Modular Transformers: Compressing Transformers into Modularized Layers for Flexible Efficient Inference

Wangchunshu Zhou[*1] **Ronan Le Bras**[2] **Yejin Choi**[2 3]
[1]ETH Zurich
[2]Allen Institute for AI
[3]Paul G. Allen School of Computer Science & Engineering, University of Washington
wangchunshu.zhou@inf.ethz.ch

Abstract

Pre-trained Transformer models like T5 and BART have advanced the state of the art on a wide range of text generation tasks. Compressing these models into smaller ones has become critically important for practical use. Common neural network compression techniques such as knowledge distillation or quantization are limited to *static* compression where the compression ratio is fixed. In this paper, we introduce Modular Transformers, a modularized encoder-decoder framework for flexible sequence-to-sequence model compression. Modular Transformers trains modularized layers that have the same function of two or more consecutive layers in the original model via module replacing and knowledge distillation. After training, the modularized layers can be flexibly assembled into sequence-to-sequence models that meet different performance-efficiency trade-offs. Experimental results show that after a single training phase, by simply varying the assemble strategy, Modular Transformers can achieve flexible compression ratios from 1.1× to 6× with little to moderate relative performance drop.

1 Introduction

The ever increasing size of pre-trained sequence-to-sequence (seq2seq) models (Lewis et al., 2020; Raffel et al., 2019; Zhang et al., 2020; Liu et al., 2020b; Xue et al., 2021; Zhou et al., 2021) has supported advances in the state of the art in a wide range of natural language processing (NLP) tasks. For example, BART (Lewis et al., 2020) has 400 million parameters while T5 (Raffel et al., 2019) pushes this number to 11 billion. This makes large pre-trained seq2seq models hard to deploy and prone to negative environmental impacts (Strubell et al., 2019; Schwartz et al., 2020a; Xu et al., 2021b), motivating researchers to investigate methods to compress large pre-trained models into smaller, faster ones that retain strong performance. Previous work has shown that BERT (Devlin et al., 2019), a popular encoder-only pre-trained Transformer (Vaswani et al., 2017), can be effectively compressed and accelerated via different neural network compression techniques (Sanh et al., 2019; Sun et al., 2019; Jiao et al., 2020; Zhou et al., 2020; Gordon et al., 2020; Shen et al., 2020).

However, there is limited research on methods to compress pre-trained seq2seq models (Shleifer and Rush, 2020). On the other hand, pre-trained seq2seq models are generally more space and time-consuming compared to their encoder-only counterparts since they require storing the decoder as well and generally decode in an auto-regressive fashion. To satisfy ever-changing resource constraints varying in different applications and over time, existing seq2seq compression techniques must separately train and store many compact models with different compression ratios, which is computationally inefficient and may also violate space constraints. Moreover, as suggested by (Kasai et al., 2020), as opposed to encoder-only models, it is nontrivial to find proper sizes for seq2seq models with a given resource constraint as the depth for the encoder and the decoder must be jointly tuned. As such, searching for compact seq2seq models meeting different resource constraints can be very costly. This motivates us to investigate the problem of *flexible* seq2seq compression that can dynamically adjust compression ratio to meet varying resource constraints without training multiple compact models.

In this work, we present Modular Transformers, a framework to compress Transformers into modularized layers for flexible efficient inference. With the guidance from the original model, Modular Transformers trains modularized layers that have the same function of different numbers of consecutive layers in the original model via multi-grained module replacing and knowledge distillation. Specifically, we first map each of the modularized Transformer layers to a sub-module

[*]Work done while interning at the Allen Institute for AI

Findings of the Association for Computational Linguistics: ACL 2023, pages 10452–10465

(i.e., a number of consecutive layers) of the encoder or decoder of the original model. We then train the modularized layers by randomly assembling a model for each training step (replacing sub-modules of the original model with their corresponding modularized layers), while keeping the original model parameters frozen. We propose a curriculum-replacing strategy to train the modularized layers in a fine-to-coarse fashion. We also use attention and representation distillation to further encourage the modularized layers to behave similarly to the original sub-modules.

After training, the modularized layers can be flexibly assembled into seq2seq Transformers that meet different performance-efficiency trade-offs. The compression ratio for encoders and decoders can also be seamlessly adjusted to find the optimal parameterization within a fixed computation budget without any additional training. In addition, we introduce two deterministic assembling strategies tailored for smaller sizes and lower latency. Both of them replace the original model following a decoder-to-encoder, top-to-bottom, and fine-to-coarse fashion.

We empirically verify the effectiveness of Modular Transformers by compressing T5 (Raffel et al., 2019), a popular pre-trained seq2seq model, on representative text generation tasks including text summarization, machine translation, and question generation. Empirical results show that our approach consistently outperforms prior seq2seq compression techniques across different datasets while also enabling users to flexibly adjust the efficiency-performance trade-off of the model to meet different requirements for deployment.

2 Related Work

Pre-trained Model Compression Prior work has shown that BERT (Devlin et al., 2019), a popular encoder-only pre-trained Transformer (Vaswani et al., 2017), can be effectively compressed and accelerated. As summarized in Xu et al. (2021b) and Xu et al. (2021a), popular BERT compression techniques include knowledge distillation (Hinton et al., 2015; Sanh et al., 2019; Sun et al., 2019; Jiao et al., 2020; Zhou et al., 2022) which trains a compact student network to mimic the behavior of the original teacher model, pruning (LeCun et al., 1989; Michel et al., 2019; Gordon et al., 2020; Sanh et al., 2020; Wang et al., 2022) which prunes redundant neurons or structures in the original model, module replacing (Xu et al., 2020) which trains compact successor sub-modules to replace that in the original model, and quantization (Shen et al., 2020; Zafrir et al., 2019) that compresses a neural network by reducing the number of bits used to represent its parameters. In addition, a number of work also investigated efficient inference with BERT-like models with dynamic inference methods including early exit (Teerapittayanon et al., 2016; Xin et al., 2020; Liu et al., 2020a; Schwartz et al., 2020b; Zhou et al., 2020) or adaptive computation time (Graves, 2016; Eyzaguirre et al., 2021). These approaches only reduce inference latency while keep the model size unchanged. Modular Transformers focuses on the first category which reduces the size of the model while also accelerating inference.

Sequence-to-sequence Compression Compared to conventional neural network compression, seq2seq compression is relatively less explored. The de-facto method for compressing seq2seq model is sequence-level knowledge distillation (Kim and Rush, 2016), which uses the original teacher model to generate pseudo-labels with beam search on the train set and train the compact student model with the input-pseudo label pairs. A number of recent studies (Junczys-Dowmunt, 2019; Kasai et al., 2020; Zhang et al., 2021) have verified the effectiveness of the pseudo labeling method for distilling pre-trained Transformer-based seq2seq models. However, (Shleifer and Rush, 2020) shows that a simple shrink and fine-tune method performs competitively with pseudo labeling while requiring fewer computations. Another work related to Modular Transformers is LayerDrop (Fan et al., 2020), which randomly dropout Transformer layers when pre-training a seq2seq model and then prunes certain layers during inference for improved efficiency. LayerDrop differs from our method for two main reasons. First, LayerDrop must be applied *during* pre-training while Modular Transformers can be applied to any pre-trained models. This makes the scope of application of our method much broader because most existing models are not pre-trained with LayerDrop. Second, during inference, LayerDrop prunes layers while Modular Transformers replaces sub-networks with compact modules with similar functionality. This hopefully reduces the performance drop, especially when the target compression ratio is relatively large.

Moreover, there is also some prior work explor-

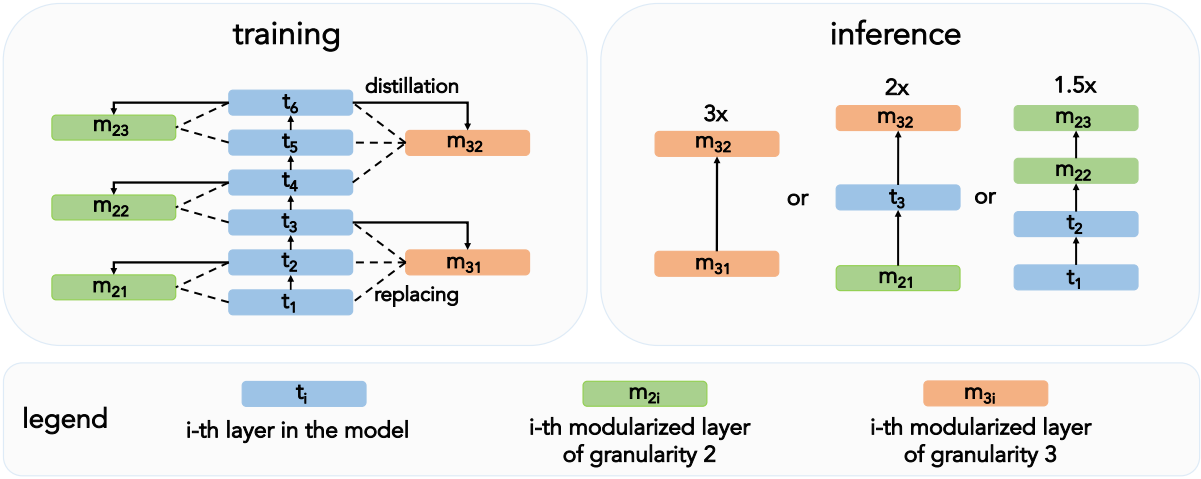

Figure 1: Illustration of the Modular Transformers framework. A set of modularized layers with different granularity are trained by multi-grained module replacing and knowledge distillation. During inference, the modularized layers are assembled to meet different resource budgets. t_i denotes the i-th layer in the original model. m_{ij} denotes j-th modularized layer with a granularity of i.

ing pruning (Michel et al., 2019; Li et al., 2021) and quantization (Li et al., 2022) techniques for seq2seq model compression. Recently, Zhou et al. (2023) proposed dynamic in-context learning for efficient text generation with prompting. These lines of work are orthogonal to knowledge distillation, pseudo labeling, and Modular Transformers and the approaches can be combined in a straightforward manner.

3 Methodology

In this section, we describe the proposed Modular Transformers framework in detail. We first recap module replacing in §3.1. We then describe the architecture design and training method for Modular Transformers in §3.2 and §3.3, respectively. Then, we present the idea of dynamic assembling in §3.4. Finally, we discuss the relationship between Modular Transformers and vanilla module replacing in §3.5.

3.1 Preliminary: Module Replacing

The goal of module replacing is essentially similar to knowledge distillation: training a compact model that behaves like the original large model. Compared to knowledge distillation which trains a student model to mimic a teacher model by minimizing the discrepancy between the predictions or hidden representations of the student and the teacher, the idea of module replacing is more direct: a compact model should behave the same way as the original model if all of its sub-networks are interchangeable with those in the original model.

Specifically, for a desired compression ratio r, we first specify a compact "successor" layer for each r consecutive layer, which we refer to as sub-networks, in the original model. Consider a model T with $n \times r$ layers, we can define a compact model S which has n layers. Let $T = \{t_1, \ldots, t_n\}$ denote the original model, t_i and s_i denote the sub-networks and layers in the original model and the compact model, respectively. The output vectors of the i-th sub-network or layer is denoted as $\mathbf{y}_i$. During compression, in each step, we sample a random variable r_{i+1} that determines whether the $(i+1)$-th sub-network in the original model is replaced by the corresponding compact layer in this step, from an independent Bernoulli distribution with a probability p of success:

$$r_{i+1} \sim \text{Bernoulli}(p) \quad (1)$$

As such, the output of the $(i+1)$-th model is calculated as:

$$\mathbf{y}_{i+1} = r_{i+1} * s_i(\mathbf{y}_i) + (1 - r_{i+1}) * t_i(\mathbf{y}_i) \quad (2)$$

where $*$ denotes the element-wise multiplication and $r_{i+1} \in \{0, 1\}$. In this way, the compact layers are trained to replace the corresponding sub-networks in the original model. After convergence, the compact layers are expected to be interchangeable (thus having the same functionality) with the original sub-networks. Moreover, the interaction between compact layers and original sub-networks and the random permutation of the hybrid model

also adds extra regularization for the training of the compact model.

After training, we collect all compact layers and combine them to be the compressed model $S = \{s_1, \ldots, s_n\}$, which will be used for efficient inference. Finally, we fine-tune the compact model to bridge the gap between module-replacing training and actual inference with the compact model.

3.2 Modularized Seq2seq Models

Modular Transformers aims to train a set of modularized layers that can be flexibly assembled into compact models with different performance-efficiency trade-offs at test time. It can directly adapted to meet different resource constraints without re-training compact models of different sizes. To achieve this goal, we propose to define modularized layers with different capacities, i.e., capturing the behavior of sub-networks of different sizes.

Specifically, given an encoder (or decoder) $T = \{t_1, \ldots, t_n\}$ with n layers and a target range of compression ratio from $1\times$ to $s\times$ (we assume n is divisible by s) , we define a suite of modularized layers $M = \{m_{ij}, i \in [n], j \in \{1, \ldots, n/i\}\}$, where $[n]$ denotes all positive integer divisors (or factors) of n except 1 and n itself, and m_{ij} denotes a modularized layer that will be trained to match a sub-network that consists of i consecutive layers starting from the $i \times (j-1) + 1$-th layer in the original model. For example, for a Transformer model with 12 encoder/decoder layers and a target maximum compression ratio of 6, Modular Transformers defines 6/4/3/2 modularized layers each corresponding to a sub-network representing 2/3/4/6 consecutive layers of the original model. Overall, Modular Transformers consists of 15 modularized layers for the encoder as well as for the decoder, which is comparable to the original model size. After training, we can combine useful modularized layers into a compact model to reach a target compression ratio and a desired level of inference efficiency (see section 3.4).

3.3 Multi-grained Module Replacing

After defining the modularized layers in Modular Transformers, we train them to have the same functionality as the original sub-networks via a combination of module replacing and knowledge distillation.

First, we extend the vanilla module replacing to multi-grained module replacing that can mix modularized layers with different granularity (i.e., target sub-network size) during training. Since the target sub-networks of modularized layers of different granularity often overlap with each other, we can not simply sample a Bernoulli random variable to determine the structure of the hybrid model used for training in the current step. Instead, we propose a greedy replacing strategy for multi-grained module replacement. Specifically, we start from the n-th (last) layer of the original model and an empty hybrid model H. Assuming that we reach the k-th layer, we sample a random variable $r_k \sim \text{Bernoulli}(p)$ where p represents the probability of replacing that layer. If $r_k = 0$, we add t_k into the hybrid model H and move to the next layer. Otherwise, we perform a module replacement and sample a random variable $r_{ki} \sim \text{Cat}(\mathbf{p})$ where $i \in [n]$ denotes the granularity of the modularized layer we want to add to the hybrid model. $\text{Cat}(\mathbf{p})$ denotes a categorical distribution on $[n]$ (or Multinoulli distribution) and $\mathbf{p}$ is the probability vector of the $|[n]|$ possible outcomes. Then we determine the nearest suitable modularized layer of granularity i as $m_{ij}, j = \lfloor k/i \rfloor$. If i divides k, we can directly add m_{ij} into H and advance to the $k-i$-th layer. Otherwise, we first append the original layers $t_{i\times j+1:k}$ to H before adding m_{ij}. In this way, for each training step, we sample a hybrid model and train its modularized layers with a task-specific objective (such as cross-entropy loss) to encourage the modularized layers to mimic the behavior of the replaced sub-networks in the original model.

In addition to module replacing, we also propose to leverage attention and hidden representation distillation (Sun et al., 2019; Jiao et al., 2020) to better align modularized layers and the corresponding sub-networks. Specifically, we train a modularized layer m_{ij} in the hybrid model to match the attention distribution and the output hidden representation of $t_{i\times j}$ in the original model. During training, we simply combine task-specific loss with distillation loss with equal weights.

Moreover, we adopt the curriculum replacement strategy from Xu et al. (2020) and progressively increase the replacing probability p to 1 during training. However, as opposed to vanilla module replacing where the model becomes static as $p = 1$, the hybrid model is still randomly combined with modularized layers of different granularity. Also, motivated by the fact that modularized layers of larger granularity are more difficult to train, we propose to train the modularized layers of different

granularity in a coarse-to-fine fashion. Specifically, for a 12-layer model (such that the modularized layers are of granularity in {2,3,4,6}), we start the training with $\mathbf{p} = [1, 0, 0, 0]$ so that only modularized layers of granularity 2 will be sampled in the hybrid model. We then progressively (linearly) change $\mathbf{p}$ to $[0.75, 0.25, 0, 0]$, $[0.5, 0.25, 0.25, 0]$, and finally to $[0.25, 0.25, 0.25, 0.25]$ so that they are sampled uniformly. We empirically set each transition phase to a quarter of all training steps.

3.4 Flexible Assembling

We describe how we can use the trained modularized layers for flexible inference. The proposed flexible assembling method starts with the original model T and gradually replaces it with modularized layers. In contrast to the random replacing strategy used for sampling the hybrid model during training, we need to take the target compression/speed-up ratio into account while also optimizing for the performance of the assembled model. To this end, we propose a deterministic decoder-first, top-down, fine-to-coarse replacing strategy for flexible assembling. The decoder-first and top-down strategy is inspired by the insight from prior work on seq2seq compression and module replacement. Specifically, Shleifer and Rush (2020) and Kasai et al. (2020) revealed that within a fixed parameter budget, seq2seq models with a deep encoder and a shallow decoder generally perform the best, and Xu et al. (2020) showed that top layers are more robust to module replacing, whereas replacing bottom layers often leads to a larger performance drop. The fine-to-coarse strategy is motivated by our conjecture that modularized layers with large granularity will more likely lead to a larger drop in performance.

In general, there exist two common compression strategies tailored for different kinds of resource budgets. The first setting focuses on the size of the compressed models and the second focuses on the inference speed. We devise two variants of our assembling strategy tailored for *speed-first* and *size-first* compression. Inspired by the previous observation of Shleifer and Rush (2020), we use a more uniform replacing strategy for size-first assembling while replacing decoder sub-networks more aggressively for speed-first assembling.

We illustrate our approach on the T5-base model with 12 encoder and 12 decoder layers. In the size-first assembling, we first replace the decoder from top to bottom with modularized layers m_{2j} corresponding to two consecutive layers. We then do the same for the encoder. After T is entirely replaced by m_{2j}, we start replacing m_{2j} with m_{3j} from decoder to encoder, from top to bottom. We do this iteratively until the entire model consists only of m_{6j} layers. As for speed-first assembling, we first replace decoder layers until they have all been replaced by m_{4j} layers. We then replace the encoder with m_{2j} and then replace the decoder with m_{6j}. Finally, we iteratively replace the encoder until the whole model consists only of m_{6j} layers.

During this process, each module replacing operation reduces the number of layers in the model by 1. As such, the compression ratio can be flexibly adjusted from 1 to the maximum granularity of the modularized layers according to different resource constraints and performance-efficiency trade-offs.

3.5 Relation with Module Replacing

Our approach differs from vanilla module replacing in four different ways: (1) module replacing is limited to compressing BERT-like encoder-only models in Xu et al. (2020) while Modular Transformers works with encoder-decoder models; (2) we propose to train modularized layers that can replace sub-networks of *different sizes* in the original model and can be connected with others, whereas the successor layers are mapped to sub-networks of a fixed size; (3) we propose multi-grained module replacing and use representation distillation to better align modularized layers with their corresponding sub-networks while vanilla module replacing only supports a fix compression ratio and is trained with cross-entropy loss only; (4) we introduce the idea of dynamic assembling that enables *flexible* adjustment of performance-efficiency trade-off, whereas the compression ratio in Xu et al. (2020) is fixed since the successor layers are directly connected together. Essentially, we can view module replacing as a method used for training modularized layers so that they are interchangeable with sub-networks in the original model.

4 Experiments

In this section, we empirically verify the effectiveness of Modular Transformers by using it to compress T5 (Raffel et al., 2019), a popular pre-trained seq2seq Transformer, on a number of representative natural language generation tasks and datasets.

Method	#Layers	Ratio	Summarization CNN/DM RG-1	CNN/DM RG-2	CNN/DM RG-L	XSum RG-1	XSum RG-2	XSum RG-L	Question Gen Squad B-4	Squad M	Squad RG-L	Translation En-De B-4
T5-base	12-12	1.0/1.0×	42.25	20.35	39.45	43.39	20.66	35.15	22.216	25.31	51.40	30.3
Compressed Models Focusing on Smaller Sizes												
Pseudo Labeling	6-6	1.5/2.0×	41.08	19.24	38.15	42.34	19.91	34.09	21.36	24.52	50.55	29.1
SFT	6-6	1.5/2.0×	41.16	19.48	38.24	42.21	19.46	33.83	21.24	24.45	50.42	28.5
KD	6-6	1.5/2.0×	41.26	19.55	38.36	42.52	19.95	34.22	21.43	24.61	50.61	29.2
Head Prune	12-12	1.2/1.9×	38.86	17.83	36.75	39.73	17.45	31.92	19.45	23.41	48.63	27.2
LayerDrop	6-6	1.5/2.0×	37.45	16.38	35.62	38.62	16.61	30.78	18.86	22.82	47.25	25.1
Quant + KD	12-12	1.1/3.9×	41.18	19.65	38.37	42.05	19.51	33.89	21.38	24.45	50.43	29.0
Modular Transformers	6-6	1.5/2.0×	**41.71**	**19.86**	**38.83**	**42.86**	**20.27**	**34.49**	**21.53**	**24.81**	**50.81**	**29.4**
Compressed Models Focusing on Lower Latency												
Pseudo Labeling	12-3	1.9/1.6×	41.25	19.38	38.27	42.49	20.01	34.26	21.40	24.45	50.47	29.3
SFT	12-3	1.9/1.6×	41.45	19.63	38.55	42.41	19.74	33.96	21.38	24.52	50.51	28.8
KD	12-3	1.9/1.6×	41.52	19.65	38.60	42.60	20.10	34.39	21.51	24.60	50.63	29.4
LayerDrop	12-3	1.9/1.6×	38.31	17.37	36.44	39.44	17.23	31.46	19.28	23.31	47.93	25.9
Modular Transformers	12-3	1.9/1.6×	**41.75**	**19.88**	**38.98**	**42.99**	**20.35**	**34.59**	**21.75**	**24.95**	**50.91**	**29.6**

Table 1: Static compression results with T5-base in both size-first (top) and speed-first (bottom) settings. For compression ratio, $a/b\times$ means the model is a times faster and b times smaller than the original model. Head Prune and Quant + KD are only considered in the size-first setting, as they cannot be adjusted for size-first or speed-first and their speed-up ratio is relatively low.

4.1 Experimental Settings

4.1.1 Models and Datasets

We conduct experiments with the T5-base and T5-large models, which have 220M and 774M parameters respectively, as the backbone models for base-size and large-size experiments.

As for tasks and datasets, we evaluate Modular Transformers on text summarization, question generation, and machine translation. For summarization, we select the CNN/DailyMail (Hermann et al., 2015) and XSUM (Narayan et al., 2018) datasets. We use the SQUAD (Rajpurkar et al., 2016) following the split in Du and Cardie (2018) for question generation and use the WMT-14 (Callison-Burch et al., 2009) En-De split for machine translation.

4.1.2 Baselines

We compare Modular Transformers with the following seq2seq compression methods: (1) **Pseudo Labeling (PL)** (Kim and Rush, 2016), also called sequence-level knowledge distillation, which uses the original model to generate pseudo labeled data for training compact models; (2) **Shrink and Fine-tune (SFT)** (Shleifer and Rush, 2020), which simply shrinks the teacher model to student size and refine-tunes the shrunk model; (3) **Knowledge Distillation (KD)** (Hinton et al., 2015), which combines logit distillation, hidden representation distillation, and attention distillation, to train a compact student model; (4) **Head Prune** (Michel et al., 2019), which prunes attention heads in the model with gradient-based head importance score; (5) **LayerDrop** (Fan et al., 2020), which randomly drops Transformer layers during fine-tuning, and selectively prunes certain layers for efficient inference; and (6) **Quant + KD** (Li et al., 2022), which combines quantization-aware training and knowledge distillation to train a quantized seq2seq model.

4.1.3 Training Details

We define modularized layers of granularity {2,3,4,6} for both T5-base and T5-large.[1] We train Modular Transformers and all related models with a warm-up ratio of 0.05, a label smoothing (Szegedy et al., 2016) rate of 0.1, and learning rate in {3e-5, 5e-5, 7e-5}. For Modular Transformers, we use a batch size of 128 and a max epoch of 24 for summarization and question generation datasets, and a batch size of 256 and a max epoch of 60 for machine translation datasets. We use the same batch size for all methods and the same number of epochs for all methods with KD. We train non-KD baselines with half the number of epochs which leads to similar performance and faster convergence. For method-specific hyperparameters, we adopt the values provided in the original contribution.

[1] We do not include granularity 12 for T5-large because it leads to negative impact on overall performance, and a model with a compression ratio of 12 has very poor performance.

			Summarization						Question Gen			Translation
			CNN/DM			XSum			SQUAD			En-De
Method	**#Layers**	**Ratio**	RG-1	RG-2	RG-L	RG-1	RG-2	RG-L	B-4	M	RG-L	B-4
T5-large	24-24	1.0/1.0×	42.61	20.72	39.81	43.83	21.05	35.44	22.75	25.45	51.62	31.2
Compressed Models Focusing on Smaller Sizes												
Pseudo Labeling	12-12	1.5/2.0×	41.54	19.61	38.51	42.75	20.23	34.44	21.61	24.73	50.65	29.6
SFT	12-12	1.5/2.0×	41.57	19.66	38.62	42.60	20.01	34.21	21.45	24.61	50.60	29.2
KD	12-12	1.5/2.0×	41.62	19.73	38.70	42.83	20.21	34.55	21.59	24.75	50.71	29.9
Quant + KD	24-24	1.1/3.9×	41.64	19.78	38.67	42.52	19.95	34.05	21.57	24.66	50.61	29.2
Modular Transformers	12-12	1.5/2.0×	**41.92**	**19.97**	**39.01**	**43.22**	**20.51**	**34.76**	**21.79**	**24.95**	**50.86**	**30.1**
Compressed Models Focusing on Lower Latency												
Pseudo Labeling	24-6	1.9/1.6×	41.76	19.79	38.68	42.92	20.28	34.51	21.72	24.94	50.81	30.1
SFT	24-6	1.9/1.6×	41.85	19.95	38.85	42.83	20.14	34.43	21.41	24.57	50.65	29.4
KD	24-6	1.9/1.6×	41.92	20.01	38.94	42.97	20.35	34.53	21.45	24.64	50.70	**30.3**
Modular Transformers	24-6	1.9/1.6×	**42.12**	**20.26**	**39.25**	**43.33**	**20.68**	**34.81**	**21.79**	**25.01**	**50.95**	**30.3**

Table 2: Static compression results with T5-large. For compression ratio, $a/b\times$ means the model is a times faster and b times smaller than the original model. We only include methods that are competitive in the T5-base experiments.

4.1.4 Evaluation

Following previous work, we report ROUGE (Lin, 2004) and BLEU (Papineni et al., 2002) for text summarization and machine translation, and report BLEU-4, METEOR (Banerjee and Lavie, 2005), and ROUGE-L for question generation. We compare Modular Transformers with baseline methods in both **fixed** and **flexible** compression ratio scenarios. In the fixed budget scenario, we experiment with both the size-first and speed-first settings. The size-first setting focuses on the size of the compressed models and compresses both the encoder and the decoder by half, while the speed-first setting focuses on the inference speed and compresses the decoder by 3/4 but leaves the encoder uncompressed. The baselines are trained twice while Modular Transformers can adjust to the two different trade-offs by simply varying the two aforementioned assembling strategies. In the flexible compression setting, we present both the trade-off between size-performance and speed-performance of Modular Transformers and compare it with several common static compression settings trained with pseudo labeling.

4.2 Experimental Results

4.2.1 Static Compression Results

We present static compression results with T5-base and T5-large as the teacher model in Table 1 and Table 2, respectively. We can see that Modular Transformers consistently outperforms all compared baselines in both size-first and speed-first settings without the need of re-training the model. We also find that while previous literature (Kim and Rush, 2016; Shleifer and Rush, 2020) observes that logits-based knowledge distillation underperforms pseudo labeling and SFT baselines, adding attention distillation and hidden representation distillation makes KD performs slightly better than these baselines. Moreover, we find LayerDrop, the other baseline that does not require re-training for different compression ratios, performs poorly when the number of layers to be dropped is relatively large, which is consistent with previous observations (Fan et al., 2020). Combining quantization and knowledge distillation performs well in terms of size-performance trade-off, which is consistent with a concurrent work by Li et al. (2022). However, the speed-up of this approach is much smaller compared to other baselines, which we believe can be partially attributed to common GPUs not being optimized for quantized models.

In addition, we find that models with 12-3 encoder/decoder layers consistently outperform their 6-6 counterparts while also leading to larger speed-ups. This is consistent with previous observations of Shleifer and Rush (2020) and it confirms the effectiveness of the speed-first assembling strategy.

4.2.2 Flexible Compression Results

Similar to the static compression experiments, we also compare Modular Transformers with the KD baseline, which is the current best-performing method, for both size-performance and speed-performance trade-offs. We present the results in Figure 2. The left figure measures the trend of Modular Transformers's performance after each replacing operation in the size-first assembling strategy. The right figure illustrates the performance of

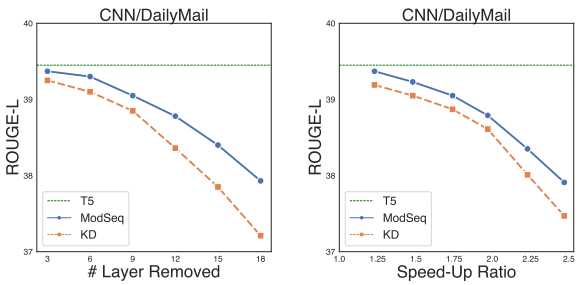

Figure 2: Flexible compression results of Modular Transformers and KD with T5-base on the CNN/DailyMail dataset in both size-first (left) and speed-first (right) settings.

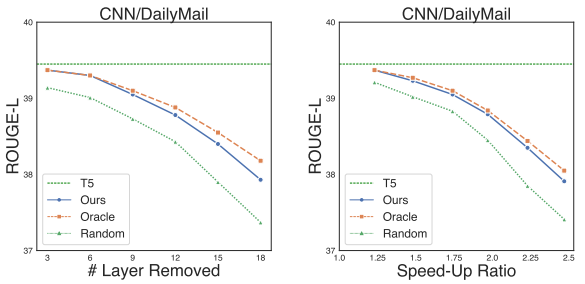

Figure 3: Comparison between our flexible assembling method, random assembling baseline, and oracle performance with T5-base on the CNN/DailyMail dataset in both size-first (left) and speed-first (right) settings.

Modular Transformers and the KD baseline with configurations that correspond to a speed-up ratio from approximately $1.25\times$ to $2.5\times$. We can see that Modular Transformers retains the performance of the original model very well when only reducing a few layers or assembling for a relatively small speed-up ratio. More importantly, Modular Transformers consistently outperforms the KD baseline, which is trained 6 times (once for each specific compression ratio), and the improvement is even larger in the extreme compression regime. This confirms the effectiveness of our approach for flexible seq2seq compression and its robustness to relatively large compression ratios.

4.3 Analysis

We then conduct a series of analyses to better understand the effectiveness of Modular Transformers. All experiments are conducted with T5-base on the CNN/DailyMail dataset.

Impact of Assembling Strategy We first analyze the impact of the proposed assembling strategies which replace layers from fine to coarse, from encoder to decoder, and from top to bottom. We compare our strategies in both the size-first and speed-first settings with the random baseline and the oracle where we exhaustively search all possibilities. The results are shown in Figure 3. We can see that our strategy is relatively close to the oracle, especially in the low compression ratio regime. In contrast, the random assembling baseline performs substantially worse, demonstrating the effectiveness of our assembling strategies.

Methods	RG-1	RG-2	RG-L
Ours	**41.53**	**19.74**	**38.62**
- w/o random replacing	41.25	19.52	38.31
- w/o curriculum replacing	41.35	19.62	38.43
- w/o knowledge distillation	41.42	19.60	38.49

Table 3: Ablation study results with T5-base on the CNN/DailyMail dataset in the size-first setting.

Impact of Random Replacing Since the assembling strategy at test time is deterministic, one may question the need for a random module replacing scheme during training. To this end, we compare Modular Transformers with the baseline where we only sample model structures used during inference. The results are shown in Table 3. We can see that this variant performs substantially worse than the random replacing method. We believe that this is due to the interaction between well-trained original layers and the modularized layers and is crucial for improving performance.

Impact of Knowledge Distillation We then study the impact of combining knowledge distillation with module replacing for training Modular Transformers. We compare the variant trained without knowledge distillation in Table 3. We find that incorporating knowledge distillation for training modularized layers improves the overall performance. This confirms the complementary nature of knowledge distillation and module replacement for the first time.

Impact of Curriculum Replacing In addition, we also analyze the impact of curriculum replacing which progressively increases the probability of including coarse-grain modularized layers in the hybrid model during training. We train a uniform sampling variant for comparison. In Table 3, we find that the uniform variant performs worse than Modular Transformers, demonstrating the effectiveness of curriculum replacing.

5 Conclusion

We introduce Modular Transformers, a framework for flexible and accurate seq2seq compression that adapts to different performance-efficiency trade-offs, which has important practical implications such as reducing the computation cost and environmental impact of deployed systems. Our method defines a set of modularized layers with different granularity and trains them to share the same functionality as the sub-networks of the original model they replace. We combine multi-grained module replacing and knowledge distillation, and design two flexible assembling strategies for size-first and speed-first inference. Empirical results on various text generation tasks show that our approach consistently outperforms previous methods while alleviating the need to re-train the compact model in order to adapt to new memory and inference time constraints.

Limitations

Our experiments focus on the T5-base and T5-large models as these are widely used, representative pre-trained seq2seq models. However, there are other pre-trained seq2seq models such as BART that we did not experiment with. It would also be interesting to experiment with pre-trained models with more layers such as T5-3B and T5-11B. We have not conducted these experiments due to resource constraints.

Ethics Statement

Our method is used to compress seq2seq Transformer models. Therefore, ethical considerations of text generation models generally apply to our method. We encourage users to assess potential biases before deploying text generation models.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 Workshop on Statistical Machine Translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 1–28, Athens, Greece. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Xinya Du and Claire Cardie. 2018. Harvesting paragraph-level question-answer pairs from Wikipedia. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1907–1917, Melbourne, Australia. Association for Computational Linguistics.

Cristóbal Eyzaguirre, Felipe del Río, Vladimir Araujo, and Álvaro Soto. 2021. Dact-bert: Differentiable adaptive computation time for an efficient bert inference. *ArXiv preprint*, abs/2109.11745.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Mitchell Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing BERT: Studying the effects of weight pruning on transfer learning. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 143–155, Online. Association for Computational Linguistics.

Alex Graves. 2016. Adaptive computation time for recurrent neural networks. *ArXiv preprint*, abs/1603.08983.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *ArXiv preprint*, abs/1503.02531.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Marcin Junczys-Dowmunt. 2019. Microsoft translator at WMT 2019: Towards large-scale document-level neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 225–233, Florence, Italy. Association for Computational Linguistics.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah A Smith. 2020. Deep encoder, shallow decoder: Reevaluating the speed-quality tradeoff in machine translation. *ArXiv preprint*, abs/2006.10369.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. *Advances in neural information processing systems*, 2.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Jiaoda Li, Ryan Cotterell, and Mrinmaya Sachan. 2021. Differentiable subset pruning of transformer heads. *Transactions of the Association for Computational Linguistics*, 9:1442–1459.

Zheng Li, Zijian Wang, Ming Tan, Ramesh Nallapati, Parminder Bhatia, Andrew Arnold, Bing Xiang, and Dan Roth. 2022. Dq-bart: Efficient sequence-to-sequence model via joint distillation and quantization. *ArXiv preprint*, abs/2203.11239.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. 2020a. FastBERT: a self-distilling BERT with adaptive inference time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6035–6044, Online. Association for Computational Linguistics.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020b. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14014–14024.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

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

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

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv preprint*, abs/1910.01108.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. 2020a. Green ai. *Communications of the ACM*, 63(12):54–63.

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A. Smith. 2020b. The right tool for the job: Matching model and instance complexities. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6640–6651, Online. Association for Computational Linguistics.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. 2020. Q-BERT: hessian based ultra low precision quantization of BERT. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI*

2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, pages 8815–8821. AAAI Press.

Sam Shleifer and Alexander M Rush. 2020. Pre-trained summarization distillation. *ArXiv preprint*, abs/2010.13002.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4323–4332, Hong Kong, China. Association for Computational Linguistics.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826. IEEE Computer Society.

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469. IEEE.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Tiannan Wang, Wangchunshu Zhou, Yan Zeng, and Xinsong Zhang. 2022. Efficientvlm: Fast and accurate vision-language models via knowledge distillation and modal-adaptive pruning.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251, Online. Association for Computational Linguistics.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. 2020. BERT-of-theseus: Compressing BERT by progressive module replacing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7859–7869, Online. Association for Computational Linguistics.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Ke Xu, Julian McAuley, and Furu Wei. 2021a. Beyond preserved accuracy: Evaluating loyalty and robustness of BERT compression. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10653–10659, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jingjing Xu, Wangchunshu Zhou, Zhiyi Fu, Hao Zhou, and Lei Li. 2021b. A survey on green deep learning. *ArXiv preprint*, abs/2111.05193.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8bert: Quantized 8bit bert. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, pages 36–39. IEEE.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020. PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Shengqiang Zhang, Xingxing Zhang, Hangbo Bao, and Furu Wei. 2021. Attention temperature matters in abstractive summarization distillation. *ArXiv preprint*, abs/2106.03441.

Wangchunshu Zhou, Tao Ge, Canwen Xu, Ke Xu, and Furu Wei. 2021. Improving sequence-to-sequence pre-training via sequence span rewriting. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 571–582, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Ryan Cotterell, and Mrinmaya Sachan. 2023. Efficient prompting via dynamic in-context learning.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian J. McAuley, Ke Xu, and Furu Wei. 2020. BERT loses patience: Fast and robust inference with early exit. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Wangchunshu Zhou, Canwen Xu, and Julian McAuley. 2022. BERT learns to teach: Knowledge distillation with meta learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7037–7049, Dublin, Ireland. Association for Computational Linguistics.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitation section

☑ A2. Did you discuss any potential risks of your work?
ethical statement

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

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

B ☑ Did you use or create scientific artifacts?

experiment section

☑ B1. Did you cite the creators of artifacts you used?
experiment section

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
they're commonly used datasets

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
they're commonly used datasets

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
they're commonly used datasets

☒ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
they're commonly used datasets

☒ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
they're commonly used datasets

C ☑ Did you run computational experiments?

experiment

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

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

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

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

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

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

Left blank.

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

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

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

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

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