PRETRAINING ONE LANGUAGE MODEL FOR ALL WITH THE TEXT-TO-TEXT FRAMEWORK AND MODEL-GENERATED SIGNALS

Anonymous authors
Paper under double-blind review

ABSTRACT

Pretrained encoder-decoder language models provide the flexibility to unify various language scenarios into one text-to-text framework, but various recent studies raised concerns about their inferior pretraining efficiency and effectiveness compared to encoder only and decoder only models. In this paper, we improve the performance of encoder-decoder language models in unifying NLP tasks by pretraining with ELECTRA-style model-generated signals. We first show the challenges of pretraining encoder-decoder models (such as T5) using model-generated signals, including ill-formed target, label leakage, and training instability. We then propose Metro-T5, a new formulation of the denoising pretraining task and multi-task learning loss for encoder-decoder models to incorporate ELECTRA-Style pretraining. Metro-T5 outperforms T5 on a variety of language tasks in standard fine-tuning and prompt-based zero/few-shot scenarios. Our analysis shows Metro-T5 achieves similar generalization ability with much better efficiency, outperforming T0 (3B) in prompt-based learning with only 8% parameters and T5 in all tasks with fewer GPU hours. Our pretraining code and model checkpoints will be open-sourced.

1 INTRODUCTION

The pretrain-and-apply scheme has become the norm for nearly all language tasks. By pretraining neural networks, often Transformers (Vaswani et al., 2017), with language modeling tasks (Devlin et al., 2019), strong generalization ability is acquired in the pretrained language models (PLMs). The benefits are observed on many downstream scenarios, for example, "natural language understanding" (NLU), such as sequence prediction (Wang et al., 2018), and "natural language generation" (NLG), such as abstractive summarization (Hermann et al., 2015). In addition to standard fine-tuning, PLMs can also be used in zero-shot or few-shot scenarios where supervision labels are limited, especially when provided with instructions and examples, i.e., in prompt-based learning (Brown et al., 2020).

NLU scenarios often employ bi-directional *encoder only* Transformers, e.g., BERT (Devlin et al., 2019) and its variants (e.g., Liu et al., 2019). These encoder models allow the flexibility of attentions patterns between all tokens in the texts and pretraining with denoising tasks. Fine-tuning from encoder models shows strong performances on many NLU tasks (Wang et al., 2018).

NLG scenarios fit naturally with *decoder only* Transformers, which perform autoregressive language modeling following natural language flow (Radford et al., 2019). The unidirectional attention patterns in decoder models provide a near identical formulation for pretraining and language generation tasks. Recent research also obtained more success in scaling up decoders and their strong zero/few-shot benefits with prompt-based learning (Brown et al., 2020; Rae et al., 2021; Chowdhery et al., 2022).

Pretraining and maintaining scenario specific PLMs are challenging on various fronts. It is not only tedious to duplicate similar pretraining workflows multiple times but also makes it complicated to choose proper pretrained models for each task[1]. As the scale of pretrained language models grew to hundreds of billions parameters (e.g., Brown et al., 2020; Chowdhery et al., 2022), it is nearly infeasible to pretrain multiple scenario-specific language models, as each of them come with enormous cost of computing, engineering, and energy resources (Zhang et al., 2022).

Unifying language scenarios in one pretrained model has many benefits like reduced pretraining cost, centralized resources, to name a few. Many recent research explorations combine the benefits of

[1]On Sep 2022, 60K models to choose from https://huggingface.co/docs/hub/index.

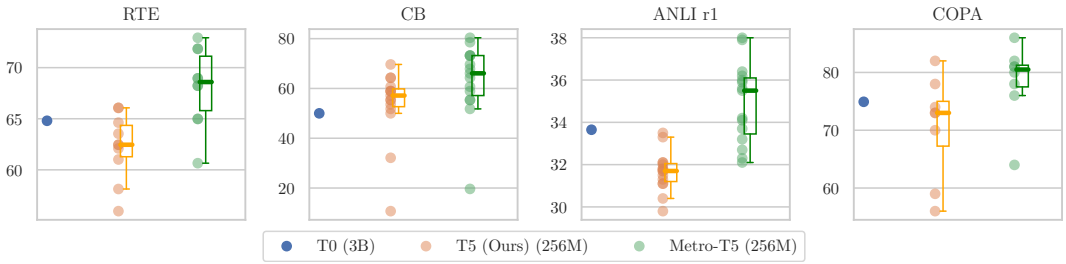


Figure 1: Prompt-based learning results of Metro-T5$_{\text{BASE++}}$ versus our T5 baseline and T0 (3B) on tasks from *T0 Eval* benchmark (Sanh et al., 2022). Metro-T5$_{\text{BASE++}}$, with only 256 million parameters, outperforms both our T5 baseline and T0$_{\text{XL}}$ with 3 billion parameters. The plots of other tasks are in Appendix A.4.

bi-directional encoders and autoregressive decoders into one Transformer (Dong et al., 2019; Lewis et al., 2019). One of the most promising unified PLMs is T5 (Raffel et al., 2019), where various tasks are reformulated in the text-to-text framework and conducted by one pretrained encoder-decoder model. However, recent research raises concerns on whether such unified models come with inferior downstream effectiveness and scaling efficiency, compared to scenario-specific encoder and decoder models (Wei et al., 2021; Bajaj et al., 2022; Wang et al., 2022).

In this paper, we strive to improve the unified language model by pretraining T5-style encoder-decoder models using ELECTRA-style model-generated signals (Clark et al., 2020). Pretraining with model generated signals provides strong efficiency and effectiveness benefits on encoder models, but our analysis shows that its direct application to T5 leads to challenges like ill-formed task, label leakage, and unstable training. To address these challenges, we propose a new pretraining model, Metro-T5, that redesigns the denoising language modeling task and multi-task learning loss, thus enables the benefits of ELECTRA-style pretraining in the unified text-to-text framework.

Our experiments on NLU (*GLUE* and *SQuAD*), NLG (*CNN/DailyMail* and *XSum*), and prompt-based learning (*T0 Eval* benchmark) demonstrate the advantage of Metro-T5. It outperforms previous unification models including BART and T5 across almost all scenarios. As a unified model, Metro-T5 achieves better or similar performance with state-of-the-art encoder and decoder only models, for example, ELECTRA, COCO-LM, and GPT-3, on their targeted scenarios. Our analysis shows that, without the new design in Metro-T5, the combination of ELECTRA and T5 underperforms vanilla T5 and sometimes results in degenerate solutions. In comparison, Metro-T5 achieves strong generalization ability with significantly better parameter efficiency. As shown in Figure 1, using only 8% parameters, Metro-T5$_{\text{base++}}$ outperforms the 3 billions parameter T0 on these zero/few-shot tasks using the same T0 prompt-based learning.

2 Related Work

Besides autoregressive and bi-directional attentions, recent research explored various other attention patterns when pretraining Transformer language models. XLNet applies the autoregressive attention pattern on the permutation of tokens (Yang et al., 2019). UniLM applies bi-directional and autoregressive attention patterns on different segments of the language sequence, as an early exploration to combine the two (Dong et al., 2019). Both BART (Lewis et al., 2019) and T5 (Raffel et al., 2019) employ the encoder-decoder architecture and study the effectiveness of different pretraining tasks.

One motivation to pretrain encoder-decoder language models is to perform natural language understanding (NLU) and natural language generation (NLG) tasks with one model. Dong et al. (2019) uses the bi-directional part of UniLM for NLU and adds in the autoregressive part for NLG. Lewis et al. (2019) feed the input sequence into both the encoder and the decoder of BART and perform NLU tasks using a classification head on the decoder. Raffel et al. (2019) proposes the text-to-text framework that converts NLU tasks into generation, by directly generating the target labels, thus unifies the format of many tasks in one framework. Zhang et al. (2020) pretrains a encoder-decoder Transformer for text summarization using a heuristic gap-sentence-generation objective.

The text-to-text framework also covers prompt-based learning scenarios, where language models are provided with additional contexts/examples (prompts) to perform zero-shot and few-shot tasks. The

encoder side takes in the prompt inputs, either manually selected tokens (hard prompts) (Gao et al., 2021), special tokens (soft prompts) (Hacohen & Weinshall, 2019), or even long demonstrations, e.g., in chain-of-thought prompts (Wei et al., 2022), and the decoder side generates the predictions for zero/few-shot tasks. Finetuning the encoder-decoder language models with prompt training data from multiple tasks also improves T5's zero/few-shot ability on other tasks (Sanh et al., 2022)

Many observed the inferior effectiveness of unified encoder-decoders compared with encoder or decoder only models. In prompt-based learning, decoders such as GPT-3 (Brown et al., 2020), PaLM (Chowdhery et al., 2022), and FLAN (Wei et al., 2021) often achieve stronger zero/few-shot accuracy. In finetuning, various techniques have been developed to improve the pretraining of bi-direction encoders, for example, model-generated pretraining signals (Clark et al., 2020), disentangled attentions (He et al., 2021), knowledge-enhanced pretraining (Sun et al., 2021), all lead to stronger empirical performances on NLU than T5. The scaling efficiency of encoder-decoder models is also a concern (Wang et al., 2022). For example, on SuperGLUE, GLaM (Du et al., 2022), a large encoder-decoder model with mixture-of-experts, only slightly outperforms the dense encoder model (Bajaj et al., 2022), although each expert in GLaM has ten times more parameters than the encoder.

The key source of effectiveness of many recent encoders is model generated pretraining signals, recently referred to as METRO (Bajaj et al., 2022). Clark et al. (2020) first invented this training mechanism in ELECTRA, which employs an auxiliary BERT to construct a corrupted text sequence for the main model to denoise. Meng et al. (2021) and Meng et al. (2022) confirmed the advantage is in the implicit learning curriculum from the auxiliary model's generated pretraining signals, which become more informative during pretraining. The benefit of METRO on encoder models also applies to multi-lingual (Chi et al., 2021) and vision tasks (Fang et al., 2022).

3 METHOD

This section first recaps preliminaries of T5 and ELECTRA-style pretraining. Then we discuss the challenges of combining them and address these challenges with Metro-T5.

3.1 PRELIMINARIES

Generally, language model pretraining can be abstracted as given the original text sequence X^{orig}, perform certain corruption to formulate a noisy input X^{noise}, and then pretrain the language model by denoising the noisy input back to the original. For example, Masked Language Modeling (MLM) randomly masks out a fraction of original tokens and pretrains a bi-directional encoder to recover the original tokens (Devlin et al., 2019).

T5 Pretraining. In T5 (Raffel et al., 2019), the noisy input is constructed by randomly deleting consecutive spans of tokens in the input, e.g., $X^{\text{noise}} = [x_1^{\text{orig}}, ..., [\texttt{MASK}]^{i:j}, ...x_n^{\text{orig}}]$, with the original tokens from position i to j replaced with a distinct special "sentinal" tokens, denoted by $[\texttt{MASK}]^{i:j}$. Then the pretraining task is to generate the deleted tokens using the encoder-decoder Transformer:

$$[x_1^{\text{orig}}, ..., [\texttt{MASK}]^{i:j}, ...x_n^{\text{orig}}] \xrightarrow{\text{Encoder}} \boldsymbol{H}^{\text{Enc}} \xrightarrow{\text{Decoder}} [[\texttt{MASK}]^{i:j}, x_i^{\text{orig}}, ..., x_j^{\text{orig}}]. \tag{1}$$

It feeds X^{noise} as input to the encoder of T5, and then asks the decoder to generate the deleted original tokens after the corresponding sentinel token. The training uses standard teacher-forcing with the decoder provided with correct tokens in all previous positions.

Text-to-Text Framework. The pretrained encoder-decoder language model is then applied to NLP tasks in a unified text-to-text framework. T5 casts every task—including classification, question answering, and summarization—into a text-to-text format, where the encoder is fed with the text input and the decoder is then asked to generate the target prediction.

ELECTRA-Style Pretraining. The key idea of ELECTRA-style pretraining is the construct the noisy input X^{noise} using another auxiliary language model, i.e., by sampling tokens in the masked-out positions, using the language modeling probability of the auxiliary model. Usually, the auxiliary is an masked language model (Clark et al., 2020; Meng et al., 2021),

$$x_i^{\text{noise}} \sim p_{\text{MLM}}(x|\boldsymbol{h}_i^{\text{Aux}}), \text{if } i \in \mathcal{M}; x_i^{\text{noise}} = x_i^{\text{orig}}, \text{otherwise}, \tag{2}$$

with $\mathcal{M}$ is a set of masked-out positions, often 15% of the entire sequence.

Table 1: Examples of encoder inputs and decoder targets of different ways to configure the denoising task. `[M]` denotes a shared mask token. The auxiliary MLM model predicts one token for each `[M]`. Grayed-out tokens are part of the target fed into the decoder but not included in pretraining loss.

Original Sentence		Thank you for inviting me to your party last week
Auxiliary Model	*Input*	Thank you `[M]` `[M]` me to your party `[M]` week
	Output	for giving apple
Main Model	*Input*	Thank you for giving me to your party apple week
Decoding Target	*Masked Tokens Only*	for inviting last
	All Tokens	Thank you for inviting me to your party last week
	All Tokens, Masked Loss	Thank you for inviting me to your party last week

Then the main Transformer encoder is pretrained to denoise the model-generated noisy input X_i^{noise}. Clark et al. (2020) originally uses the replaced token detection task (RTD), a simple binary classification on whether each token is replaced by the auxiliary model or from the original:

$$X^{\text{noise}} \xrightarrow{\text{Encoder}} \boldsymbol{H}^{\text{Enc}} \xrightarrow{\text{RTD Head}} \boldsymbol{Y}; y_i = \mathbb{1}(x_i^{\text{noise}} \neq x_i^{\text{orig}}). \tag{3}$$

The binary classification objective can effectively pretrains the main model for NLU tasks. However, the pretrained model has limited language modeling capability which is necessary for NLG and many prompt-based learning tasks. Later, Meng et al. (2021) introduces the corrective language modeling task (CLM) that re-enabled the language modeling capacity upon the main encoder:

$$X^{\text{noise}} \xrightarrow{\text{Encoder}} \boldsymbol{H}^{\text{Enc}} \xrightarrow{\text{CLM Head}} X^{\text{orig}}. \tag{4}$$

The CLM task often trains together with the RTD task by multi-task learning.

ELECTRA-style models achieved strong empirical performance on a wide range of NLU tasks. Recent studies revealed their main source of effectiveness resides in the model-generated pretraining signals. Pretrained side-by-side with the main Transformer, the auxiliary model also gets stronger and stronger, thus generating fewer but more confusing noises in Eq. (2). This model-generated denoising training objective, recently referred to as METRO (Bajaj et al., 2022), significantly improves the efficiency and generalization ability of encoder language model pretraining, as observed in variant model scales (Bajaj et al., 2022) and application scenarios (Chi et al., 2021; Fang et al., 2022).

3.2 Challenges of Pretraining T5 in ELECTRA-Style

A natural step to enhance T5 is to also pretrain the encoder-decoder with model-generated noises as ELECTRA-style. A straightforward way to combine the two is to employ the auxiliary language model to generate replacement tokens (Eq. (2)) in the masked out spans (Eq. (1)):

$$[x_1^{\text{orig}}, ..., [\texttt{MASK}]^{i:j}, ...x_n^{\text{orig}}] \xrightarrow{\text{Auxiliary LM}} [x_1^{\text{orig}}, ..., x_i^{\text{noise}}, ..., x_j^{\text{noise}}, ..., x_n^{\text{orig}}]; \tag{5}$$

and then pretrain the encoder-decoder Transformer to denoise the corrupted input, for example, in the RTD+CLM multi-task setup:

$$[x_1^{\text{orig}}, ..., x_i^{\text{noise}}, ..., x_j^{\text{noise}}, ..., x_n^{\text{orig}}] \xrightarrow{\text{encoder}} \boldsymbol{H}^{\text{enc}} \xrightarrow{\text{decoder}} \boldsymbol{H}^{\text{dec}}; \tag{6}$$

$$[\boldsymbol{h}_1^{\text{dec}}, ..., \boldsymbol{h}_{1+i-j}^{\text{dec}}] \xrightarrow{\text{CLM Head}} [x_i^{\text{orig}}, ..., x_j^{\text{orig}}]; \tag{7}$$

$$[\boldsymbol{h}_1^{\text{dec}}, ..., \boldsymbol{h}_{1+i-j}^{\text{dec}}] \xrightarrow{\text{RTD Head}} [y_i, ..., y_j]; y_k = \mathbb{1}(x_k^{\text{noise}} \neq x_k^{\text{orig}}). \tag{8}$$

However, our preliminary experiments show that this straightforward design yields unstable and diverging pretraining runs. Many of the challenges arise from the difference between the unique formulation of encoder-decoder language models, as we discuss next.

Ill-formed Target. In Eq. (7), the denoising is performed by the decoder, which has to learn to select which positions to denoise first. This is much more challenging than using the encoder who has the explicit position correlation information. In this formulation, where the target consists of *masked tokens only*, it is impossible for the decoder to distinguish between the original tokens (e.g., x_1^{orig}) or those correctly predicted by the auxiliary model in the masked positions $\mathcal{M}$.

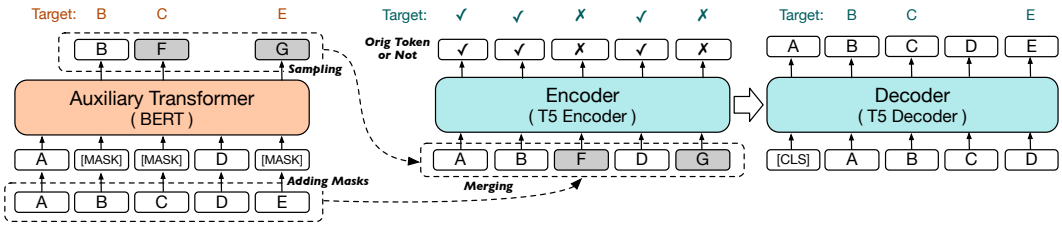


Figure 2: The architecture of Metro-T5 using BERT as auxiliary and T5 as the main model.

Table 1 shows an example of such ill-formed target in the (masked token only) setting. The masked tokens are "for", "inviting", and "last", but it is impossible for the model to know that it should decode the token "for", as it is a correct token, the same as the unmasked ones (e.g. "Thank" and "you"). This ill-formed target is impossible to learn for the pretraining model.

Label Leakage. Consecutive span masking in the auxiliary model in Eq. (6) leads to label leakage. At the k-th position during teacher-forced training, the decoder has access to the ground truth token at $k-1$. It can compare the ground truth with the input at position $k-1$. If the two disagree, it is likely the following positions are also masked out, a shortcut for the model to exploit.

Unstable Training. When pretraining encoder only models (Meng et al., 2021; Bajaj et al., 2022), the RTD task provides an easier target for the main model and improve the optimization stability. However, decoding RTD target is complicated. In Eq. (8), the encoder-decoder model needs connect the information from the encoder position i to decode y_i. This is more complicated than RTD in encoder models as it requires a much longer attention path and defeats the purpose of RTD in providing a trivial task to stabilize optimization.

3.3 METRO-T5

In the rest of this section, we first present the overview of Metro-T5, then its specific designs that addresses the problems found in previous analysis. We also discuss its updated Transformer architecture based on findings in recent research.

Overview. As illustrated in Figure 2, Metro-T5 also uses a BERT-style MLM encoder as the auxiliary model and a T5-style encoder-decoder as the main model. The overall pretraining setup is:

$$X^{\text{orig}} \xrightarrow{\text{Random Mask}} [x_1^{\text{orig}}, \ldots, \texttt{[MASK]}, \ldots, \texttt{[MASK]}, \ldots, x_n^{\text{orig}}] \xrightarrow{\text{Auxiliary}} X^{\text{noise}}; \tag{9}$$

$$X^{\text{noise}} \xrightarrow{\text{Encoder}} \boldsymbol{H}^{\text{enc}} \xrightarrow{\text{RTD Head}} \mathbb{1}(x_i^{\text{orig}} = x_i^{\text{noise}}); \tag{10}$$

$$X^{\text{noise}} \xrightarrow{\text{Encoder}} \boldsymbol{H}^{\text{enc}} \xrightarrow{\text{Decoder}} \boldsymbol{H}^{\text{dec}} \xrightarrow{\text{CLM Head}} X^{\text{orig}}. \tag{11}$$

In each of the three equations Metro-T5 introduces a new design to enable effective pretraining of the encoder-decoder main model using signals generated by the auxiliary BERT encoder: *Random Masks on Auxiliary* (Eq. (9)), *RTD on Encoder* (Eq. (10)), and *Decoding Entire Origin* (Eq. (11)).

Random Masks on Auxiliary. To avoid the label leakage problem, Metro-T5 rolls back to the vanilla random mask at individual token level in Eq. (9). Specifically, with the 15% randomly sampled masked positions $\mathcal{M}$, the masked LM loss for the auxiliary is

$$\mathcal{L}_{\text{MLM}} = -\mathbb{E}_{i \in \mathcal{M}} \log p_{\text{MLM}}(x_i^{\text{orig}} | \mathbf{h}_i^{\text{aux}}). \tag{12}$$

RTD on Encoder. As we discussed in Section 3.2, decoding binary RTD labels with the decoder is difficult for an encoder-decoder model. Thus Metro-T5 moves the RTD task to the encoder side:

$$\mathcal{L}_{\text{RTD}} = -\mathbb{E} \log p_{\text{RTD}}(\mathbb{1}(x_i^{\text{orig}} = x_i^{\text{noise}}) | \mathbf{h}_i^{\text{enc}}). \tag{13}$$

Decoding Entire Origin. As decoding only the masked out position leads to ill-formed task, Metro-T5 changes the decoding target to the entire original sequence X^{orig}. It trains the encoder-decoder model using the following corrective language modeling loss:

$$\mathcal{L}_{\text{CLM}} = -\mathbb{E}_{i \in \mathcal{M}} \log p_{\text{LM}}(x_i^{\text{orig}} | \mathbf{h}_i^{\text{dec}}). \tag{14}$$

We use two different designs compared to previous research. The first one is that we eliminated the copy mechanism used on previous ELECTRA-style models (Meng et al., 2021), which provides a short path for the model to direct copy tokens from its input. The second is that we only apply the loss on masked positions $\mathcal{M}$, instead of on all positions on the decoder side. This also avoids training on the unmasked tokens which are also trivial copy-and-pastes. Both designs aim to encourage the model to focus on learning deep language semantics, to obtain generalization ability more efficiently, rather than learning to copy from its inputs. Table 1 shows examples of different configurations of decoding target, including "All Tokens, Masked Loss" used in Metro-T5.

With these designs, the final pretraining objective of Metro-T5 is:

$$\mathcal{L} = \mathcal{L}_{\text{MLM}} + \lambda_{\text{RTD}}\mathcal{L}_{\text{RTD}} + \lambda_{\text{CLM}}\mathcal{L}_{\text{CLM}}, \tag{15}$$

where λ_{RTD} and λ_{CLM} are hyperparameters. Similar to ELECTRA, the auxiliary model and the main model are trained side-by-side. Only the main model, the T5-style encoder-decoder, is used for downstream applications, i.e., in the same text-to-text framework as vanilla T5.

Architectural Upgrades over T5. We also incorporate model architecture changes that are proved to be beneficial in earlier works. As recent research shows only relative position embeddings are not as effective (Luo et al., 2022), we use absolute positional embeddings combined with relative position embedding (Meng et al., 2021). We use Post-LayerNorm instead of T5's Pre-LayerNorm which usually leads to better performance on downstream tasks (Bajaj et al., 2022).

4 Experimental Setup

This section layouts main experiment configurations. More details can be found in Appendix A.2.

Model Architecture. The main model of Metro-T5 is similar to T5-Base (Raffel et al., 2019). Specifically, both the encoder and decoder consist of 12 layers and a hidden dimension of 768. The auxiliary model of Metro-T5 is a 4-layer Transformer Encoder with the same hidden size. We follow Clark et al. (2020) and share token embeddings between the main model and the auxiliary model.

Pretraining. We consider two standard pretraining settings: *base* and *base++*. *Base* (Devlin et al., 2019) is to pretrain on English Wikipedia and BookCorpus (16GB of texts) for 131 billion tokens (512 tokens per sequence, 2,048 sequences per batch, and 125k steps). *Base++* is the training configuration first used in RoBERTa (Liu et al., 2019): pretraining on a mixed corpus of 160GB texts, which consists of English Wikipedia, BookCorpus, OpenWebText (Gokaslan & Cohen, 2019), CC-News (Liu et al., 2019), and STORIES (Trinh & Le, 2018), for 2.1 trillion tokens (512 tokens per sequence, 2,048 sequences per batch, and 2M steps).

Evaluation. We evaluate pretrained models on five benchmarks: *GLUE* (Wang et al., 2018) for natural language understanding, *SQuAD v1.1* (Rajpurkar et al., 2016) for question-answering, *CNN/DailyMail v3.0.0* (Hermann et al., 2015) and *XSum* (Narayan et al., 2018) for sequence-to-sequence natural language generation, and *T0 Eval* (Sanh et al., 2022) for prompt-finetuning and zero-shot generalization. We follow standard practices (Liu et al., 2019; Meng et al., 2021) and perform hyperparameter searches on GLUE, SQuAD, CNN/DM and XSum. We also follow Sanh et al. (2022) and perform prompt-finetuning on three multitask mixtures, *T0 Train*, *T0+ Train*, and *T0++ Train*, and evaluate each on the *T0 Eval* benchmark.

Baselines. The main baseline is our own T5 run, which uses the same Transformer architecture and training data, except METRO style pretraining. Please see Appendix A.3 for implementation details. We also compare with the reported numbers of other encoder-decoder models that unify multiple scenarios, mainly BART (Lewis et al., 2019) and the original T5 (Raffel et al., 2019). For reference we include the state-of-the-arts results from many scenario specific models, if applicable, for example encoder-only ones on NLU, decoder-only ones on NLG, and also models at larger scales when informative.

5 Evaluation

In this section we evaluate the overall accuracy of Metro-T5 as a unified model, its ablation variants, the effectiveness of its new designs to enable ELECTRA-Style training with T5, and the benefits of pretraining efficiency from METRO on encoder-decoders.

Table 2: Results on the development sets of GLUE, SQuAD v1.1, and XSum. All results are single-task, single-model fine-tuning. The GLUE score is the average score of the eight tasks on GLUE. Results not available in public reports are marked as "-". The question answering task (SQuAD) is formulated as an NLG task for encoder-decoder models and NLU for encoder-only models.

	NLU									NLG		
Task	MNLI	QQP	QNLI	SST-2	CoLA	RTE	MRPC	STS-B	**GLUE**	**SQuAD**	**CNN/DM**	**XSum**
Metric	ACC	ACC	ACC	ACC	MCC	ACC	ACC	PCC	AVG	F1	R-2-F	R-2-F
Base Setting: BERT Base Size, Wikipedia + Book Corpus (16GB)												
BERT (Devlin et al., 2019)	84.5/-	91.3	91.7	93.2	58.9	68.6	87.3	89.5	83.1	88.5	-	-
RoBERTa (Liu et al., 2019)	85.8/85.5	91.3	92.0	93.7	60.1	68.2	87.3	88.5	83.3	90.4	-	-
ELECTRA (Clark et al., 2020)	86.9/86.7	91.9	92.6	93.6	66.2	75.1	88.2	89.7	85.5	90.8	-	-
COCO-LM (Meng et al., 2021)	88.5/88.3	92.0	93.1	93.2	63.9	84.8	91.4	90.3	87.2	-	-	-
CLM Only	88.6/88.4	92.0	93.2	93.7	67.4	80.1	90.0	90.4	86.9	-	-	
BART (Lewis et al., 2019)	83.8/-	-	-	-	-	-	-	-	-	90.8	-	-
PEGASUS (Zhang et al., 2020)	-	-	-	-	-	-	-	-	-	-	18.8	16.6
T5 (WikiBook) (Raffel et al., 2019)	84.4/83.5	90.9	91.9	92.8	55.5	76.9	90.8	86.5	83.7	89.7	19.3	-
T5 (Ours)	87.1/87.1	91.7	93.1	93.8	62.3	78.0	88.7	88.2	85.4	90.5	**20.7**	18.5
Metro-T5	**88.4/87.9**	**92.1**	**93.2**	**94.4**	**68.8**	**82.7**	**90.4**	**90.0**	**87.5**	**90.8**	**20.7**	**18.9**
Base++ Setting: BERT Base Size, Bigger Training Data, and/or More Training Steps												
RoBERTa (Liu et al., 2019)	87.6/-	91.9	92.8	94.8	63.6	78.7	90.2	91.2	86.4	94.6	-	-
COCO-LM (Meng et al., 2021)	90.2/90.0	92.2	94.2	94.6	67.3	87.4	91.2	91.8	88.6	-	-	-
UniLM v2 (Bao et al., 2020)	88.5/-	91.7	93.5	95.1	65.2	81.3	91.8	91.0	87.3	93.1	20.4	21.1
T5 (C4) (Raffel et al., 2019)	84.2/84.6	91.6	90.5	92.7	53.8	76.3	88.9	88.0	83.3	88.8	20.3	-
T5 (Ours)	89.7/89.2	91.9	94.1	95.7	61.6	83.4	90.2	**90.8**	87.1	92.2	**21.4**	20.9
Metro-T5	**90.0/90.0**	**92.0**	**94.3**	**96.0**	**70.7**	**86.3**	**90.7**	**90.8**	**88.8**	**92.4**	21.3	**21.2**

Table 3: Prompt learning results on the *T0 Eval* dataset. "Wino.", "SC.", and "HS" refer to Winogrande, StoryCloze, and hellaSwag. All reported datasets use accuracy as their metric. *Italic* results are produced under the supervised setting. Others are under the zero-shot setting.

Model	**Params**	**NLI**			**Coref.**		**Compl.**			**WSD**	
		RTE	**CB**	**ANLI r1/r2/r3**	**WSC**	**Wino.**	**COPA**	**SC.**	**HS.**	**WiC**	**AVG**
Pretraining only											
GPT-3_{SMALL} (Brown et al., 2020)	125M	47.70	0.00	33.40/33.20/33.60	59.60	52.00	66.00	63.30	33.70	0.00	38.41
GPT-3_{MED} (Brown et al., 2020)	350M	49.80	32.10	34.20/31.90/34.00	56.70	52.10	68.00	68.50	43.60	0.00	42.81
T5+LM (Lester et al., 2021)	11B	53.03	34.34	32.89/33.76/33.82	54.09	50.65	54.88	27.00	48.16	50.30	42.99
Prompt Finetune on *T0 Train*											
T5_{BASE} (Ours)	226M	62.85	45.30	30.82/32.37/32.14	**62.16**	50.77	70.63	**81.03**	24.86	**50.78**	49.43
Metro-T5_{BASE}	226M	**65.18**	**45.60**	**31.64/32.98/33.81**	55.77	**51.07**	**70.81**	80.97	**25.28**	50.69	**49.44**
$\text{T5}_{\text{BASE++}}$ (Ours)	256M	62.24	53.45	31.68/32.94/34.88	**61.73**	51.65	70.63	87.62	25.88	**51.21**	51.26
Metro-$\text{T5}_{\text{BASE++}}$	256M	**68.16**	**63.21**	**34.92/33.81/36.82**	60.48	**52.03**	**78.50**	**89.23**	**27.68**	50.88	**54.15**
T0_{XL} (Sanh et al., 2022)	3B	64.55	45.36	33.84/33.11/33.33	65.10	50.97	72.40	84.03	27.29	50.69	50.97
Prompt Finetune on *T0+ Train*											
T5_{BASE} (Ours)	226M	63.57	**48.93**	31.76/32.92/33.02	**60.96**	**51.93**	**72.38**	*81.71*	*40.11*	**51.32**	51.69
Metro-T5_{BASE}	226M	**70.56**	47.08	**33.05/34.53/34.37**	57.98	51.75	69.13	***83.08***	***49.00***	50.78	**52.85**
$\text{T5}_{\text{BASE++}}$ (Ours)	256M	68.30	60.24	33.77/34.31/35.00	60.96	51.59	70.00	*89.29*	*56.10*	51.39	55.54
Metro-$\text{T5}_{\text{BASE++}}$	256M	**71.44**	**60.71**	**36.91/35.24/36.46**	**62.21**	**54.08**	**78.88**	***90.29***	***67.57***	**51.60**	**58.67**
Prompt Finetune on *T0++ Train*											
T5_{BASE} (Ours)	226M	69.06	48.39	31.90/33.61/33.94	*55.72*	*51.15*	***76.06***	*82.55*	*39.62*	*63.18*	53.20
Metro-T5_{BASE}	226M	**72.04**	**58.63**	**33.85/35.29/36.57**	***56.11***	***52.15***	*74.06*	***83.65***	***48.66***	***64.29***	**55.94**
$\text{T5}_{\text{BASE++}}$ (Ours)	256M	**77.87**	63.10	36.15/34.61/38.18	*56.44*	*51.78*	*75.38*	*89.33*	*55.95*	*65.53*	58.57
Metro-$\text{T5}_{\text{BASE++}}$	256M	77.80	**69.52**	**39.69/36.61/40.08**	***61.44***	***54.55***	***83.88***	***90.88***	***68.54***	***67.59***	**62.78**

5.1 Overall Results

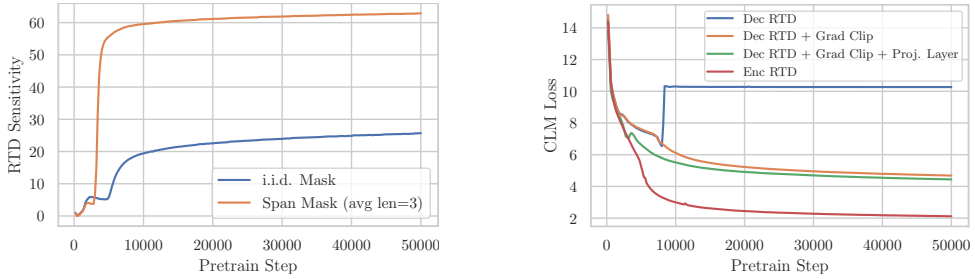
We present the overall results in two groups, finetuning and prompt-based learning.

Finetuning results are listed in Table 2. Metro-T5 outperforms our own T5, which is much stronger than the T5 checkpoints released by Google, on nearly all tasks in all settings. On NLU tasks, it achieves comparable or slightly better performances than the strong ELECTRA-style encoder, COCO-LM (Meng et al., 2021), especially its CLM Only version which does not use additional contrastive pretraining. In comparison, previous encoder-decoder models, BART and T5, are quite behind on GLUE. Metro-T5 provides the most robust effectiveness among these unification models across all finetuning scenarios.

Prompt-based Learning results are shown in Table 3. Metro-T5 outperforms T5 (Ours) under all four settings. With better prompt-based learning results, i.e., from T0 to T0+ and then to T0++, the

Table 4: Performance of Metro-T5 variations on finetuning tasks. All ablations are done in the *base* pretraining setting using exactly the same finetuning pipeline.

Task	MNLI	QQP	QNLI	SST-2	CoLA	RTE	MRPC	STS-B	GLUE	SQuAD	CNN/DM	XSum
Metro-T5	88.4/87.9	92.1	93.2	94.4	68.8	82.7	90.4	90.0	**87.5**	**90.8**	**20.7**	**18.9**
w. RTD on Decoder	86.1/85.8	91.5	92.1	92.5	63.6	76.5	87.5	33.0	77.8	88.1	19.7	16.6
+ Projection Layer on CLM	87.6/86.9	91.8	92.7	92.9	63.1	77.3	88.7	88.9	85.3	89.2	20.4	17.4
w. Continuous Span Mask	87.3/86.7	92.0	92.4	93.5	69.2	81.6	89.7	89.7	86.9	90.5	20.5	18.0
w. CLM Loss on All Position	88.1/87.6	92.0	92.7	93.3	68.4	83.4	90.7	89.7	87.3	90.5	20.5	17.7
w. CLM with Copy Mechanism	88.5/88.4	92.0	93.0	93.9	67.8	82.3	90.2	90.1	87.2	90.2	20.5	17.8
T5 (Ours)	87.1/87.1	91.7	93.1	93.8	62.3	78.0	88.7	88.2	**85.4**	**90.5**	**20.7**	18.5
w. All-token LM loss	85.8/85.6	91.4	92.3	92.8	57.5	77.6	88.2	88.3	84.2	90.2	20.5	**18.7**



(a) Random Mask versus Span Mask

(b) Encoder RTD versus Decoder RTD

Figure 3: Pretraining behaviors of different designs to pretrain T5 in ELECTRA-style.

empirical advantage of Metro-T5 becomes more significant. Notably, in the T0 setting, Metro-T5$_{\text{base++}}$ outperforms T0$_{\text{XL}}$ which starts from the 3 billion parameter T5, 30 times more than Metro-T5 and pretrained on much larger C4 corpus. This demonstrate that Metro-T5 achives generalization ability with significant better parameter and pretraining efficiency.

5.2 Ablation Studies

This experiment studies the influences of different design choices in Metro-T5. We use the finetuning scenarios which are more stable for this study. The results are shown in Table 4.

Using RTD on decoder leads to much worse performance on both NLU and NLG tasks. The RTD task on the decoder side is complex and may confuse the decoder in the multi-task setup. Using a projection layer on the CLM head on the decoder side mitigate some of its problem but does not resolve it. Using continuous span mask instead of random masks also leads to worse performance on nearly all tasks. In next experiment we further study the challenges introduce by these designs.

Enabling the copy mechanism on the CLM task reduces model performance. It introduces too many trivial copy-and-paste task that hinders the learning of actual language semantics. The same applies to using the language modeling loss on positions, either CLM for Metro-T5 or standard language model on T5 (Ours). Both hurts model's generalization ability with too many trivial copying tasks.

5.3 Pretraining Behaviors

In this experiment, we show how Metro-T5 addresses the challenges of combing T5 with ELECTRA-style training. We mentioned these challenges in Section 3.2. We omit the *"ill-formed task"* challenge where the model just does not pretrain, and Metro-T5 has to address it to provide any meaningful results.

Avoid Label Leakage. Figure 3a shows the RTD sensitivity (true positive rate) if Metro-T5 when using random masking on the auxiliary model versus using T5's span masking. As discussed in Section 3.2, span masking has label leakage, as a token after a replaced token is likely to be noise. This leads to trivial solutions on many masked positions, as shown in the more than doubled pretraining RTD accuracy on masked positions with *Span Mask*. As expected, this label leakage hurts the model generalization ability as is shown in Table 4.

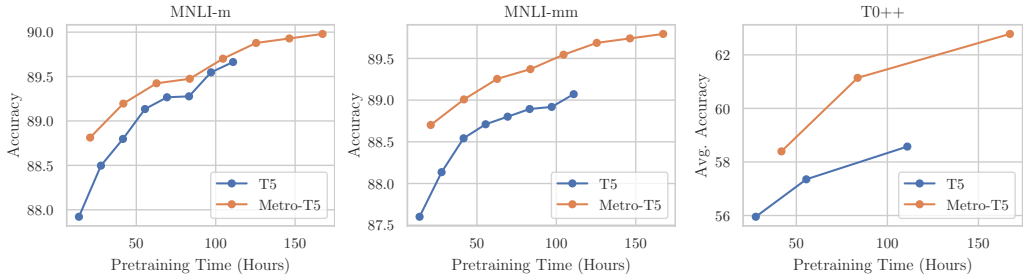


Figure 4: Downstream performance with different pretraining GPU hours of $T5_{base++}$ and Metro-$T5_{base++}$, measured on the exact training pipeline and computing environments.

Remove Training Instability. Figure 3b shows the loss on the CLM task in pretraining with the RTD objective applied on the encoder side (Metro-T5) versus on the decoder side. The pretraining diverged with RTD on the decoder. Adding strong gradient clipping and an additional projection layer for the CLM task mitigates the problem. The model converges, but with much higher pretraining loss and thus worse generalization ability in downstream tasks, as is shown in Table 4.

5.4 Pretraining Efficiency

In this section, we study the pretraining efficiency of Metro-T5 in comparison to vanilla T5 from the aspect of pretraining computational costs. Note that Metro-T5 uses an auxiliary model that introduces additional computations per training step. Also, its "decoding entire original sentence" target is also longer than T5's target for each text sequence, resulting in a more computationally expensive decoder during pretraining.

As a fair comparison between Metro-T5 and the T5 baseline regarding their compute-efficiency, Figure 4 plots the downstream performance of Metro-$T5_{base++}$ and $T5_{base++}$ at different *pretraining time* on the exact sample computing environments. Specifically, we evaluate their intermediate checkpoints on MNLI-m, MNLI-mm, and T0++, and record their pretraining wall time, which is the exact reflection of their pretraining computation cost. The results show that the efficiency benefits of METRO pretraining overwhelm the increased computations *per step*. Metro-T5 achieves better generalization ability than T5 at *every point* we measured on all three tasks. The efficient generalization ability is also more observed in the prompt-based learning tasks including zero-shot evaluations.

6 Conclusion and Future Work

In this paper we present Metro-T5 which brings in the advantage of model-generated pretraining signals to encoder-decoder language models, with the goal of improving the performance of unified pretraining model on multiple language scenarios. Metro-T5 redesigns the pretraining task and multi-task learning in T5 and ELECTRA-style models to address the challenges we observed in combing them. Our experiments demonstrate the advantage of Metro-T5 on both finetuning and prompt-based learning scenarios. Our analysis shows that Metro-T5 achieved strong generalization ability with efficiency, both on the network size aspect, outperforming $T0_{XL}$ on T0 with only 8% of its parameters, and the pretraining computing cost aspect, reaching better downstream performance with fewer GPU hours than T5.

One future work direction is the scale up Metro-T5 to bigger Transformer networks and more pretraining steps, to explore whether the benefits observed in the common research setups can adapt to the large scale setting. It will be interesting to see whether certain emerging abilities observed on other multi-billion parameters models will also show up in scaled up Metro-T5, earlier or later. The unification of not only the pretraining step, but also the downstream application step, is another interesting future research direction with prompt-based learning and scaled up models.

REPRODUCIBILITY STATEMENT

We plan to release our code, including pretraining pipeline, and model checkpoints if this work is accepted. We will add an internal link to the anonymous repository that is visible to reviewers and ACs. Besides open-source, we follow the standard experimental settings, evaluate with standard benchmarks, and use common model configurations in all our studies. We also include details of our implementations in Section 4 and Appendix.

REFERENCES

Payal Bajaj, Chenyan Xiong, Guolin Ke, Xiaodong Liu, Di He, Saurabh Tiwary, Tie-Yan Liu, Paul Bennett, Xia Song, and Jianfeng Gao. Metro: Efficient denoising pretraining of large scale autoencoding language models with model generated signals. *arXiv preprint arXiv:2204.06644*, 2022.

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.

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

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

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *International Workshop on Semantic Evaluation*, 2017.

Zewen Chi, Shaohan Huang, Li Dong, Shuming Ma, Saksham Singhal, Payal Bajaj, Xia Song, and Furu Wei. Xlm-e: Cross-lingual language model pre-training via electra. *arXiv preprint arXiv:2106.16138*, 2021.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, 2005.

Marie-Catherine de Marneffe, Mandy Simons, and Judith Tonhauser. The CommitmentBank: Investigating projection in naturally occurring discourse. 2019. To appear in *Proceedings of Sinn und Bedeutung 23*. Data can be found at `https://github.com/mcdm/CommitmentBank/`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*, pp. 4171–4186, 2019. URL `https://aclanthology.org/N19-1423`.

William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *International Workshop on Paraphrasing (IWP)*, 2005.

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 *NeurIPS*, 2019.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pp. 5547–5569. PMLR, 2022.

Yuxin Fang, Li Dong, Hangbo Bao, Xinggang Wang, and Furu Wei. Corrupted image modeling for self-supervised visual pre-training. *arXiv preprint arXiv:2202.03382*, 2022.

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

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third PASCAL recognizing textual entailment challenge. In *ACL-PASCAL workshop on textual entailment and paraphrasing*, 2007.

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

Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. In *ICML*, 2019.

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

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. DeBERTa: Decoding-enhanced bert with disentangled attention. In *ICLR*, 2021.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pp. 1693–1701, Cambridge, MA, USA, 2015. MIT Press.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.243. URL `https://aclanthology.org/2021.emnlp-main.243`.

Hector Levesque, Ernest Davis, and Leora Morgenstern. The Winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2012. URL `http://dl.acm.org/citation.cfm?id=3031843.3031909`.

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. In *ACL*, 2019.

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.

Shengjie Luo, Shanda Li, Shuxin Zheng, Tie-Yan Liu, Liwei Wang, and Di He. Your transformer may not be as powerful as you expect. *arXiv preprint arXiv:2205.13401*, 2022.

Yu Meng, Chenyan Xiong, Payal Bajaj, Saurabh Tiwary, Paul Bennett, Jiawei Han, and Xia Song. COCO-LM: Correcting and contrasting text sequences for language model pretraining. In *Proceedings of NeurIPS 2021*, 2021. URL `https://proceedings.neurips.cc/paper/2021/hash/c2c2a04512b35d13102459f8784f1a2d-Abstract.html`.

Yu Meng, Chenyan Xiong, Payal Bajaj, Saurabh Tiwary, Paul Bennett, Jiawei Han, and Xia Song. Pretraining text encoders with adversarial mixture of training signal generators. *arXiv preprint arXiv:2204.03243*, 2022.

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James Allen. Lsdsem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pp. 46–51, 2017.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. Aug 2018. doi: 10.48550/arXiv.1808.08745. URL https://arxiv.org/abs/1808.08745v1.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020.

Mohammad Taher Pilehvar and Jose Camacho-Collados. WiC: The word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. Association for Computational Linguistics, 2019. URL https://arxiv.org/abs/1808.09121.

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.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

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. *Journal of Machine Learning Research*, 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*, 2016.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*, 2011.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, aug 2021. ISSN 0001-0782. doi: 10.1145/3474381. URL https://doi.org/10.1145/3474381.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=9Vrb9D0WI4.

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

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 *EMNLP*, 2013.

Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, et al. Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *arXiv preprint arXiv:2107.02137*, 2021.

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

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *EMNLP Workshop BlackboxNLP*, 2018.

Thomas Wang, Adam Roberts, Daniel Hesslow, Teven Le Scao, Hyung Won Chung, Iz Beltagy, Julien Launay, and Colin Raffel. What language model architecture and pretraining objective work best for zero-shot generalization? *arXiv preprint arXiv:2204.05832*, 2022.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. In *TACL*, 2019.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*, 2018.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. XLNet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, 2019.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11328–11339. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/zhang20ae.html`.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

A APPENDIX

A.1 DETAILS OF EVALUATION BENCHMARKS

We show the statistics of all the datasets in Table 5.

Table 5: The overview of GLUE, SQuAD v1.1, CNN/DM 3.0.0, XSum and T0. We list their validation data size, language tasks, evaluation metrics, and domain of corpus.

		Size	Task	Metric(s)	Domain
GLUE	MNLI	19,647	Natural language inference	Accuracy	Misc.
	QQP	40,430	Sentence similarity	Accuracy	Social QA
	QNLI	5,463	Natural language inference	Accuracy	Wikipedia
	SST-2	872	Sentiment analysis	Accuracy	Movie Reviews
	CoLA	1,043	Sentence acceptability judgement	Matthews Corr.	Misc.
	RTE	277	Natural language inference	Accuracy	Misc.
	MRPC	408	Paraphrasing	Accuracy	News
	STS-B	1,500	Sentence similarity	Pearson Corr.	Misc.
SQuAD v1.1		10,570	Question answering	F1	Wikipedia
CNN/DM v3.0.0		13,368	Summarization	Rouge-2-F	News Articles
XSum		11,332	Summarization	Rouge-2-F	News Articles
T0-Eval	RTE	277	Natural language inference	Accuracy	
	CB	56	Natural language inference	Accuracy	
	ANLI	3,200	Natural language inference	Accuracy	
	WSC	104	Coreference resolution	Accuracy	
	Winogrande XL	1,267	Coreference resolution	Accuracy	
	COPA	100	Sentence completion	Accuracy	
	StoryCloze 2016	1,871	Sentence completion	Accuracy	
	HellaSwag	10,042	Sentence completion	Accuracy	
	WiC	638	Word Sense Disambiguation	Accuracy	

A.1.1 GLUE BENCHMARK

The details of the tasks in the GLUE benchmark are as follow:

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 (Bentivogli et al., 2009; Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007) 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.

Table 6: Our pretraining Hyperparameters for Metro-T5.

Parameters	**Base**	**Base++**
Max Steps	125K	2M
Peak Learning Rate	4e-4	2e-4
Batch Size	2,048	2,048
Warm-Up Steps	10,000	10,000
Total Steps	125,000	2,000,000
Sequence Length	512	512
Relative Position Encoding Buckets	32	32
Relative Position Encoding Max Distance	128	128
Loss multipliers (λ_{MLM}, λ_{RTD}, λ_{CLM})	(1, 50, 1)	(1, 50, 1)
Adam ϵ	1e-6	1e-6
Adam (β_1, β_2)	(0.9, 0.98)	(0.9, 0.98)
Clip Norm	-	2.0
Dropout	0.1	0.1
Weight Decay	0.01	0.01

A.1.2 T0-Eval Benchmark

We also provide the details of all the tasks in T0-Eval:

RTE: Recognizing Textual Entailment (Bentivogli et al., 2009; Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007) comes from a series of annual competitions on textual entailment. All datasets are combined and converted to two-class classification: *entailment* and *not_entailment*.

CB: CommitmentBank (de Marneffe et al., 2019) contains corpus of short texts in which at least one sentence contains an embedded clause. The resulting task is framed as three-class textual entailment on examples that are drawn from the Wall Street Journal, fiction from the British National Corpus, and Switchboard.

ANLI: Adversarial Natural Language Inference (Nie et al., 2020) is a new large-scale NLI benchmark dataset, collected via an iterative, adversarial human-and-model-in-the-loop procedure.

WSC: Winograd Schema Challenge (Levesque et al., 2012) is a coreference resolution dataset in which examples consist of a sentence with a pronoun and a list of noun phrases from the sentence. The system must determine the correct referrent of the pronoun from among the provided choices. Winograd schemas are designed to require everyday knowledge and commonsense reasoning to solve.

Winogrande (Wino.): WinoGrande (Sakaguchi et al., 2021) is a large-scale coreference resolution dataset of 44k problems. It is inspired by the original WSC design, but adjusted to improve both the scale and the hardness of the dataset.

COPA: (Roemmele et al., 2011) Choice of Plausible Alternatives is a causal reasoning dataset in which a system is given a premise sentence and must determine either the cause or effect of the premise from two possible choices. All examples are handcrafted and focus on topics from blogs and a photography-related encyclopedia.

StoryCloze: Story Cloze (Mostafazadeh et al., 2017) is a new commonsense reasoning framework for evaluating story understanding, story generation, and script learning.This test requires a system to choose the correct ending to a four-sentence story.

HellaSwag: HellaSwag (Zellers et al., 2019) is a challenge dataset for evaluating commonsense NLI that is specially hard for state-of-the-art models, though its questions are trivial for humans (>95% accuracy).

WiC: (Pilehvar & Camacho-Collados, 2019) Word-in-Context is a word sense disambiguation dataset cast as binary classification of sentence pairs. Given two text snippets and a polysemous word that appears in both sentences, the task is to determine whether the word is used with the same sense in both sentences.

A.2 Hyperparameters for Pre-Training and Finetuning

The hyperparameters used in pretraining are listed in Table 6. For the base setting, we encode the pretraining corpus with an uncased vocabulary of 32,768 BPE tokens. For the base++ setting,

Table 7: Hyper-parameters for fine-tuning.

Hyper-parameters	
Learning Rates	{1e-5, 2e-5, 3e-5, 4e-5, 5e-5}
Batch Size	{16,32}
Maximum Training Epochs	{2, 3, 5, 10}
Dropout	0.1
Warmup Step Rate	0.06
Weight Decay	0.1

we encode the corpus with a cased vocabulary of 64,000 BPE tokens. In pretraining, we use 15% masking ratio for the auxiliary MLM pretraining task. We create a `[MASK]` symbol for each masked token. Each token in X^{noise} is sampled from the softmax distribution predicted by the auxiliary model for each `[MASK]` symbol. The weight of each pretraining objective is $\lambda_{\text{MLM}} = 1$, $\lambda_{\text{RTD}} = 50$, and $\lambda_{\text{CLM}} = 1$, following Meng et al. (2021). In both the auxiliary transformer and the main transformer, we use shared token embeddings in the embedding layer and the language modeling head.

For GLUE, SQuAD, CNN/DM and XSum, the hyperparameter search space we use in finetuning are listed in Table 7. In all the reported datasets, we use exactly the same hyperparameter search space for T5 (ours), Metro-T5, and all the ablations. For T0-Eval, the hyperparameter setting is the same as pretraining except that the peak learning rate is reduced to the 1/10 of the pretraining LR.

For all the models and ablations we implemented, we report the median results of the same set of five different random seeds on GLUE, SQuAD and CNN/DM. We use three random seeds on XSum and two random seeds on T0 Eval.

A.3 Implementation Details

Implementation We implement our T5 baseline and Metro-T5 based on `fairseq`[2]. We evaluate pretrained models on the *T0 Eval* benchmark using `transformers`[3] and `t-zero`[4]. Pretraining Metro-T5 in the Base setting takes 20.8 hours on 64x NVIDIA A100 (40GB) GPUs. Pretraining Metro-T5 in the Base++ setting takes 159 hours on 128x NVIDIA A100 (40GB) GPUs.

Pretraining and Fine-tuning Costs. The pretraining cost of Metro-T5 is T5 (our implementation) plus the auxiliary transformer, whose number of layers is 1/3 of the main transformer's encoder. Under the base setting, we pretrain Metro-T5 and T5 (ours) with 64x NVIDIA A100 GPUs (40 GB Memory). The pretraining time is around 12h for T5 (ours) and 20.8h for Metro-T5. Under the base++ setting, we use 128x NVIDIA A100 GPUs and the pretraining time is around one week for both T5 (ours) and Metro-T5. In finetuning, we remove the auxiliary transformer and the RTD and CLM heads, so the finetuning cost of Metro-T5 and T5 (ours) are the same.

Projection Heads. We have three projection heads in our model: MLM head on the auxiliary transformer, RTD head on the main transformer's encoder, and CLM head on the main transformer's decoder. Both the MLM and CLM head are a single linear transformation. We use RoBERTa-style projection head for the RTD head, which contains a linear projection, a ReLU activation, a layer norm and another linear projection. For the RTD on decoder (complex CLM head) ablation, we use a RoBERTa-style head as the architecture of the CLM head.

A.4 Full Prompt-Finetuning Results on T0 Eval

Figure 5 shows full prompt-finetuning results of Metro-$T5_{\text{BASE++}}$, our $T5_{\text{BASE++}}$ baseline, and T0 (3B) (Sanh et al., 2022). Each boxplot shows the distribution of accuracies of each model using various prompt templates. Each point of T0 (3B) is the median accuracy of the model on each task reported by Sanh et al. (2022).

[2] `https://github.com/facebookresearch/fairseq`

[3] `https://huggingface.co/docs/transformers/index`

[4] `https://github.com/bigscience-workshop/t-zero`

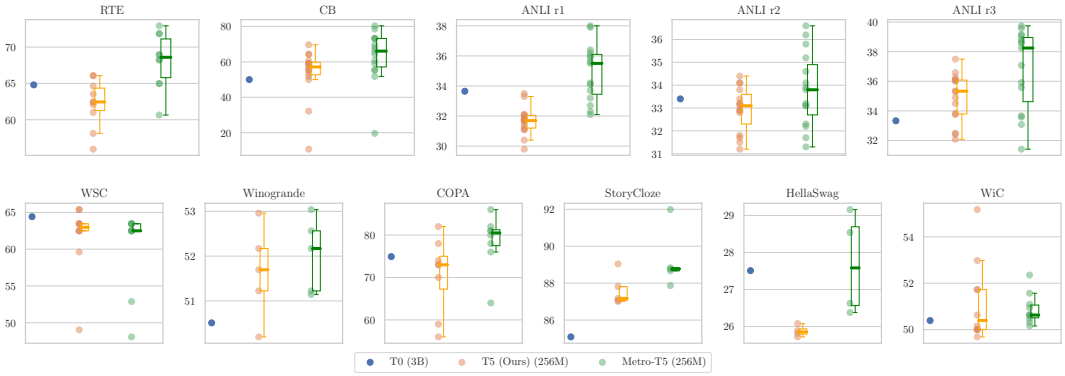


Figure 5: Full prompt-based learning results of Metro-T5$_{\text{base++}}$ versus our T5$_{\text{base++}}$ baseline and T0 (3B) on tasks from T0 benchmark (Sanh et al., 2022).

A.5 Example of the Challenge of Ill-Formed Target

In Section 3.2, we described the challenge of *Ill-formed Target* in pretraining t5 in ELECTRA-Style. This section shows a concrete example where such ill-formed target leads to ambiguities.

Table 8: An example where ill-formed target leads to ambiguities. Each number denotes a distinct subword token. `M` denotes the special token `[MASK]`. In "Auxiliary Model Prediction", a token shown in green denotes a correct prediction, where a token shown in red denotes a wrong prediction.

Sentence	1	2	3	4	5
Auxiliary Model Input 1	1	M	M	M	5
Auxiliary Model Prediction		2	6	4	
Main Model Input	1	2	6	4	5
Main Model Target	2	3	4		
Auxiliary Model Input 2	1	2	M	M	5
Auxiliary Model Prediction			6	4	
Main Model Input	1	2	6	4	5
Main Model Target	3	4			

In Table 8, the original sentence is "`1 2 3 4 5`". Using different random samples of masked positions, we can derive two masked sequences as the input of the auxiliary model: "`1 M M M 5`" and "`1 2 M M 5`". The difference is whether "`2`" is masked or not. So the target for the decoder corrective LM objective will be "`2 3 4`" and "`3 4`" respectively. After we have the masked input, the auxiliary model, which is a *masked language model (MLM)*, tries to fill masked positions with predicted tokens "`2 6 4`" and "`6 4`" respectively. The resulting main model input is "`1 2 6 4 5`" for both cases, but the target is "`2 3 4`" for case 1 and "`3 4`" for case 2. This is an ambiguity where the main model is unsure where it should begin to generate predictions: "`2`" or "`3`".