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

Anonymous Author(s)
Affiliation
Address
email

Abstract

We present a self-supervised learning framework, COCO-LM, that pretrains Language Models by COrrecting and COntrasting corrupted text sequences. Following ELECTRA-style pretraining, COCO-LM employs an auxiliary language model to corrupt text sequences, upon which it constructs two new tasks for pretraining the main model. The first token-level task, Corrective Language Modeling, is to detect and correct tokens replaced by the auxiliary model, in order to better capture token-level semantics. The second sequence-level task, Sequence Contrastive Learning, is to align text sequences originated from the same source input while ensuring uniformity in the representation space. Experiments on GLUE and SQuAD demonstrate that COCO-LM not only outperforms recent state-of-the-art pretrained models in accuracy, but also improves pretraining efficiency. It achieves the MNLI accuracy of ELECTRA with 50% of its pretraining GPU hours. With 367 million parameters, it matches the MNLI average accuracy of Megatron with 3.9 billion parameters. Our code and pretrained models will be open-sourced.

1 Introduction

Pretrained language models (PLMs) have reshaped the way AI systems process natural language [11, 36, 39, 40]. Before task-specific fine-tuning, it is now a common practice to first pretrain the deep neural networks, often Transformers [53], via a self-supervised token-level language modeling framework [29, 32, 40]. Whether it is autoregressive [39], permutational [63], or masked language modeling (MLM) [11], the Transformer networks are pretrained to recover some omitted tokens using the rest of input texts. Then the language semantics captured during pretraining are conveyed to downstream tasks via the pretrained Transformer parameters [5, 8, 44].

Recent research [15, 17, 26, 43] observed several challenges in this self-supervised learning framework. One challenge is its efficiency. After pretrained for a while with the standard token-level language modeling, the networks have already captured the basic language patterns, making a large fraction of pretraining signals no longer informative. Linear improvement in the model effectiveness often requires exponentially more pretraining compute and parameters [26], which is unsustainable. Another challenge is the anisotropy of text representations from pretrained models. The sequence representations from many pretrained models are quite irregular [30, 43] and require dedicated fine-tuning approaches to be useful in sequence-level applications [33, 61].

Clark et al. [7] proposed a new pretraining strategy, ELECTRA, that uses an auxiliary language model ("generator") to replace tokens in input texts and pretrains the main Transformer ("discriminator") to detect replaced tokens. This improves the pretraining efficiency and effectiveness, but at the same time, pretraining only via binary classification hinders the model's usage on applications that require language modeling capability [6]. It could further distort the representation space as the Transformers are pretrained to output the same "non-replacement" label for all actual tokens.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

In this paper, we present a new self-supervised learning approach, COCO-LM, that pretrains Language Models by COrrecting and COntrasting corrupted text sequences. Following ELECTRA-style pretraining, COCO-LM employs an auxiliary model to corrupt the input texts, upon which it introduces two new pretraining tasks for the main Transformer, one at token level and one at sequence level. The token-level task, corrective language modeling (CLM), pretrains the main Transformer to detect and correct the tokens in the corrupted sequences. It uses a multi-task setup to combine the benefits of replaced token detection and language modeling. The sequence-level task, sequence contrastive learning (SCL), pretrains the model to align text sequences originated from the same source sequence and enforce uniformity of the representation space.

In our experiments on GLUE [54] and SQuAD [41] benchmarks, COCO-LM not only outperforms recent state-of-the-art pretraining approaches in effectiveness, but also significantly improves the pretraining efficiency. Under the same setting, COCO-LM matches the MNLI accuracy of RoBERTa and ELECTRA with 60% and 50% of their GPU hours in pretraining, respectively. With 367 million parameters, COCO-LM$_{\text{Large}}$ matches the MNLI average accuracy of Megatron$_{3.9\text{B}}$ [49], one of the largest BERT-style model with 3.9 billion parameters. Our analyses provide further insights on the benefit of SCL in ensuring alignment and uniformity in the representation space for robust generalization, as well as the advantage of CLM in learning token representations and improving the accuracy on token-level tasks.

2 Related Work

Various token-level tasks have been used to pretrain language models. The most classic auto-regressive language modeling is to predict a token given all the previous tokens, or subsequent ones [36, 39]. BERT uses masked language modeling (MLM) that recovers randomly masked tokens using the rest input. XLNet proposes permutation language modeling that conducts MLM in an autoregressive manner [63]; UniLM uses pseudo MLM which unifies autoregressive and MLM tasks [1, 13].

Sequence-level tasks are also explored in pretraining. They often pretrain the model to predict certain co-occurrences of sequence pairs in the corpus. For example, next sentence prediction [11], sentence ordering [28] and previous sentence prediction [56] concatenate two sentences (either correlated or random), feed them to the Transformer, and train the model to classify the pair.

Empirically, MLM is still among the most effective tasks to pretrain encoders [29, 32, 40]. RoBERTa [32] found the sentence-level task in BERT not benefitial and discarded it. BART [29] and T5 [40] both observed that MLM is often the most effective task. The empirical advantages of other pretraining tasks are more task-specific, for example, entity related masks for knowledge intensive applications [21, 25], and sequence-level tasks for long form text modeling [42].

Instead of randomly altering texts, ELECTRA [7] uses a smaller auxiliary Transformer pretrained by MLM to replace some tokens in the text sequences using its language modeling probability, and pretrains the main Transformer to detect the replaced tokens. ELECTRA achieves state-of-the-art accuracy in many language tasks [7]. Later, Clark et el. [6] developed ELECTRIC, which pretrains encoders by contrasting original tokens against negatives sampled from a cloze model. ELECTRIC re-enables the language modeling capability but underperforms ELECTRA in downstream tasks.

Our work is also related to contrastive learning which has shown great success in visual representation learning [4, 23, 34]. Its effectiveness of in language is more observed in the fine-tuning stage, for example, in sentence representation [17], dense retrieval [61], and GLUE fine-tuning [20].

3 Method

In this section, we present the preliminaries of pretraining language models, their challenges, and the self-supervised learning framework of COCO-LM.

3.1 Preliminary on Language Model Pretraining

In this work we focus on pretraining BERT-style bidirectional Transformer encoders [11] widely used in language representation tasks. We first recap the masked language modeling (MLM) task introduced by BERT [11] and then discuss the pretraining framework from ELECTRA [7].

BERT Pretraining uses the masked language modeling task (MLM) [11], which is to take an input sequence $X^{\text{orig}} = [x_1^{\text{orig}}, \ldots, x_i^{\text{orig}}, \ldots, x_n^{\text{orig}}]$, with 15% random tokens replaced by `[MASK]` symbols (*e.g.*, the i-th token), and train the model to predict the original tokens at the masked positions:

$$\left[x_1^{\text{orig}}, \ldots, \texttt{[MASK]}_i, \ldots, x_n^{\text{orig}}\right] \xrightarrow{\text{Transformer}} \boldsymbol{H} \xrightarrow{\text{MLM Head}} p_{\text{MLM}}(x|\boldsymbol{h}_i),$$

where the Transformer generates contextualized representations $\boldsymbol{H} = \{\boldsymbol{h}_i\}_{i=1}^n$. The MLM Head predicts the masked token from the vocabulary V using the hidden representation $\boldsymbol{h}_i$ and token embeddings $\boldsymbol{x}$. The pretraining minimizes the MLM loss on the set of masked positions $\mathcal{M}$. Specifically,

$$p_{\text{MLM}}(x|\boldsymbol{h}_i) = \frac{\exp(\boldsymbol{x}^\top \boldsymbol{h}_i)}{\sum_{x_t \in V} \exp(\boldsymbol{x}_t^\top \boldsymbol{h}_i)}; \quad \mathcal{L}_{\text{MLM}} = \mathbb{E}\left(-\sum_{i \in \mathcal{M}} \log p_{\text{MLM}}\left(x_i^{\text{orig}} \middle| \boldsymbol{h}_i\right)\right).$$

ELECTRA Pretraining uses two Transformers, a "generator" pretrained by MLM, and a "discriminator" pretrained using the generator's outputs. We refer them as *auxiliary* and *main* Transformers, as the former is discarded after pretraining.

The auxiliary model outputs a corrupted sequence X^{MLM} by sampling from its predicted probability:

$$x_i^{\text{MLM}} \sim p_{\text{MLM}}\left(x|\boldsymbol{h}_i\right), \text{ if } i \in \mathcal{M}\ ; \quad x_i^{\text{MLM}} = x_i^{\text{orig}}, \text{ else.} \tag{1}$$

The masked positions are replaced by sampled tokens considered plausible in context by the auxiliary Transformer, which are more deceiving than random replacements. ELECTRA uses a skinnier auxiliary network (*e.g.*, hidden dimension is $1/3$ of the main model) to control the signal difficulty.

The main Transformer takes X^{MLM} and classifies the replaced tokens:

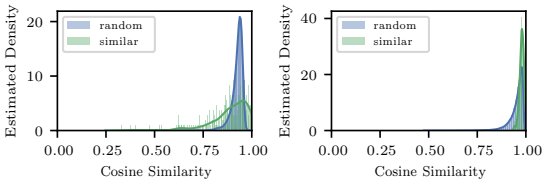
$$X^{\text{MLM}} \xrightarrow{\text{Main Transformer}} \boldsymbol{H} \xrightarrow{\text{RTD Head}} p_{\text{RTD}}\left(\mathbb{1}(x_i^{\text{MLM}} = x_i^{\text{orig}})\middle|\boldsymbol{h}_i\right),$$

where $\mathbb{1}(\cdot)$ is the indicator function. The Replaced Token Detection (RTD) head uses a sigmoid linear layer to output the binary probability, and the main Transformer is trained with binary cross entropy loss. The RTD task is trained on all tokens instead of masked ones and improves efficiency.

The two Transformers are pretrained jointly. The auxiliary model gradually generates more realistic replacement tokens and the main model learns to better detect them. This forms a natural learning curriculum and significantly improves ELECTRA's accuracy in downstream tasks [7].

3.2 Challenges of ELECTRA-Style Pretraining

Missing Language Modeling Benefits. The classification task in ELECTRA is easier and more stable [62], but raises two challenges. The first is the lack of language modeling capability which is a necessity in some tasks [6]. For example, prompt-based learning requires a language model to generate labels [16, 45, 46]. The second is that the binary classification task may not be sufficient to capture certain word-level semantics that are critical for token-level tasks.



(a) RoBERTa. (b) ELECTRA.

Figure 1: Distribution of cosine similarities between random/similar sequence pairs using their `[CLS]` embeddings from pretrained models. Histograms are distribution bins. Curves are their kernel density estimates.

Squeezing Representation Space. Another challenge is that the representations from Transformer-based language models often reside in a narrow cone, where two random sentences have high similarity scores (lack of uniformity), and closely related sentences may have more different representations (lack of alignment) [15, 17, 31]. Figure 1 illustrates such behaviors with random sentence pairs (from pretraining corpus) and semantically similar pairs (those annotated with maximum similarity from STS-B [3]). With RoBERTa, the cosine similarities of most random sentence pairs are near 0.8, bigger than many semantically similar pairs. The representation space from ELECTRA is even more squeezed. Nearly all sentence pairs, both random and similar ones,

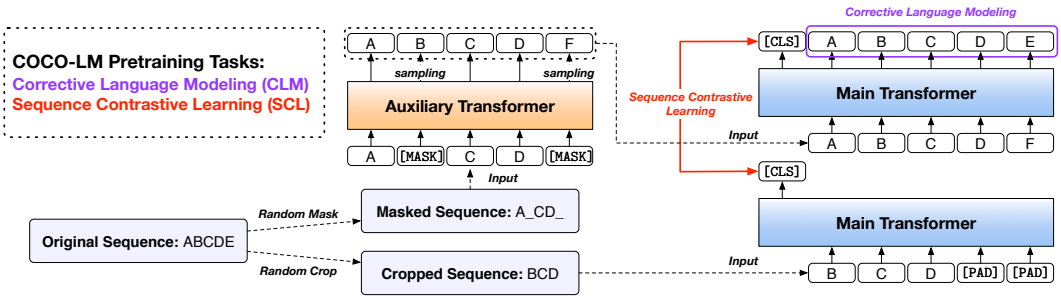


Figure 2: The overview of COCO-LM. The auxiliary Transformer is pretrained by MLM. Its corrupted text sequence is used as the main Transformer's pretraining input in Corrective Language Modeling and paired with the cropped original sequence for Sequence Contrastive Learning.

have around 0.9 cosine similarity. This may not be surprising as ELECTRA is pretrained to predict the same output ("non-replacement") for all tokens in these sequences. The irregular representation space raises the risk of degeneration [37, 55] and often necessitates sophisticated post-adjustment or fine-tuning to improve the sequence representations [17, 31, 33, 61].

3.3 COCO-LM Pretraining

COCO-LM also employs an auxiliary Transformer to construct the corrupted text sequence, as in Eqn. (1), but it introduces two new pretraining tasks upon the corrupted sequences, in order to address the challenges previously described. In the rest of this section, we present these two tasks and then the detailed configurations of COCO-LM. Its framework is illustrated in Figure 2.

Corrective Language Modeling (CLM) trains the main Transformer to recover the original tokens, given the corrupted text sequence X^{MLM}:

$$X^{\text{MLM}} \xrightarrow{\text{Main Transformer}} \boldsymbol{H} \xrightarrow{\text{CLM Head}} p_{\text{CLM}}(x|\boldsymbol{h}_i).$$

The CLM Head uses the hidden representations $\boldsymbol{H}$ to output a language modeling probability, instead of a binary classification score. The forward pass of the CLM Head is the same as All-Token MLM, an ablation version of ELECTRA [7] that consists of a language modeling layer and a binary classification layer for the copy mechanism:

$$p_{\text{LM}}(x_i|\boldsymbol{h}_i) = \mathbb{1}\left(x_i = x_i^{\text{MLM}}\right) p_{\text{copy}}(1|\boldsymbol{h}_i) + p_{\text{copy}}(0|\boldsymbol{h}_i) \frac{\exp(\boldsymbol{x}_i^\top \boldsymbol{h}_i)}{\sum_{x_t \in V} \exp(\boldsymbol{x}_t^\top \boldsymbol{h}_i)},$$
$$p_{\text{copy}}(y_i|\boldsymbol{h}_i) = \exp(y_i \cdot \boldsymbol{w}_{\text{copy}}^\top \boldsymbol{h}_i) / \left(\exp(\boldsymbol{w}_{\text{copy}}^\top \boldsymbol{h}_i) + 1\right),$$

where $\boldsymbol{w}_{\text{copy}}$ is a learnable weight and $p_{\text{copy}}(y_i|\boldsymbol{h}_i)$ is the copy mechanism ($y_i = 1$ when the input token is original and can be directly copied to the output; $y_i = 0$ when the input token needs to be corrected by generating another token from the vocabulary).

In ELECTRA, All-Token MLM performs worse than RTD [7]. Language modeling on the corrupted text sequence X^{MLM} is hard as the replaced tokens from the auxiliary model are more deceiving than `[MASK]`. To improve the language model learning, different from All-Token MLM, CLM employs a multi-task setup that combines the RTD task to explicitly train the copy mechanism $p_{\text{copy}}(\cdot)$:

$$\mathcal{L}_{\text{copy}} = -\mathbb{E}\left(\sum_{i=1}^{n} \mathbb{1}\left(x_i^{\text{MLM}} = x_i^{\text{orig}}\right) \log p_{\text{copy}}(1|\boldsymbol{h}_i) + \mathbb{1}\left(x_i^{\text{MLM}} \neq x_i^{\text{orig}}\right) \log p_{\text{copy}}(0|\boldsymbol{h}_i)\right), \quad (2)$$
$$\mathcal{L}_{\text{LM}} = -\mathbb{E}\left(\sum_{i=1}^{n} \log p_{\text{LM}}\left(x_i^{\text{orig}}|\boldsymbol{h}_i\right)\right)$$
$$= -\mathbb{E}\left(\sum_{i=1}^{n} \log\left(\mathbb{1}\left(x_i^{\text{MLM}} = x_i^{\text{orig}}\right) p_{\text{copy}}^{\text{sg}}(1|\boldsymbol{h}_i) + p_{\text{copy}}^{\text{sg}}(0|\boldsymbol{h}_i) \frac{\exp(\boldsymbol{x}_i^\top \boldsymbol{h}_i)}{\sum_{x_t \in V} \exp(\boldsymbol{x}_t^\top \boldsymbol{h}_i)}\right)\right),$$
$$\mathcal{L}_{\text{CLM}} = \lambda_{\text{copy}} \mathcal{L}_{\text{copy}} + \mathcal{L}_{\text{LM}}.$$

The hyperparameter λ_{copy} balances the weights of the two tasks. The binary cross entropy loss in Eqn. (2) explicitly trains the copy probability. We also use stop gradient ($\texttt{sg}$) to decouple the gradient backpropagation to $p_{\text{copy}}(\cdot)$ from the LM task. This way, the main Transformer first learns the easier classification task and then uses it to help learn the harder LM task.

CLM combines the advantages of MLM and ELECTRA: The main Transformer is trained with a language modeling objective on all tokens with the help of the binary classification task, thus enjoying the efficiency benefits of ELECTRA and preserving the language modeling benefits.

Sequence Contrastive Learning (SCL) forms a contrastive learning objective upon the sequence encodings to learn more robust representations. Broadly, contrastive learning is to align a positive pair of instances, often different views of the same information [4, 34], in contrast to unrelated negative instances [23, 61]. The different views are often obtained by applying data augmentations on the same input, for example, rotation, cropping, and blurring on visual representations [4, 34], so that the neural networks can learn representations robust to these data alterations.

In COCO-LM, the corrupted sequence X^{MLM} already provides a form of data augmentation. We pair it with another argumentation, X^{crop}, a randomly cropped contiguous span of X^{orig} (the length of X^{crop} is 90% of X^{orig} so that the major sequence meaning is preserved), to construct the positive pair and to contrast with random negatives.

Specifically, a training batch B in SCL includes a random set of corrupted and cropped sequences: $B = \{(X_1^{\text{MLM}}, X_1^{\text{crop}}), \ldots, (X_N^{\text{MLM}}, X_N^{\text{crop}})\}$, with X_k^{MLM} and X_k^{crop} originated from X_k^{orig}. A positive contrastive pair (X, X^+) consists of either $(X_k^{\text{MLM}}, X_k^{\text{crop}})$ or $(X_k^{\text{crop}}, X_k^{\text{MLM}})$ (symmetrical contrast). The negative instances are all the remaining sequences in the batch $B^- = B \setminus \{(X, X^+)\}$. The contrastive loss is formulated as:

$$\begin{aligned}\mathcal{L}_{\text{SCL}} &= -\mathbb{E}\left(\log \frac{\exp(\cos(\boldsymbol{s}, \boldsymbol{s}^+)/\tau)}{\exp(\cos(\boldsymbol{s}, \boldsymbol{s}^+)/\tau) + \sum_{X^- \in B^-} \exp(\cos(\boldsymbol{s}, \boldsymbol{s}^-)/\tau)}\right), \\ &= -\mathbb{E}\left(\cos(\boldsymbol{s}, \boldsymbol{s}^+)/\tau - \log\left(\exp(\cos(\boldsymbol{s}, \boldsymbol{s}^+)/\tau) + \sum_{X^- \in B^-} \exp\left(\cos(\boldsymbol{s}, \boldsymbol{s}^-)/\tau\right)\right)\right), \quad (3)\end{aligned}$$

where $\boldsymbol{s}, \boldsymbol{s}^+, \boldsymbol{s}^-$ are the representations of X, X^+, X^-, respectively, from the main Transformer (*i.e.*, $\boldsymbol{h}_{\texttt{[CLS]}}$). The similarity metric is cosine similarity (cos) and the temperature τ is set to 1.

As shown in Wang et al. [55], the first term in Eqn. (3) ($\cos(\boldsymbol{s}, \boldsymbol{s}^+)$) improves *alignment* of the space. It encourages representations to be robust to the corruptions and the alterations on the original text. The second term in Eqn. (3) promotes *uniformity*. It pushes unrelated sequences apart in the representation space and ensures low cosine similarity between random data points. Several studies have observed improved generalization ability from better alignment and uniformity [17, 37, 55].

Aligning X^{MLM} with X^{crop} requires the main Transformer to produce sequence representations robust to both token-level (*i.e.*, MLM replacements) and sequence-level (*i.e.*, cropping) alterations. The model is thus encouraged to reason more using partially altered sequences to recover the original information.

Overall Training. COCO-LM uses the following loss function:

$$\mathcal{L}_{\text{COCO-LM}} = \mathcal{L}_{\text{MLM}}^{\text{Aux.}} + \mathcal{L}_{\text{CLM}}^{\text{Main}} + \mathcal{L}_{\text{SCL}}^{\text{Main}}. \quad (4)$$

The auxiliary Transformer is pretrained by masked language modeling (MLM) and generates corrupted sequences. The main Transformer is pretrained to correct the corruption (CLM) and to contrast the corrupted sequences with the cropped sequences (SCL). The two Transformers are pretrained jointly with the loss in Eqn. (4). The main Transformer is used in downstream applications.

Network Configurations. Similar to ELECTRA, the auxiliary Transformer is smaller than the main model, but we use different configurations in the auxiliary model: (1) We reduce the number of layers to $1/3$ or $1/4$ but keep its hidden dimension the same with the main model, instead of shrinking its hidden dimensions; (2) We disable the dropouts in it when sampling replacement tokens. We find such configurations empirically more effective and use them as the backbone of COCO-LM. The main Transformer follows the standard architecture of BERT/ELECTRA and can be easily adopted by pipelines in downstream applications with almost no changes.

Model	Params	GLUE DEV Single Task									SQuAD 2.0 DEV	
		MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	RTE	MRPC	STS-B	AVG	EM	F1
Base Setting: BERT Base Size, Wikipedia + Book Corpus (16GB)												
BERT [11]	110M	84.50/-	91.30	91.70	93.20	58.90	68.60	87.30	89.50	83.13	73.70	76.30
RoBERTa [32]	125M	84.70/-	–	–	92.70	–	–	–	–	–	–	79.70
XLNet [63]	110M	85.84/85.43	–	–	92.66	–	–	–	–	–	78.46	81.33
ELECTRA [7]	110M	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 [62]	110M	85.68/85.24	89.65	91.34	92.34	62.10	74.96	85.96	88.01	83.73	–	–
DeBERTa [24]	134M	86.30/86.20	–	–	–	–	–	–	–	–	79.30	82.50
TUPE [27]	110M	86.21/86.19	91.30	92.17	93.26	63.56	73.56	89.89	89.23	84.90	–	–
RoBERTa (Ours)	110M	85.76/85.54	91.31	91.98	**93.69**	60.10	68.23	87.25	88.49	83.34	77.71	80.51
ELECTRA (Ours)	110M	86.91/86.73	91.86	92.57	93.58	**66.17**	75.09	88.24	89.69	85.50	79.74	82.58
COCO-LM	110M	**88.51/88.29**	**92.04**	**93.14**	93.23	63.86	**84.84**	**91.42**	**90.26**	**87.15**	**82.41**	**85.23**
Base++ Setting: BERT Base Size, Bigger Training Data, and/or More Training Steps												
XLNet [63]	110M	86.80/-	91.40	91.70	94.70	60.20	74.00	88.20	89.50	84.56	80.20	–
RoBERTa [32]	125M	87.60/-	91.90	92.80	94.80	63.60	78.70	90.20	91.20	86.35	80.50	83.70
UniLM V2 [1]	110M	88.50/-	91.70	93.50	**95.10**	65.20	81.30	**91.80**	91.00	87.09	83.30	86.10
DeBERTa [24]	134M	88.80/88.50	–	–	–	–	–	–	–	–	83.10	86.20
CLEAR [60]	110M	86.70/-	90.00	92.90	94.50	64.30	78.30	89.20	89.80	85.71	–	–
COCO-LM	134M	**89.70/89.59**	**92.28**	**93.98**	94.50	**66.96**	**85.56**	90.93	**91.53**	**88.17**	**83.96**	**86.79**
Large++ Setting: BERT Large Size, Bigger Training Data, and More Training Steps												
XLNet [63]	360M	90.80/90.80	92.30	94.90	**97.00**	69.00	85.90	90.80	92.50	89.15	87.90	90.60
RoBERTa [32]	356M	90.20/90.20	92.20	94.70	96.40	68.00	86.60	90.90	92.40	88.93	86.50	89.40
ELECTRA [7]	335M	90.90/-	92.40	95.00	96.90	69.10	88.00	90.80	**92.60**	89.43	**88.00**	90.60
DeBERTa [24]	384M	91.10/91.10	92.30	**95.30**	96.80	70.50	–	–	–	–	**88.00**	**90.70**
COCO-LM	367M	**91.32/91.48**	**92.89**	95.26	96.22	**72.59**	**91.34**	**91.18**	92.23	**90.39**	87.80	90.48
$\text{Megatron}_{1.3\text{B}}$ [49]	1.3B	90.90/91.00	92.60	–	–	–	–	–	–	–	87.10	90.20
$\text{Megatron}_{3.9\text{B}}$ [49]	3.9B	91.40/91.40	92.70	–	–	–	–	–	–	–	88.50	91.20

Table 1: Results on GLUE and SQuAD 2.0 development set. All results are single-task, single-model fine-tuning. Results not available in public reports are marked as "–". DeBERTa reported RTE, MRPC and STS-B results by fine-tuning from MNLI checkpoints which are not single-task results. We use Spearman correlation for STS, Matthews correlation for CoLA, and accuracy for the rest on GLUE. AVG is the average of the eight tasks on GLUE. All baseline results unless marked by (Ours) are reported by previous research.

4 Experimental Setup

Pretraining Settings. We employ three standard settings, *base*, *base*++, and *large*++. *Base* is the $\text{BERT}_{\text{Base}}$ training configuration [11]: Pretraining on Wikipedia and BookCorpus [64] (16 GB of texts) for 256 million samples on 512 token sequences (125K batches with 2048 batch size). We use the same corpus and $32,768$ uncased BPE vocabulary [47] as with TUPE [27].

Base++ trains the base size model with larger corpora and/or more training steps. Following recent research [1, 32, 63], we add in OpenWebText [19], CC-News [32] and STORIES [52], to a total of 160 GB texts, and train for 4.0 billion (with 2048 batch size) samples [32]. We follow the prepossessing of UniLMV2 [1] and use $64,000$ cased BPE vocabulary.

Large++ uses the same training corpora as *base*++ but pretrains for 2.7 billion samples (2048 batch size) , about 68% of $\text{RoBERTa}_{\text{Large}}$. Its Transformer configuration is the same with $\text{BERT}_{\text{Large}}$ [11].

Model Architecture. Our *base*/*base*++ model uses the $\text{BERT}_{\text{Base}}$ architecture [11]: 12 layer Transformer, 768 hidden size, plus T5 relative position encoding [40]. Our *large*++ model is the same with $\text{BERT}_{\text{Large}}$, 24 layer and 1024 hidden size, plus T5 relative position encoding [40]. Our auxiliary network uses the same hidden size but a shallow 4-layer Transformer in *base*/*base*++ and a 6-layer one in *large*++. When generating X^{MLM} we disable the dropout in the auxiliary model.

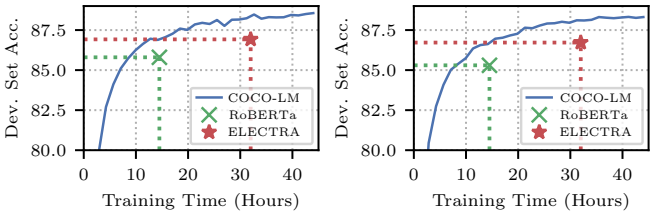
Downstream Tasks. We use the tasks included in GLUE [54] and SQuAD 2.0 reading compression [41]. The fine-tuning protocols use the open-source implementation from TUPE [27] on GLUE tasks and huggingface [59] on SQuAD. All our models are evaluated with the same fine-tuning protocols. The reported results are the median/average of five/ten random seeds in GLUE/SQuAD, to be consistent with recent research [1, 32]. Please refer to Appendix for more details.

Baselines. We compare with various pretrained models in each setting. To reduce the variances in data processing/environments, we also pretrain and fine-tune RoBERTa and ELECTRA under exact the same setting with COCO-LM, marked with "(Ours)". All numbers unless marked by "(Ours)" are from reported results in recent research (more details in Appendix C).

Implementation Details. Our implementation builds upon the open-source implementation from MC-BERT [62] and fairseq [35]. Standard hyperparameters in pretraining and fine-tuning are used. More details are listed in Appendix D.

Group	Method	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	RTE	MRPC	STS-B	AVG
	COCO-LM$_{\text{Base}}$	88.51/88.29	92.04	93.14	93.23	63.86	84.84	91.42	90.26	87.15
Pretraining Task	RTD Only	88.36/88.16	92.10	93.52	92.66	67.32	80.51	88.97	90.90	86.78
	CLM Only	88.63/88.40	92.03	93.15	93.69	67.35	80.14	89.95	90.40	86.90
	SCL + RTD	88.61/88.15	92.13	93.47	93.81	64.32	82.67	90.20	90.57	86.94
Network Setting	w/o. Rel-Pos	88.19/87.74	92.17	93.43	93.69	68.78	82.67	91.18	90.62	87.56
	w. ELECTRA's Auxiliary	88.01/87.73	91.89	92.70	93.46	64.33	81.23	89.46	89.70	86.33
Training Signal	w. Random Replacements	84.94/84.74	91.36	91.07	91.40	41.60	70.04	87.25	87.14	80.59
	w. Converged Auxiliary	88.29/88.08	91.97	92.79	94.27	64.16	78.34	90.44	90.17	86.29
CLM Setup	All-Token LM Only	87.17/86.96	91.75	92.62	93.69	60.63	74.01	88.48	89.67	84.74
	CLM w/o. Copy	88.04/87.86	91.81	93.10	94.38	66.64	76.90	89.46	90.05	86.28
	CLM w/o. Stop-grad	88.54/88.19	91.97	92.90	94.27	66.49	80.87	89.95	90.63	86.93

Table 2: Ablations on GLUE Dev. that eliminate (w/o.), keep (Only) or switch (w.) one component.



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

Figure 3: COCO-LM$_{\text{Base}}$ on MNLI Dev. (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) measured in the same settings are marked.

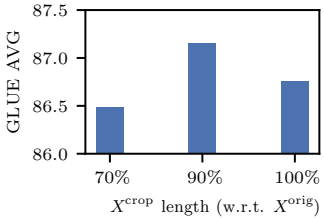


Figure 4: The performance of COCO-LM$_{\text{Base}}$ when pretrained with different crop fractions. The x-axis is the fraction of X^{orig} being kept (no cropping is 100%).

5 Evaluation Results

Three groups of experiments are conducted to evaluate COCO-LM and its two new pretraining tasks.

5.1 Overall Results and Ablations

Overall Results are listed in Table 1. Under all three settings, COCO-LM outperforms all recent state-of-the-art pretraining models on GLUE average and is competitive on SQuAD. It improves the state-of-the-art GLUE score by about one point under all three settings. COCO-LM also enjoys significantly better parameter efficiency. Using less than 10% parameters, COCO-LM$_{\text{Large}}$ matches the average MNLI accuracy of Megatron$_{3.9\text{B}}$, one of the largest pretrained BERT-style encoders.

Efficiency. In downstream tasks, the efficiency of COCO-LM is the same with BERT. In pretraining, the auxiliary model and SCL introduce extra cost. However, as shown in Figure 3, COCO-LM is more efficient in GPU hours. It outperforms RoBERTa & ELECTRA by 1+ point on MNLI with the same GPU hours and reaches their accuracy with around 60% & 50% GPU hours, respectively.

Ablation Studies. Table 2 shows the ablations of COCO-LM in *base* setting on GLUE DEV.

Pretraining Task. With only RTD, our backbone model with the shallow auxiliary Transformer is quite effective. CLM and SCL both provide additional improvements on MNLI and GLUE average. Their advantages are better observed on different tasks, for example, CLM on MNLI-mm and SCL on RTE and MRPC. Combining the two in COCO-LM provides better overall effectiveness. In later experiments, we further analyze the benefits of these two tasks.

Architecture. Removing Rel-Pos leads to better numbers on some tasks but significantly hurts MNLI. Using a shallow auxiliary network and keep the same hidden dimension (768) is more effective than ELECTRA's 12-layer but 256-hidden dimension generator.

Pretraining Signal Construction. Using randomly replaced tokens to corrupt text sequence hurts significantly. Using a converged auxiliary network to pretrain the main model also hurts. It is better to pretrain the two Transformers together, as the auxiliary model gradually increases the difficulty of the corrupted sequences and provides a natural learning curriculum for the main transformer.

CLM Setup. Disabling the multi-task learning and using All-Token MLM [7] reduces model accuracy. The copy mechanism is effective. The benefits of the stop gradient operation are more on stability.

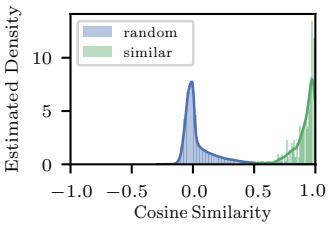
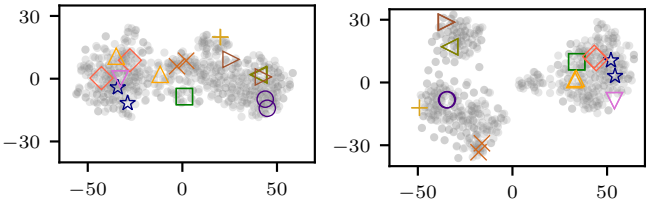
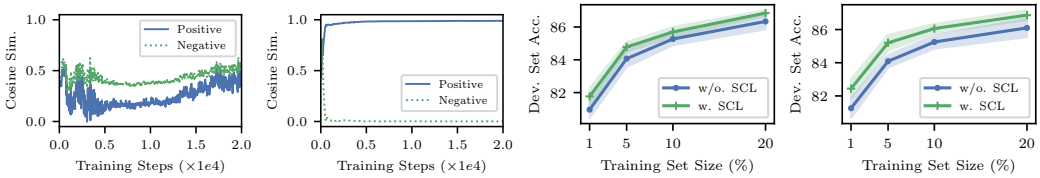


Figure 5: Distribution of cosine similarities of sequence pairs randomly sampled from pretraining corpus and most semantically similar pairs from STS-B using `[CLS]` from COCO-LM$_{\text{Base}}$.



(a) Without SCL (b) With SCL

Figure 6: The t-SNE of sequence representations learned 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 not fine-tuned. Some randomly selected similar pairs are marked by same shapes.



(a) Without SCL (b) With SCL (c) MNLI-m (d) MNLI-mm

Figure 7: Analyses of SCL. Figs. (a) and (b) show the average cosine similarity between the `[CLS]` embeddings of positive and negative contrast pairs during pretraining. Figs. (c) and (d) show the 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.

5.2 Analyses of Contrastive Learning with SCL

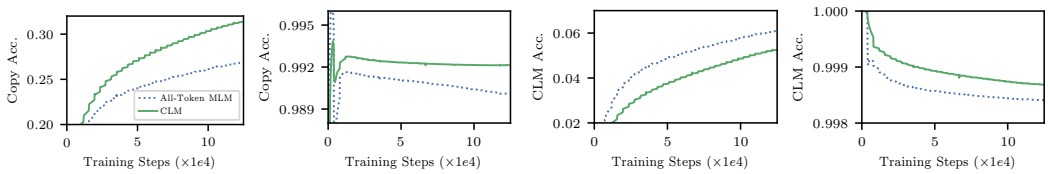
This group of experiments analyzes the behavior of SCL. All experiments use the *base* setting.

Ablation on Data Argumentation. Figure 4 shows the effects of the cropping operation when forming positive SCL pairs with the corrupted sequence. Using the original sequence results in worse GLUE accuracy. It is less informative as the model no longer needs to learn representations robust to sequence-level alteration. Cropping too much (*e.g.*, only keeping 70% of the original sequence), may hurt as it can alter the semantics too much. Empirically a simple alteration works the best, similar to the observations in recent research [4, 17, 23].

Alignment and Uniformity. Figure 5 plots the distribution of cosine similarities between random sequence pairs and similar ones using representations pretrained by COCO-LM. The representation space from COCO-LM is drastically different from those in Figure 1. With COCO-LM, similar pairs are more *aligned* and random pairs are distributed more *uniformly*. Many similar pairs have near 1 cosine similarity and are clearly separated from random pairs which center around 0. The t-SNE [9] plot in Figure 6 further demonstrates the benefits of SCL. 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. This better alignment and uniformity is achieved by COCO-LM with SCL via pretraining, without using task-specific data nor supervised labels.

Regularizing the Representation Learning for Better Few-Shot Ability. One would expect any pretrained Transformers to easily align a pair of corrupted sequence and cropped sequence as the two share about 80% tokens. However, as shown in Figure 7a, that is not the case: Without SCL, the cosine similarity of the positive pairs is even lower than random negatives. SCL is necessary to regularize the representation space and to reduce the risk of degeneration (Figure 7b).

Similar to empirical observations and theoretical analyses in recent research [15, 17, 55], a more regularized representation space results in better generalization ability in scenarios with limited labels. Figure 7c and 7d show the results when COCO-LM are fine-tuned with only a fraction of MNLI labels. The improvements brought by SCL are more significant when fewer fine-tuning labels are available. With 1% MNLI labels, pretraining with SCL improves MNLI-m/mm accuracy by $0.8/0.5$ compared to that without SCL. Using only $10\%/20\%$ labels, COCO-LM with SCL reaches similar MNLI accuracy with RoBERTa (Ours)/ELECTRA (Ours) fine-tuned with all labels, respectively.



(a) Copy Acc. (Replaced) (b) Copy Acc. (Original) (c) CLM Acc. (Replaced) (d) CLM Acc. (Original)

Figure 8: The copying accuracy and the language modeling accuracy (y-axes) of CLM and All-Token MLM at different pretraining steps (x-axes, in 10K scale). The accuracy is averaged on tokens that are replaced by the auxiliary Transformer (Replaced) or those from the original input text (Original).

5.3 Analyses of Language Modeling with CLM

The last group of experiments studies the effectiveness of CLM in pretraining and downstream tasks.

Ablations on Training Configurations. Figure 8 illustrates pretraining with CLM and All-Token MLM. The large gaps between the two groups demonstrate the difficulty of language modeling upon corrupted text sequences. It is quite a unbalanced task. For the majority of the tokens (Original) the task is simply to copy its input at the same position. For the replaced tokens ($7-8\%$ total), however, the model needs to detect the abnormality brought by the auxiliary model and recover the original token. Implicitly training the copy mechanism as part of the hard LM task is not effective: The copy accuracy of All-Token MLM is much lower, and thus the LM head may confuse original tokens with replaced ones. As shown in Table 2 and ELECTRA [7], pretraining with All-Token MLM performs worse than using the RTD task, which is equivalent to only training the copy mechanism. The multi-task learning of CLM is necessary for the main Transformer to stably learn the language modeling task upon the corrupted text sequence.

Better Representations for Token-Level Tasks. Table 3 shows word similarity analyses [14] and the SQuAD accuracy of COCO-LM when pretrained without CLM (SCL + RTD) and with CLM (full model). The RTD task pushes all token representations close as it trains the model to output the same category ("non-replacement") for all these tokens. Random tokens have 0.724 cosine similarity and the gap between Self and Random is small. With CLM, random tokens have lower similarities and the gap is larger, showing the semantics about the token itself is better preserved. On SQuAD which is a token-level task, the better token representations from CLM are beneficial and improve EM & F1 by $0.3+$.

Model	Word Similarity			SQuAD 2.0 DEV	
	Self	Rand	Gap	EM	F1
COCO-LM	0.747	0.517	**0.230**	**83.96**	**86.79**
w.o. CLM	0.832	0.724	0.108	83.63	86.48

Table 3: The word similarity on MNLI corpus (pretrained, no fine-tuning) from COCO-LM (with and without CLM) and their SQuAD results. *Self* is the average cosine similarity between the same word's $\boldsymbol{h}$ when appearing in different contexts [14]. *Random* is that of two random words. Gap is their difference. The analyses uses *base++* setting for robust observations.

6 Conclusions and Future Work

In this paper, we present COCO-LM, which pretrains language models using Corrective Language Modeling and Sequence Contrastive Learning upon corrupted text sequences. With standard pretraining settings and Transformer architectures, COCO-LM improves the accuracy on the GLUE and SQuAD benchmarks, while also being more efficient in utilizing pretraining computing resources and Transformer parameters.

One limitation of this work is that the contrastive pairs are constructed by simple cropping and MLM replacements. Recent research have shown the effectiveness of advanced data argumentation techniques in fine-tuning language models [17, 38, 51]. A future research direction is to explore better ways to construct contrast pairs in language model pretraining.

Despite the empirical advantage of this auxiliary-main dual model framework, the auxiliary Transformer training is not influenced by the main Transformer, thus the training signals from the auxiliary model may not be optimal for the main model. To better understand and automate the training of the auxiliary model is another important future research direction.

References

[1] Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Songhao Piao, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *ICML*, 2020.

[2] Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth pascal recognizing textual entailment challenge. In *TAC*, 2009.

[3] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, pages 1597–1607. PMLR, 2020.

[5] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 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*, pages 276–286, August 2019.

[6] Kevin Clark, Minh-Thang Luong, Quoc Le, and Christopher D Manning. Pre-training transformers as energy-based cloze models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 285–294, 2020.

[7] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020.

[8] Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. *arXiv preprint arXiv:2002.05867*, 2020.

[9] Andy Coenen, Emily Reif, Ann Yuan, Been Kim, Adam Pearce, Fernanda Viégas, and Martin Wattenberg. Visualizing and measuring the geometry of bert. *arXiv preprint arXiv:1906.02715*, 2019.

[10] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer, 2005.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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*, pages 4171–4186, 2019.

[12] William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[13] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13063–13075, 2019.

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

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

[16] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. In *Association for Computational Linguistics (ACL)*, 2021.

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

[18] Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9, 2007.

[19] Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`, 2019.

[20] Beliz Gunel, Jingfei Du, Alexis Conneau, and Ves Stoyanov. Supervised contrastive learning for pre-trained language model fine-tuning. *arXiv preprint arXiv:2011.01403*, 2020.

[21] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*, 2020.

[22] R Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, 2006.

[23] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19*, pages 9726–9735. IEEE, 2020.

[24] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

[25] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *arXiv preprint arXiv:1907.10529*, 2019.

[26] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[27] Guolin Ke, Di He, and Tie-Yan Liu. Rethinking the positional encoding in language pre-training. *arXiv preprint arXiv:2006.15595*, 2020.

[28] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[29] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[30] Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. On the sentence embeddings from pre-trained language models. *arXiv preprint arXiv:2011.05864*, 2020.

[31] Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. On the sentence embeddings from pre-trained language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, November 2020.

[32] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*, 2019.

[33] Yi Luan, Jacob Eisenstein, Kristina Toutanove, and Michael Collins. Sparse, dense, and attentional representations for text retrieval. *arXiv preprint arXiv:2005.00181*, 2020.

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

[35] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[36] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237, 2018.

[37] Senthil Purushwalkam and Abhinav Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. *arXiv preprint arXiv:2007.13916*, 2020.

[38] Yanru Qu, Dinghan Shen, Yelong Shen, Sandra Sajeev, Jiawei Han, and Weizhu Chen. Coda: Contrast-enhanced and diversity-promoting data augmentation for natural language understanding. *arXiv preprint arXiv:2010.08670*, 2020.

[39] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[40] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[41] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016.

[42] Anirudh Ravula, Chris Alberti, Joshua Ainslie, Li Yang, Philip Minh Pham, Qifan Wang, Santiago Ontanon, Sumit Kumar Sanghai, Vaclav Cvicek, and Zach Fisher. Etc: Encoding long and structured inputs in transformers. 2020.

[43] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[44] Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*, 2020.

[45] Timo Schick and Hinrich Schütze. Exploiting cloze questions for few-shot text classification and natural language inference. *Computing Research Repository*, arXiv:2001.07676, 2020.

[46] Timo Schick and Hinrich Schütze. It's not just size that matters: Small language models are also few-shot learners. *Computing Research Repository*, arXiv:2009.07118, 2020.

[47] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[48] Iyer Shankar, Dandekar Nikhil, and Csernai Kornél. First quora dataset release: Question pairs, 2017.

[49] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[50] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[51] Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. Augmented sbert: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. *arXiv preprint arXiv:2010.08240*, 10 2020.

[52] Trieu H Trinh and Quoc V Le. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*, 2018.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[54] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[55] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. *arXiv preprint arXiv:2005.10242*, 2020.

[56] Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Jiangnan Xia, Liwei Peng, and Luo Si. Structbert: Incorporating language structures into pre-training for deep language understanding. *arXiv preprint arXiv:1908.04577*, 2019.

[57] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

[58] Adina Williams, Nikita Nangia, and Samuel Bowman. 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)*, pages 1112–1122, 2018.

[59] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

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

[61] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*, 2020.

[62] Zhenhui Xu, Linyuan Gong, Guolin Ke, Di He, Shuxin Zheng, Liwei Wang, Jiang Bian, and Tie-Yan Liu. Mc-bert: Efficient language pre-training via a meta controller. *arXiv preprint arXiv:2006.05744*, 2020.

[63] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Advances in Neural Information Processing Systems*, pages 5754–5764, 2019.

[64] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 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*, pages 19–27, 2015.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We provide comprehensive evaluation results and comparison with prior work in Section 5 to validate the effectiveness of our proposed methods.
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We provide discussions about the long training cycle and expensive costs of pretraining in Appendix F.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendices A and E.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We report the median/average results of multiple random seeds, following previous studies. See Section 4.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Figure 3 and Appendix D
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 4 and Appendix A.
   (b) Did you mention the license of the assets? [No] The license can be found in the original source of the assets.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]