transformer detects whether the input token is kept or replaced: $y_i = 1$ iff $x_i = x_i^{\mathrm{mlm}}$, using the sigmoid binary classification head. The two networks are pretrained side-by-side: The auxiliary Transformer is trained by MLM and outputs more plausible and challenging token replacements $x_i^{\mathrm{mlm}} \neq x_i$; the main network learns to better detect the deceiving replacements from X^{mlm}.

The auxiliary network is solely used to construct pretraining signals and is discarded after pretraining. The main Transformer is fine-tuned for downstream tasks and is quite

effective in a wide range of tasks (Clark et al., 2020b). The source of its empirical advantage, however, is somehow a mystery (Clark et al., 2020a). After all, the main Transformer is not even trained as a language model; it merely learns from the binary classification task, yet performs well on tasks where modeling sophisticated language semantics is required (Wang et al., 2018).

Clark et al. (2020b) explored an All-Token MLM task which trains the main Transformer to predict the original token besides detecting replacements, but it decreased the generalization ability of ELECTRA. Later, Clark et al. (2020a) proposed ELECTRIC that uses language modeling probabilities to distinguish the original tokens from noises sampled from a cloze model instead of the auxiliary language model. It does not outperform ELECTRA but maintains the language modeling capability, which is necessary in some applications (Clark et al., 2020a).

3.2. Pretraining by Correcting and Contrasting

COCO-LM also employs an auxiliary Transformer to corrupt text sequences with more challenging noises—we later show this is critical for the pretrained model's generalization ability (Sec. 5.2). Different from ELECTRA (Clark et al., 2020b), COCO-LM first revives the language modeling task on the corrupted text sequences. Then it introduces a new sequence level task with contrastive learning (Sec. 3.2.2). The framework of COCO-LM is illustrated in Figure 1.

3.2.1. CORRECTIVE LANGUAGE MODELING

Corrective Language Modeling (CLM) is a token level pretraining task: Given a text sequence X^{mlm} corrupted by the auxiliary network, CLM aims to recover the original tokens:

$$X^{\mathrm{orig}} \xrightarrow{\text{Aux. Transformer}} X^{\mathrm{mlm}} \xrightarrow{\text{CLM}} X^{\mathrm{orig}}.$$

The noises in X^{mlm} are considered plausible in the context by the auxiliary network thus are more challenging.

The main Transformer performs the CLM task as:

$$X^{\text{mlm}} \xrightarrow{\text{Main Transformer}} \boldsymbol{H}^{\text{clm}} \xrightarrow{\text{CLM Head}} X^{\text{orig}};$$

$$\text{CLM Head}(\boldsymbol{h}_i) = \arg\max_x \left(p_{\text{CLM}}(x_i|\boldsymbol{h}_i) \right).$$

Here $\boldsymbol{H}^{\text{clm}}$ are the representations from the main Transformer. The CLM head is similar to the one used in All-Token MLM (Clark et al., 2020b): A standard language modeling softmax plus a copy mechanism:

$$p_{\text{CLM}}(x_i|\boldsymbol{h}_i) = \mathbb{1}(x_i = x_i^{\text{mlm}}) p_{\text{copy}}(y_i = 1|\boldsymbol{h}_i)$$
$$+ p_{\text{copy}}(y_i = 0|\boldsymbol{h}_i) \frac{\exp(\boldsymbol{x}_i^\top \boldsymbol{h}_i)}{\sum_j \exp(\boldsymbol{x}_j^\top \boldsymbol{h}_i)},$$

$$p_{\text{copy}}(y_i|\boldsymbol{h}_i) = \frac{\exp(y_i \cdot \boldsymbol{w}_{\text{copy}}^\top \boldsymbol{h}_i)}{\exp(\boldsymbol{w}_{\text{copy}}^\top \boldsymbol{h}_i) + 1},$$

where $\mathbb{1}(\cdot)$ is the indicator function; $\boldsymbol{w}_{\text{copy}}$ is a learnable weight. The copy mechanism adds the probability $p_{\text{copy}}(y_i = 1|\boldsymbol{h}_i)$ to copy the input word using the binary classification layer $p_{\text{copy}}(\cdot)$.

As shown in (Clark et al., 2020b), pretraining $p_{\text{CLM}}(\cdot)$ only using a language modeling loss (All-Token MLM) performs worse than using the simple binary replaced token detection. We find that it is mainly due to the All-Token MLM's ineffectiveness in handling the noises from the auxiliary language model—To recover the original token is much harder than just to detect the replacement.

CLM improves the learning of token recovery using two techniques: A standard multi-task setting that explicitly learns the copy mechanism $p_{\text{copy}}(\cdot)$ using binary labels, and a stop gradient (sg) layer to avoid the disturbance from the hard language modeling task to the copy mechanism:

$$\mathcal{L}_{\text{copy}} = -\sum_i \log p_{\text{copy}}(y_i^*|\boldsymbol{h}_i), \qquad (1)$$

$$\mathcal{L}_{\text{LM}} = -\sum_i \log p_{\text{CLM}}(x_i|\boldsymbol{h}_i)$$
$$= -\sum_i \log \left(\mathbb{1}(x_i = x_i^{\text{mlm}}) p_{\text{copy}}^{\text{sg}}(y_i = 1|\boldsymbol{h}_i) \right.$$
$$\left. + p_{\text{copy}}^{\text{sg}}(y_i = 0|\boldsymbol{h}_i) \frac{\exp(\boldsymbol{x}_i^\top \boldsymbol{h}_i)}{\sum_j \exp(\boldsymbol{x}_j^\top \boldsymbol{h}_i)} \right), \qquad (2)$$

$$\mathcal{L}_{\text{CLM}} = \lambda_{\text{copy}} \mathcal{L}_{\text{copy}} + \mathcal{L}_{\text{LM}},$$

where y_i^* is the ground truth of the copy mechanism. The sg in Eqn. (2) denote that gradients from $\mathcal{L}_{\text{LM}}$ do not update $p_{\text{copy}}(\cdot)$; λ_{copy} is a hyperparameter balancing the two tasks. The binary cross entropy loss in Eqn. (1) is dedicated to learning the copying probability. We prevent the learning of $p_{\text{copy}}(\cdot)$ from being disturbed by the harder LM task $\mathcal{L}_{\text{LM}}$. This way, the main Transformer first learns the easier binary classification task, and uses the learned copy mechanism to improve the learning of the harder task.

3.2.2. SEQUENCE CONTRASTIVE LEARNING

Besides token level pretraining, COCO-LM introduces a sequence contrastive learning task (SCL), which pretrains the model to provide better sequence level representations.

Specifically, in SCL, each original sequence is transformed separately via MLM replacement (X^{mlm}) and random cropping (X^{crop}). A training batch $B = \{X_1^{\text{mlm}}, \ldots, X_N^{\text{mlm}}, X_1^{\text{crop}}, \ldots, X_N^{\text{crop}}\}$ contains both MLM replaced and cropped sequences (the crop operation keeps a random 90% contiguous span of the original sequences to maintain major meanings). We use the following contrastive learning loss to align the sequence representations of the positive pairs $(X_j^{\text{mlm}}, X_j^{\text{crop}})$ and $(X_j^{\text{crop}}, X_j^{\text{mlm}})$, in contrast to random pairs as negatives.

$$X^{\text{orig}} \xrightarrow{\text{Crop}} X^{\text{crop}},$$

$$\mathcal{L}_{\text{SCL}} = - \sum_{\text{pos. pair } (X_j, X_k) \in B} \log \frac{\exp(\boldsymbol{s}_j^\top \boldsymbol{s}_k)}{\sum_{X_l \in B \setminus \{X_j\}} (\boldsymbol{s}_j^\top \boldsymbol{s}_l)},$$

where $\boldsymbol{s}_j = \boldsymbol{h}_{j:[\texttt{CLS}]} / \|\boldsymbol{h}_{j:[\texttt{CLS}]}\|$ is the $l2$-normalized [CLS] sequence representation. This contrastive learning task requires the sequence embeddings of X_j^{mlm} and X_j^{crop} to be close with each other while away from other random sequences from the same batch B. This encourages the main network to produce representations invariant to minor token-level alterations (Purushwalkam & Gupta, 2020).

As the first step to leverage contrastive learning in sequence level pretraining, we keep everything straightforward: A simple cropping as data augmentation and the default temperature 1 is used in the softmax. Advanced data transformations (Qu et al., 2020) and hyperparameter explorations (Oord et al., 2018; Chen et al., 2020) may further improve COCO-LM but are reserved for future work.

3.2.3. COCO-LM TRAINING

Putting the two tasks together, the pretraining framework of COCO-LM can be summarized as:

$$X^{\text{orig}} \xrightarrow{\text{MLM}} X^{\text{mlm}}, \qquad X^{\text{orig}} \xrightarrow{\text{Crop}} X^{\text{crop}}; \qquad (3)$$

$$X^{\text{mlm}} \xrightarrow{\text{CLM}} X^{\text{orig}}, \qquad X^{\text{mlm}} \xleftrightarrow{\text{SCL}} X^{\text{crop}}. \qquad (4)$$

In Eqn. (3) we construct the pretraining signals for COCO-LM: The auxiliary network is pretrained by standard MLM to provide corrupted training sequences X^{mlm}; the original sequence is cropped to form a simple augmentation X^{crop}. In Eqn. (4) we leverage these signals to pretrain the main network, by correcting the replaced tokens at the token level (CLM) and by contrasting the representations of the replaced and cropped texts at sequence level (SCL). The auxiliary network and the main network are pretrained side-by-side in COCO-LM's self-supervised learning framework.

Model	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	RTE	MRPC	STS-B	AVG	EM	F1
				GLUE DEV						**SQuAD 2.0 DEV**	
Base Models Wikipedia + BookCorpus											
BERT (Devlin et al., 2019)	84.50/-	91.30	91.70	93.20	58.90	68.60	87.30	**89.50**	83.13	73.70	76.30
RoBERTa (Liu et al., 2019)	84.70/-	–	–	92.70	–	–	–	–	–	–	79.70
XLNet (Yang et al., 2019)	85.80/-	–	–	92.70	–	–	–	–	–	–	81.33
ALBERT (Lan et al., 2019)	83.50/-	–	–	91.70	–	–	–	–	–	79.40	82.30
ELECTRA (Clark et al., 2020b)	86.00/85.29	89.96	91.85	93.38	64.33	70.80	84.88	89.10	83.74	80.50	83.30
MC-BERT (Xu et al., 2020)	85.68/85.24	89.65	91.34	92.34	62.10	74.96	85.96	88.01	83.73	–	–
BERT+Rel-Pos (Ke et al., 2020)	85.81/85.84	91.12	92.16	92.90	55.43	71.46	89.26	88.94	83.39	–	–
UniLM V2 (Bao et al., 2020)	86.10/86.10	–	–	93.20	–	–	–	–	–	80.90	83.60
DeBERTa (He et al., 2020)	86.30/86.20	–	–	–	–	–	–	–	–	79.30	82.50
TUPE (Ke et al., 2020)	86.21/86.19	91.30	92.17	93.26	63.56	73.56	89.89	89.23	84.90	–	–
RoBERTa (Ours)	85.61/85.51	91.34	91.80	**93.86**	58.64	69.03	87.50	86.53	83.03	77.71	80.51
ELECTRA (Ours)	86.92/86.72	91.86	92.56	93.64	**66.50**	75.28	88.46	88.04	85.39	79.74	82.58
COCO-LM Base	**88.52/88.30**	**92.04**	**93.13**	93.30	64.01	**85.42**	**91.51**	88.61	**87.05**	**82.32**	**85.12**
Base++ Models Bigger Training Data and/or More Training Steps											
XLNet (Yang et al., 2019)	86.80/-	91.40	91.70	94.70	60.20	74.00	88.20	89.50	84.56	80.20	–
RoBERTa (Liu et al., 2019)	87.60/-	91.90	92.80	94.80	63.60	78.70	90.20	**91.20**	86.35	80.50	83.70
UniLM V2 (Bao et al., 2020)	88.50/-	91.70	93.50	**95.10**	65.20	81.30	**91.80**	91.00	87.09	83.30	86.10
DeBERTa (He et al., 2020)	88.80/88.50	–	–	–	–	–	–	–	–	83.10	86.20
CLEAR (Wu et al., 2020)	86.70/-	90.00	92.90	94.50	64.30	78.30	89.20	89.80	85.71	–	–
COCO-LM Base++	**89.66/89.60**	**92.14**	**93.64**	94.32	**68.31**	84.03	90.63	89.54	**87.78**	**83.53**	**86.61**

Table 1. Single model results on GLUE and SQuAD 2.0 development set. All ours runs are the five-run medians on GLUE and averages on SQuAD 2.0. Results not available in public reports are marked as "–". Our evaluation metrics are Spearman correlation for STS, Matthews correlation for CoLA, and accuracy for the other GLUE tasks. AVG is the average of the eight tasks on GLUE.

4. Experimental Setup

This section describes our experiment setups.

Pretraining Setting: We employ two standard pretraining settings, *base* and *base++*. *Base* is the standard BERT base training configuration (Devlin et al., 2019): Pretraining on Wikipedia[2] and BookCorpus (Zhu et al., 2015) (16 GB of texts) for 256 million samples on 512 token sequences (or 125K batches with 2048 batch size).

Base++ is to train the model with the same configuration but larger corpora and/or more training steps. We follow the settings in XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019), and UniLM V2 (Bao et al., 2020), which add in OpenWebText1[3], CC-News (Liu et al., 2019) and STORIES (Trinh & Le, 2018), to the total of 160 GB texts. We train for 4.0 billion (with 2048 batch size) samples, the same with Liu et al. (2019).

There are inevitable variations in the pretraining corpora used in different work. Our *base* corpus is obtained from the authors of MC-BERT (Xu et al., 2020) and TUPE (Ke et al., 2020). Our *base++* corpus is the most similar with those used in UniLM (Dong et al., 2019; Bao et al., 2020).

Downstream Tasks: We use the tasks included the GLUE benchmark (Wang et al., 2018) and SQuAD 2.0 reading compression (Rajpurkar et al., 2016). The fine-tuning protocols are based on the open-source implementation released by Ke et al. (2020) on GLUE tasks and by huggingface (Wolf

---

[2] https://dumps.wikimedia.org/enwiki/
[3] https://skylion007.github.io/OpenWebTextCorpus/

et al., 2019) on SQuAD. All pretrained models are evaluated with the same fine-tuning protocols and the reported results are the median/average of five/ten random seeds in GLUE/SQuAD. Please refer to Appendix for more details.

Model Architecture: Our main network uses the RoBERTa base architecture (Liu et al., 2019): 12 layer Transformer, 768 hidden size, and BPE tokenization with 32,768 vocabulary size (Sennrich et al., 2015), plus T5 relative position encoding (Raffel et al., 2019). Our auxiliary network is the same except we used a shallower 4-layer Transformer (still with 768 hidden size).

Baselines: We list the reported numbers from many recent studies on GLUE and SQuAD, if available (more details in Appendix C). To reduce the variances in data processing/environments and provide fair comparisons, we also implement, pretrain, and fine-tune RoBERTa and ELECTRA under exact the same setting marked with "(Ours)".

Implementation Details: Our implementation is built upon the open-source release of MC-BERT (Xu et al., 2020) and its ELECTRA reproduction based on fairseq (Ott et al., 2019). Standard hyperparameters in pretraining and fine-tuning are used. We conduct pretraining on our Nvidia DGX-2 boxes. The hyperparameter settings and pretraining environments are listed in Appendix D.

5. Evaluation Results

In this section, we first present the overall evaluation results and ablations of various techniques in COCO-LM. Then we analyze the influence of its two pretraining tasks.

Group	Method	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	RTE	MRPC	STS-B	AVG
Baseline	RoBERTa (Ours)	85.61/85.51	91.34	91.80	93.86	58.64	69.03	87.50	86.53	83.03
	ELECTRA (Ours)	86.92/86.72	91.86	92.56	93.64	66.50	75.28	88.46	88.04	85.39
Original	COCO-LM Base	88.67/88.35	92.02	93.00	94.08	65.41	85.42	91.51	88.61	87.05
Pretraining Task	CLM Only	88.64/88.40	92.03	93.14	93.86	66.95	80.90	89.90	88.45	86.72
	SCL Only	88.62/88.14	92.14	93.45	93.86	64.70	82.57	90.38	89.35	86.86
Architecture	w/o. Rel-Pos	88.20/87.75	92.17	93.44	93.75	68.09	82.64	91.19	88.90	87.27
	w/o. Shallow-Aux	88.05/87.75	91.88	92.71	93.64	63.73	81.53	89.50	88.24	86.14
Noise Construction	w. Randomly Sampled Noises	84.94/84.74	91.36	91.08	91.63	40.82	70.50	87.34	84.86	80.30
	w. Fixed Auxiliary	87.94/87.98	92.03	92.96	93.18	64.68	81.53	89.98	88.22	86.32
CLM Setup (No SCL)	All-Token LM Only	87.17/86.97	91.74	92.58	93.75	61.02	73.54	88.70	87.70	84.51
	CLM w/o. Copy	88.02/87.87	91.81	93.11	94.53	65.71	76.60	89.42	88.17	85.91
	CLM w/o. Stop-grad	88.53/88.19	91.95	92.88	94.32	67.52	80.76	89.66	88.78	86.78

Table 2. Ablation results on GLUE Dev. Variations in each group include eliminate (w/o.), keep (Only) or switch (w.) one component.
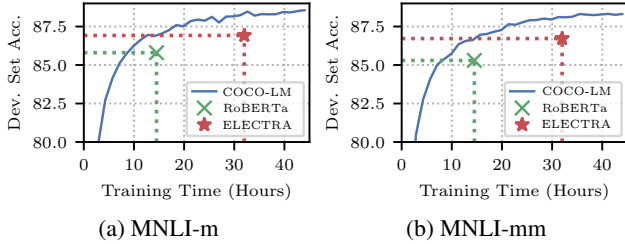

(a) MNLI-m   (b) MNLI-mm

Figure 2. COCO-LM Base accuracy on MNLI Dev. sets (y-axes) at different pretraining hours on four DGX-2 nodes (64 V100 GPUs). The final training hours and accuracy of RoBERTa (ours) and ELECTRA (Ours) are measured in the exact same settings and computing environments.

5.1. Overall Results

Table 1 shows the results of COCO-LM. The smaller GLUE tasks (CoLA, RTE, MPRC, and STS-B) are unstable: Many pretraining research omit them and more advanced fine-tuning strategies are required to achieve stable evaluations (Aghajanyan et al., 2020). On tasks where fine-tuning is more stable (*e.g.*, MNLI and SQuAD), COCO-LM provides the biggest improvement, for example, 1.6 points on MNLI-m, 1.42 on SQuAD EM, and 1.66 on GLUE AVG over best *base* setting baselines.

For fine-tuning and inference, COCO-LM does not incur extra computation cost as it has the same architecture with BERT besides relative position embedding. The extra computation cost in pretraining for better pretrained models is often considered a worthwhile one time investment. Still, we show the MNLI accuracy of COCO-LM at different pretraining hours versus the full RoBERTa (Ours) and ELECTRA (Ours) runs in Figure 2. The full pretraining of COCO-LM requires 37% more GPU hours compared to ELECTRA, with the cost from CLM and SCL, while both are more costly than RoBERTa due to the auxiliary network. However, COCO-LM turns out to be a better choice in *both accuracy and efficiency*: It outperforms ELECTRA and RoBERTa by more than 1 point on MNLI with the same compute, while requires less than 60% compute to reach the same accuracy.

5.2. Ablations

We conduct ablation studies on COCO-LM *base* on GLUE Dev. sets (Table 2). We reduce variance by fixing seeds and picking median-performing checkpoints from multiple pretraining runs. Nevertheless, there still exists randomness in pretraining that leads to ±0.1 on MNLI.

Pretraining Task. CLM or SCL individually provides significantly better performance than previous approaches on MNLI. Their advantages are better observed on different tasks, for example, CLM on MNLI-mm and SCL on STS-B. Combining the two in COCO-LM provides a better overall average. In later experiments we further analyze the behavior of these two pretraining tasks.

Architecture. The two notable differences in the Transformer architecture of COCO-LM is relative position encoding (Rel-Pos) and a shallow auxiliary network instead of a deeper but skinnier one in ELECTRA (Clark et al., 2020b). Removing Rel-Pos leads to better numbers on some tasks but significantly hurts MNLI; its GLUE AVG is contributed by CoLA. Using a 4-layer shallow auxiliary network is more effective than ELECTRA's 12-layer but 256-hidden dimension generator.

Pretraining Signal Construction. Similar to ELECTRA, COCO-LM uses the auxiliary network to sample more challenging noises to push the main language model. This is critical as the same model but with Randomly Sampled Noises performs worse than vanilla RoBERTa. Pretraining the two networks side-by-side provides a learning curriculum for the main network, as the noises from the auxiliary network start from near random and become more challenging along the way. Pretraining the main network with a pretrained and Fixed Auxiliary network performs worse.

CLM Setup. Switching the multi-task learning in CLM to the All-Token MLM loss (Clark et al., 2020b) significantly reduces the model's generalization ability. The copy mechanism and the stop gradient operation are also important to maintain CLM's effectiveness. The next experiments analyze how our CLM setup helps handle the challenging noises from the auxiliary network.

(a) Copy Acc. (Replaced)   (b) Copy Acc. (Original)   (c) CLM Acc. (Replaced)   (d) CLM Acc. (Original)
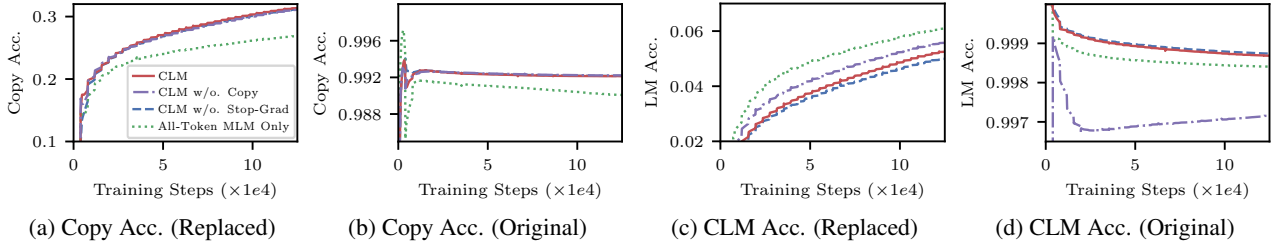
Figure 3. The training curves of CLM variations in COCO-LM. All x-axes are training steps (in 10K scale) and y-axes mark the tracked status: Copy mechanism Accuracy (*i.e.*, the main network's binary classification accuracy) on (a) the replaced tokens and (b) the original tokens. CLM Accuracy (*i.e.*, the accuracy of outputting the original tokens) on (c) the replaced tokens and (d) the original tokens.

Figure 4. The entropy of learned attention weights (after softmax) in different Transformer layers. The x-axis marks the layer index (smaller means closer to input tokens). The entropy is averaged on all tokens in the MNLI corpus without fine-tuning.

	All Words			**Stop Words**		
Model	Self	Rand.	Diff.	Self	Rand.	Diff.
RoBERTa	0.781	0.603	0.178	0.812	0.664	0.148
ELECTRA	0.722	0.603	0.119	0.682	0.606	0.076
COCO-LM	0.738	0.626	0.112	0.699	0.639	0.060
CLM Only	0.721	0.616	0.105	0.718	0.651	0.067
SCL Only	0.680	0.595	0.085	0.669	0.619	0.050

Table 3. Average cosine similarity between representations of the same word in different contexts (Self), random word pairs (Rand.) and their differences (Diff.) in MNLI corpus, without fine-tuning.

5.4. More Contextualized Transformer

What are the differences made by pretraining a Transformer with COCO-LM? In the following, we analyze the learned attentions and token representations in the main Transformer pretrained by COCO-LM and baselines.

More Spread Attention Weights. We first calculate the entropy of the attention weights (Clark et al., 2019) learned by COCO-LM *base* variations and compare it with RoBERTa (Ours) in Figure 4. All COCO-LM variations have significantly higher attention entropy in their last four layers compared to RoBERTa, indicating each token attends to more other tokens rather than concentrates on a few. The more challenging noises in COCO-LM requires the main Transformer to consider a wider range of context.

More Contextualized Token Representation. Next, we calculate the self-similarity of the representations of a same word when appearing in different contexts (Ethayarajh, 2019)—A less self-similar token representation indicates the Transformer is more contextualized. The results are shown in Table 3.

Both ELECTRA and COCO-LM have significantly more contextualized representations than RoBERTa. Their challenging noises require the main Transformers to rely on more contexts to distinguish and recover replaced tokens, while `[MASK]` tokens in RoBERTa can be easily recognized by the model. Among COCO-LM variations, SCL Only is significantly more contextualized than CLM Only. We further analyze the behaviors and influences of the SCL task in the next experiment.
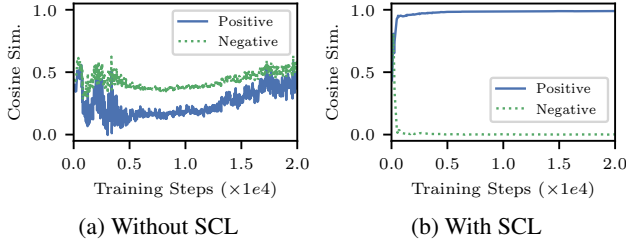
5.3. Language Modeling More Challenging Noises

In this experiment we analyze how CLM helps overcome the challenging noises and enable better pretraining the main network with a language modeling loss. In Figure 3, we show the pretraining curves of CLM and its three variations: CLM without copy mechanism (w/o. Copy), CLM without stop gradient operation (w/o. Stop-Grad), and All-Token MLM Only which only uses the LM loss.

The copy mechanism is important to help the model detect challenging noises: CLM (w/o. Copy) mistakes many original tokens as noises, with much worse LM accuracy on orignal tokens (Figure 3d). This hurts its generalization ability as shown in Table 2. Also, the noises from the auxiliary network make it quite challenging to learn to copy and correct solely from the language modeling loss, as can be seen from the a large gap between the copy accuracy of All-Token MLM Only and the rest (Figure 3a & 3b). The gap becomes even larger as pretraining goes on, showing the training of $p_{copy}(\cdot)$ is disturbed too much and never recovers. The stop gradient operation further helps avoid the disturbance from the hard LM loss to the classification loss.

In summary, the multi-task learning and stop gradient designs in CLM are important for the language model to effectively learn from the challenging pretraining signals instead of being confused by it.

(a) Without SCL

(b) With SCL

Figure 5. The cosine similarity between the `[CLS]` embeddings of positive and negative sequence pairs during pretraining.
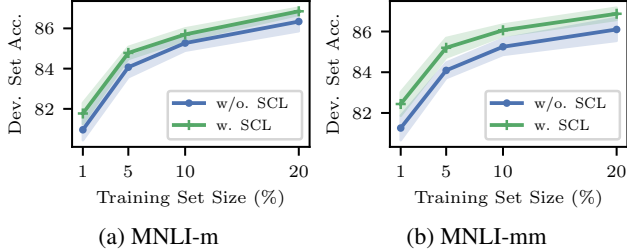

(a) MNLI-m

(b) MNLI-mm

Figure 6. Few-shot accuracy on MNLI with a fraction of MNLI training set used (x-axes). The error bars mark the max/min and the solid lines are the average of five fine-tuning runs.

5.5. Contrastive Learning Analyses

The last group of experiments show various notable characteristics of sequence contrastive learning. All the experiments are conducted under the *base* pretraining setting.

Contrastive Learning As Regularization. Our contrastive learning task is simple: Matching a positive sequence pair (*i.e.*, cropped sub-sequence and MLM corrupted sequence) among random pairs. Even without SCL, one would expect Transformer to map two sequences with many overlapping terms closer in the representation space by default. However, as shown in Figure 5, this is not the case: When pretrained without SCL, the cosine similarity of the positive pairs is actually lower than random negatives. The representation space without SCL is also so anisotropic that random pairs have near 0.5 cosine similarity. The explicit training at sequence level with SCL is necessary to regularize the sequence representation space to align similar sequences and decouple random ones.

Better Few-Shot Ability. One advantage of the more regularized sequence representations with SCL is the improved few-shot ability. As shown in Figure 6, SCL provides notable improvements under few-shot settings. Specifically, "w. SCL" outperforms "w/o. SCL" by $0.8/0.5$ on MNLI-m/mm when only 1% fine-tuning labels are used. With only 10% MNLI labels, "w. SCL" reaches $85.7/86.06$ MNLI which is better than RoBERTa (Ours) fine-tuned with full data. Using 20% labels, it performs on par with ELECTRA (Ours) fine-tuned with full data.
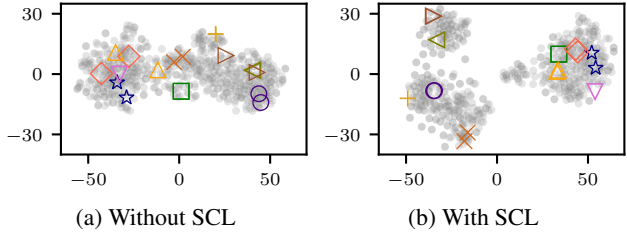

(a) Without SCL

(b) With SCL

Figure 7. The t-SNE plots of learned sequence representations with or without SCL. The points are sampled from the most semantically similar sentences pairs from STS-B (with 5-score labels). The `[CLS]` embeddings are obtained without fine-tuning. Some similar pairs are random selected and marked by the same shapes.

Alignment and Uniformity. Another advantage of contrastive learning known in visual representations is to provide better alignment of related pairs and to allocate random points more uniformly in the space (Wang & Isola, 2020). To study whether this holds in language representation learning, we plot the representations of semantically similar STS-B sentence pairs from COCO-LM in Figure 7 using t-SNE (Coenen et al., 2019). The similar sentence pairs (marked by same shapes) are *aligned* closer when pretrained with SCL. Their average cosine similarity is 0.925 when pretrained with SCL, while is 0.863 without SCL.

The uniformity is less observed. Both figures show non-uniform patterns, perhaps because the random negatives used in SCL are not sufficient to regularize the representation space. More sophisticated negative sample construction might improve the uniformity of the language representation space (He et al., 2019; Xiong et al., 2020).

6. Conclusions

In this paper, we present COCO-LM, a self-supervised learning framework that pretrains language models via correcting and contrasting text sequences with more challenging noises. The advantages of COCO-LM over previous pretraining approaches include pretraining with more challenging noises from the auxiliary language model, a multi-task corrective language modeling setting that robustly learns to recover original tokens, and a sequence contrastive learning task that regularizes sequence representations during pretraining.

Our experiments demonstrate that COCO-LM not only provides better generalization ability, but also enjoys higher efficiency in terms of downstream task performance achieved per pretraining hour. More importantly, we conduct extensive analyses on the influence of each technique in COCO-LM and their behaviors in different conditions. We hope our studies will inspire more future explorations for more effective and efficient pretraining frameworks including better construction of pretraining signals, more contrastive learning techniques, and new pretraining tasks.

References

Aghajanyan, A., Shrivastava, A., Gupta, A., Goyal, N., Zettlemoyer, L., and Gupta, S. Better fine-tuning by reducing representational collapse. *arXiv preprint arXiv:2008.03156*, 2020.

Bao, H., Dong, L., Wei, F., Wang, W., Yang, N., Liu, X., Wang, Y., Piao, S., Gao, J., Zhou, M., and Hon, H.-W. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *Preprint*, 2020.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

Bentivogli, L., Clark, P., Dagan, I., and Giampiccolo, D. The fifth pascal recognizing textual entailment challenge. In *TAC*, 2009.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. 2020.

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., and Specia, L. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, 2020.

Chi, Z., Dong, L., Wei, F., Yang, N., Singhal, S., Wang, W., Song, X., Mao, X.-L., Huang, H., and Zhou, M. Infoxlm: An information-theoretic framework for cross-lingual language model pre-training. *arXiv preprint arXiv:2007.07834*, 2020.

Clark, K., Khandelwal, U., Levy, O., and Manning, C. D. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 276–286, August 2019.

Clark, K., Luong, M.-T., Le, Q., and Manning, C. D. Pre-training transformers as energy-based cloze models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 285–294, 2020a.

Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020b.

Coenen, A., Reif, E., Yuan, A., Kim, B., Pearce, A., Viégas, F., and Wattenberg, M. Visualizing and measuring the geometry of bert. *arXiv preprint arXiv:1906.02715*, 2019.

Dagan, I., Glickman, O., and Magnini, B. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pp. 177–190. Springer, 2005.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. 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*, pp. 4171–4186, 2019.

Dolan, W. B. and Brockett, C. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., Gao, J., Zhou, M., and Hon, H.-W. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pp. 13063–13075, 2019.

Ethayarajh, K. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv preprint arXiv:1909.00512*, 2019.

Fang, H. and Xie, P. Cert: Contrastive self-supervised learning for language understanding. *arXiv preprint arXiv:2005.12766*, 2020.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021.

Giampiccolo, D., Magnini, B., Dagan, I., and Dolan, B. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pp. 1–9, 2007.

Gunel, B., Du, J., Conneau, A., and Stoyanov, V. Supervised contrastive learning for pre-trained language model fine-tuning. *arXiv preprint arXiv:2011.01403*, 2020.

Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M.-W. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*, 2020.

Haim, R. B., Dagan, I., Dolan, B., Ferro, L., Giampiccolo, D., Magnini, B., and Szpektor, I. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, 2006.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019.

He, P., Liu, X., Gao, J., and Chen, W. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L., and Levy, O. Spanbert: Improving pre-training by representing and predicting spans. *arXiv preprint arXiv:1907.10529*, 2019.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Ke, G., He, D., and Liu, T.-Y. Rethinking the positional encoding in language pre-training. *arXiv preprint arXiv:2006.15595*, 2020.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

Lewis, M., Ghazvininejad, M., Ghosh, G., Aghajanyan, A., Wang, S., and Zettlemoyer, L. Pre-training via paraphrasing. *Advances in Neural Information Processing Systems*, 33, 2020.

Li, B., Zhou, H., He, J., Wang, M., Yang, Y., and Li, L. On the sentence embeddings from pre-trained language models. *arXiv preprint arXiv:2011.05864*, 2020.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*, 2019.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

Pham, H., Xie, Q., Dai, Z., and Le, Q. V. Meta pseudo labels. *arXiv preprint arXiv:2003.10580*, 2020.

Purushwalkam, S. and Gupta, A. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. *arXiv preprint arXiv:2007.13916*, 2020.

Qu, Y., Shen, D., Shen, Y., Sajeev, S., Han, J., and Chen, W. Coda: Contrast-enhanced and diversity-promoting data augmentation for natural language understanding. *arXiv preprint arXiv:2010.08670*, 2020.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, 2016.

Ravula, A., Alberti, C., Ainslie, J., Yang, L., Pham, P. M., Wang, Q., Ontanon, S., Sanghai, S. K., Cvicek, V., and Fisher, Z. Etc: Encoding long and structured inputs in transformers. 2020.

Roberts, A., Raffel, C., and Shazeer, N. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*, 2020.

Rosset, C., Xiong, C., Phan, M., Song, X., Bennett, P., and Tiwary, S. Knowledge-aware language model pretraining. *arXiv preprint arXiv:2007.00655*, 2020.

Sennrich, R., Haddow, B., and Birch, A. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

Shankar, I., Nikhil, D., and Kornél, C. First quora dataset release: Question pairs, 2017. URL https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Song, K., Tan, X., Qin, T., Lu, J., and Liu, T.-Y. Mass: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning*, pp. 5926–5936, 2019.

Thakur, N., Reimers, N., Daxenberger, J., and Gurevych, I. Augmented sbert: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. *arXiv preprint arXiv:2010.08240*, 10 2020. URL https://arxiv.org/abs/2010.08240.

Trinh, T. H. and Le, Q. V. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Wang, T. and Isola, P. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. *arXiv preprint arXiv:2005.10242*, 2020.

Wang, W., Bi, B., Yan, M., Wu, C., Bao, Z., Xia, J., Peng, L., and Si, L. Structbert: Incorporating language structures into pre-training for deep language understanding. *arXiv preprint arXiv:1908.04577*, 2019.

Warstadt, A., Singh, A., and Bowman, S. R. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

Williams, A., Nangia, N., and Bowman, S. 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, Volume 1 (Long Papers)*, pp. 1112–1122, 2018.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Wu, Z., Wang, S., Gu, J., Khabsa, M., Sun, F., and Ma, H. Clear: Contrastive learning for sentence representation. *arXiv preprint arXiv:2012.15466*, 2020.

Xiong, L., Xiong, C., Li, Y., Tang, K.-F., Liu, J., Bennett, P., Ahmed, J., and Overwijk, A. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*, 2020.

Xu, Z., Gong, L., Ke, G., He, D., Zheng, S., Wang, L., Bian, J., and Liu, T.-Y. Mc-bert: Efficient language pre-training via a meta controller. *arXiv preprint arXiv:2006.05744*, 2020.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Advances in Neural Information Processing Systems*, pp. 5754–5764, 2019.

Ye, Q., Li, B. Z., Wang, S., Bolte, B., Ma, H., Ren, X., Yih, W.-t., and Khabsa, M. Studying strategically: Learning to mask for closed-book qa. *arXiv preprint arXiv:2012.15856*, 2020.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.

	Size	Task	Metric(s)	Domain
MNLI	393K	Inference	Accuracy	Misc.
QQP	364K	Similarity	Accuracy/F1	Social QA
QNLI	108K	QA/Inference	Accuracy	Wikipedia
SST-2	67K	Sentiment	Accuracy	Movie Reviews
CoLA	8.5K	Acceptability	Matthews corr.	Misc.
RTE	2.5K	Inference	Accuracy	Misc.
MRPC	3.7K	Paraphrase	Accuracy/F1	News
STS-B	5.7K	Similarity	Pearson/Spearman.	Misc.

Table 4. The list of benchmarks in GLUE, their training data size, language tasks, evaluation metrics, and domain of corpus.

A. GLUE Tasks

We provide more details of the tasks included in the GLUE benchmark. Their statistics are listed in Table 4.

MNLI: Multi-genre Natural Language Inference (Williams et al., 2018) contains 393K train examples obtained via crowdsourcing. The task is to predict whether a given premise sentence entails, contradicts or neutral with respect to a given hypothesis sentence.

QQP: Question Pairs (Shankar et al., 2017) contains 364K train examples from the Quora question-answering website. The task is to determine whether a pair of questions asked are semantically equivalent.

QNLI: Question Natural Language Inference contains 108K train examples derived from the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016). The task is to predict whether a given sentence contains the answer to a given question sentence.

SST-2: Stanford Sentiment Treebank (Socher et al., 2013) contains 67K train examples extracted from movie reviews with human-annotated sentiment scores. The tasks is to determine if the sentence has positive or negative sentiment.

CoLA: Corpus of Linguistic Acceptability (Warstadt et al., 2019) contains 8.5K train examples from books and journal articles on linguistic theory. The task is to determine whether a given sentence is linguistically acceptable or not.

RTE: Recognizing Textual Entailment (Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009) contains 2.5K train examples from textual entailment challenges. The task is to predict whether a given premise sentence entails a given hypothesis sentence or not.

MRPC: Microsoft Research Paraphrase Corpus (Dolan & Brockett, 2005) contains 3.7K train examples from online news sources. The task is to predict whether two sentences are semantically equivalent or not.

STS-B: Semantic Textual Similarity (Cer et al., 2017) contains 5.8K train examples drawn from multiple sources with human annotations on sentence pair semantic similarity. The task is to predict how semantically similar two sentences are on a 1 to 5 scoring scale.

B. SQuAD Fine-Tuning Details

Our pre-training code is built on top of the MC-BERT codebase including its data training pipelines. We have noticed that as an artifact of the data pre-processing, specifically the punctuation and white-space handling in fairseq, we have to adjust the start and end span offsets in the SQuAD training datasets to match those in the pre-processed data. After model inference, we have to post-process the predicted offsets in the processed data by reversing the previous adjustment to obtain the desired output offsets in the raw data format. As a result, our SQuAD implementation is not exactly the same with those used in previous approaches based on the huggingface codebase. This makes the SQuAD score comparison between our methods and previous reported methods not perfect due to the different pre-processing, post-processing applied, and also our smaller hyperparameter search space in fine-tuning. The SQuAD results of our own baseline runs are fair comparisons.

C. The Origins of Reported Baseline Scores

The baseline results listed in Table 1 are obtained from their original papers except the following: BERT from (Bao et al., 2020), RoBERTa GLUE from and SQuAD from (Bao et al., 2020), ELECTRA GLUE from (Xu et al., 2020), XLNet *base++* from (Bao et al., 2020), RoBERTa *base++* SQuAD from (Bao et al., 2020). When multiple papers report different scores for the same method, we use the highest of them in our comparisons.

D. More Implementation Details

Pretraining and Fine-tuning Costs. The *pretraining cost* of COCO-LM's CLM task is similar to ELECTRA, which is BERT plus the auxiliary network, which is about $1/3$ of the main network in size. The addition of SCL task requires one more forward and backward pass on the cropped sequence X^{crop}. With 256 V100 (32 GB Memory), one pretraining run takes about 20 hours in *base* setting and about two-three weeks in *base++* setting. The *fine-tuning costs* are the same with BERT plus relative positive encodings as the same Transformer model is used.

MLM Mode for Corrective Language Modeling. When creating the MLM replaced sequence X^{mlm}, we find it slightly improves the downstream task performance to disable dropout (*i.e.*, set the auxiliary MLM in inference mode) for computing the auxiliary network's output distribution where plausible replacing tokens are sampled. We hypothesize that this leads to more stable generation of challenging replaced tokens to be corrected by the main Transformer and thus improves downstream task results.

Parameters	Pre-training (*base*)	Pre-training (*base++*)	GLUE Fine-tuning	SQuAD Fine-tuning
Max Steps	125K	1.95M	-	-
Max Epochs	-	-	{3, 5, 10}	2
Peak Learning Rate	5e-4	2e-4	{1e-6, 5e-6, 2e-5}	{2e-5, 5e-5}
Batch Size	2048	2048	{16, 32}	{32, 48}
Learning Rate Decay	Linear	Linear	Linear	Linear
Warm-up Proportion	8%	2.5e-4%	6%	10%
Sequence Length	512	512	512	384
Adam ϵ	1e-6	1e-6	1e-6	1e-6
Adam (β_1, β_2)	(0.9, 0.98)	(0.9, 0.98)	(0.9, 0.98)	(0.9, 0.98)
Clip Norm	2.0	2.0	-	-
Dropout	0.1	0.1	0.1	-

Table 5. Hyperparameters used in pretraining and hyperparameter ranges searched for fine-tuning.

Masking Special Tokens for MLM Training. BERT only masks real tokens (other than artificial symbols like `[SEP]` and `[CLS]`) for MLM training, while RoBERTa also masks special tokens. We follow the RoBERTa setting which results in slightly improved performance for some tasks.

E. Hyperparameter Settings

Tuning hyperparameter of pretraining is often too costly and we keep most hyperparameters as default. The auxiliary MLM pretraining uses the standard 15% `[MASK]` ratio. The crop transformation in the SCL task uses 10% crop ratio, resulting in a sub-sequence 90% of the original sequence. The softmax temperature in the SCL task is 1. All pretraining tasks in COCO-LM have equal weights except $\lambda_{\text{copy}} = 50$ since the loss of the binary classification task is much lower those of the LM tasks, which are $30,000$-way classification tasks. All token embeddings are shared between the auxiliary Transformer and the main Transformer. The detailed hyperparameters used in pretraining and fine-tuning are listed in Table 5.

All reported methods use the exact same (or equivalent) set of hyperparameters for pretraining and fine-tuning for fair comparison. For COCO-LM and all the baselines implemented under our setting, all fine-tuning hyperparameters are searched per task; the medium/average of five/ten runs with the same set of five/ten random seeds are reported on GLUE/SQuAD.

F. More Discussions on PLM Research

Currently, the biggest challenge with PLM research development is perhaps its prohibitive computation cost. On the one hand, PLMs have influenced a wide range of tasks, and any further technical improvement can matter a lot for their downstream task applications, considering PLMs have been deployed in so many NLP tasks. On the other hand, its expensive computing cost and long experimental circles pose great challenges for careful and thorough studies of the problem space, as any test of new designs comes with a considerable computing cost—pretraining a new language model can easily consume thousands of dollars, or even millions for extra large models.

Such challenges call for more systematic evaluation pipelines that can accurately and reliably judge whether or not a new PLM is really better than previous ones. Currently, the evaluation of PLMs largely relies on GLUE-style benchmark which contains a set of different tasks that are weighted equally for PLM evaluations—usually the average performance over these tasks is treated as a final measure for the effectiveness of a PLM. However, we find that the small tasks in GLUE have very high variances which may provide unreliable indications for a PLM's performance. For example, on CoLA and RTE, fine-tuning with different random seeds from the same pretrained checkpoint can easily result in a 5-point difference between the best and the worst seed. In Table 1, the standard BERT model has the best STS-B performance, but that does not mean that later models are inferior to BERT. In contrast, large tasks like MNLI give relatively stable and consistent results for the same model pretrained/finetuned with different random seeds, and thus serve as better indicators for PLMs' effectiveness.

In this paper, we try to improve the robustness of our observations, for example, by reporting the downstream performance with different training time for future comparisons under limited computing budget, and also by making our code publicly available for the reproducibility of our model. We hope our efforts will facilitate more future research to improve the community's understanding and development of this important problem space.