Novelty Controlled Paraphrase Generation with Retrieval Augmented Conditional Prompt Tuning

Jishnu Ray Chowdhury [1*], **Yong Zhuang** [2], **Shuyi Wang** [2]

[1] University of Illinois, at Chicago
[2] Bloomberg
jraych2@uic.edu, yzhuang52@bloomberg.net, swang1072@bloomberg.net

Abstract

Paraphrase generation is a fundamental and long-standing task in natural language processing. In this paper, we concentrate on two contributions to the task: (1) we propose Retrieval Augmented Prompt Tuning (RAPT) as a parameter-efficient method to adapt large pre-trained language models for paraphrase generation; (2) we propose Novelty Conditioned RAPT (NC-RAPT) as a simple model-agnostic method of using specialized prompt tokens for controlled paraphrase generation with varying levels of lexical novelty. By conducting extensive experiments on four datasets, we demonstrate the effectiveness of the proposed approaches for retaining the semantic content of the original text while inducing lexical novelty in the generation.

Introduction

The task of paraphrase generation aims at rephrasing a given text while retaining its meaning. The task has several applications including text simplification, semantic parsing, query re-writing, and data augmentation.

Recently, the use of pre-trained Transformer-based language models has become nearly ubiquitous for different natural language processing (NLP) tasks (Devlin et al. 2019; Radford et al. 2019) including paraphrase generation (Witteveen and Andrews 2019; West et al. 2021). While the standard method of utilizing a pre-trained language model for NLP tasks is to fine-tune all the parameters in the model, it is not the most parameter-efficient. For example, MegatronLM (Shoeybi et al. 2019) has more than 11 billion parameters to fine-tune. Various methods have emerged to utilize large-scale pre-trained language models in a more parameter-efficient manner. Such methods include variants of adapter tuning (Houlsby et al. 2019), prompt tuning (Shin et al. 2020; Li and Liang 2021; Lester, Al-Rfou, and Constant 2021), and Low Rank Adaptation (LoRA) (Hu et al. 2021). Particularly, prompt tuning and LoRA can cut down the number of trainable parameters by a factor of thousands. These approaches are particularly important today as pre-trained models continuously become larger than before. In the direction of parameter-efficient methods, we propose Retrieval Augmented Prompt Tuning (RAPT) as a method of augmenting prompt tuning with kNN-based retrieved examples to further enhance the quality of paraphrases.

Besides adaptation of pre-trained language models, we also explicitly focus on the novelty of generated paraphrases because trivial paraphrases with minimal changes may not be as helpful for applications such as data augmentation. We consider the novelty of a paraphrase to be associated with the edit distance of the paraphrase from the input. Roughly, the more edits (word deletion, word insertion, substitution, etc.) there are in a paraphrase, the more "novel" we consider it to be. Paraphrases can be generated with different levels of novelty. For example, simply changing one word would still count as paraphrasing. Usually, however, the level of lexical novelty of a generated paraphrase is left upon the model to decide. In this work, we instead propose a simple model-agnostic method to put more control on the users themselves to decide the level of novelty of the generation. To this end, we propose Novelty Conditioned RAPT (NC-RAPT) that uses specialized prompts for different levels of novelty for novelty-controlled paraphrase generation. Overall, we make the following contributions:

1. We propose Retrieval Augmented Prompt Tuning (RAPT) as a parameter-efficient method for paraphrase generation.
2. We propose Novelty Conditioned RAPT (NC-RAPT) that uses specialized prompts to generate paraphrases with different levels of novelty.

Task Definition

We define paraphrase generation as a sequence-to-sequence problem. Given some input sequence of tokens $x_{1:n} = (x_1, x_2, x_3, \ldots, x_n)$ as the input query, we want a model to paraphrase the input into another sequence $y_{1:m} = (y_1, y_2, y_3, \ldots, y_m)$ serving as the output.

In our experiments, we mainly explore GPT-based models which are pre-trained on autoregressive language modeling or auto-completion. Thus, we frame our problem similar to an auto-completion task such that the downstream task remains similar to the pre-training task (potentially making the transfer of pre-trained knowledge easier).

To frame paraphrasing as an auto-completion task for

[*] The work was done during an internship with Bloomberg. Correspondence to: Jishnu Ray Chowdhury, Yong Zhuang, Shuyi Wang.

an autoregressive language model, we can design an **input prompt** such as "Input: $x_1, x_2, x_3, \ldots, x_n$ \n Paraphrase: ". The task of the model is to auto-complete the given prompt. The completed sequence will look like: "Input: $x_1, x_2, x_3, \ldots, x_n$ \n Paraphrase: $y_1, y_2, y_3, \ldots, y_m$". We treat the generated sequence $y_{1:m}$ as the paraphrase of the original input $x_{1:n}$.

In this overall sequence, "input: " is the **prompt prefix**, "\n Paraphrase: " is the **prompt infix**, "$x_1, x_2, x_3, \ldots, x_n$" is the input, and "$y_1, y_2, y_3, \ldots, y_m$" is the paraphrased output. The prefix and infix together forms the prompt template: "Input: _____\n Paraphrase: _____".

We can further generalize the input prompt format by formalizing a generic sequence of tokens denoting the prompt prefix as $p_{1:s} = (p_1, p_2, p_3, \ldots, p_s)$ and a generic sequence of tokens denoting the prompt infix as $q_{1:t} = (q_1, q_2, q_3, \ldots, q_t)$. Overall the generalized input prompt will be: "$p_1, p_2, p_3, \ldots, p_s, x_1, x_2, x_3, \ldots, x_n, q_1, q_2, q_3, \ldots, q_t$".

Baselines

In this paper, we explore the following methods to adapt large pre-trained language models. Particularly, we use GPT-based models to adapt for paraphrase generation.

Fine Tuning

Fine Tuning (FT) is the standard method of adapting a pre-trained model. In this strategy all the pre-trained parameters undergo gradient updates in some downstream tasks (in this case, paraphrase generation). In our implementation, we manually design an input prompt as "Input: $x_1, x_2, x_3, \ldots, x_n$ \n Paraphrase: " where $x_{1:n} = x_1, x_2, x_3, \ldots, x_n$ is the input as defined in the previous section. We trained the model in the auto-completion framework as discussed in the previous section.

Adapter Tuning (AT)

AT introduces some new sublayers (i.e., adapter layers) acting as low-rank bottlenecks within each Transformer layer. Generally, instead of tuning all parameters of a pre-trained model, AT focuses on tuning mainly the adapter layers. Following Houlsby et al. (2019), we only tune the adapter layer parameters and the layer normalization parameters during training. We train the model with the same input prompt format and the same framework as used in fine tuning.

Low Rank Adaptation (LoRA)

Given a pre-trained matrix $W_x \in \mathbb{R}^{d \times k}$, LoRA (Hu et al. 2021) constrains the update of W_x through a low-rank matrix $W_\delta \in \mathbb{R}^{d \times k}$. More precisely, instead of directly updating W_x, LoRA first reformulates W_x to $W'_x = W_x + W_\delta$, and then only updates the parameters involved in the construction of W_δ. W_δ itself is constructed through the multiplication of two matrices $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$ as: $W_\delta = BA$. We maintain W_δ as a low-rank matrix by keeping $r \ll min(d, k)$. Thus instead of tuning dk parameters we only need to tune $r \cdot (d + k)$ parameters. Similar to Hu et al. (2021), we only apply LoRA to the query transformation and value transformation matrices in the multi-head attention sublayers. We train the model in the same framework as used in fine tuning.

Prompt Tuning

Large-scale pre-trained language models can already contain substantial implicit knowledge for several natural language tasks before further fine tuning. For example, Radford et al. (2019) show that language models can serve as unsupervised multi-task learners. They show that using task-specific natural language prompts enable the pre-trained language models to do specific tasks like translation or summarization without further training. However, manually designing prompt templates is not ideal because it requires human involvement and the ideal prompt may not correspond to human intuitions. Instead recent work (Shin et al. 2020; Li and Liang 2021; Lester, Al-Rfou, and Constant 2021) focuses on automatically tuning the prompt template itself. Moreover, some of these works also tend to tune only the prompt template parameters keeping all the pre-trained parameters frozen. As such, these methods also serve as lightweight parameter-efficient options to adapt large language models.

For our implementation of prompt tuning, we consider the generalized input prompt format formalization as discussed in the previous section: "$p_1, p_2, p_3, \ldots, p_s, x_1, x_2, x_3, \ldots, x_n, q_1, q_2, q_3, \ldots, q_t$". The initial representation after passing this input prompt sequence through the embedding layer can be presented as: "$e(p_1), \ldots, e(p_s), e(x_1), \ldots, e(x_n), e(q_1), \ldots, e(q_t)$". Similar to Lester, Al-Rfou, and Constant (2021), we directly initialize and tune the prefix $(e(p_1), \ldots, e(p_s))$ and infix embeddings $(e(q_1), \ldots, e(q_t))$. When using prompt tuning alone, we only tune the prefix and infix embeddings (unique embedding vectors for each position) while keeping all other parameters frozen. We train the model in the same auto-completion framework as used in fine tuning.

LoRA + Prompt Tuning (LPT)

In this approach, we apply both prompt tuning and LoRA at the same time. We tune both the prefix-infix parameters and the newly introduced LoRA parameters. Both LoRA and prompt tuning tune only a minor fraction of the total parameters. Thus, we can easily combine them without increasing the overall parameter count significantly.

Proposed Approaches

In this section, we introduce our proposed approaches: Retrieval Augmented Prompt Tuning (RAPT) and Novelty Conditioned Retrieval Augmented Prompt Tuning (NC-RAPT).

Retrieval Augmented Prompt Tuning (RAPT)

In this section, we introduce our first proposed approach, RAPT, which is a synergy between an automated example retrieval strategy and an example-augmented prompt tuning strategy. We discuss both of them below. Like LPT, we also add LoRA because it does not add too many parameters (Table 2) and allows more tuning flexibility.

Example-Augmented Prompts: Brown et al. (2020) show that large-scale pre-trained auto-regressive models like GPT3 can locate or learn to perform a task from only a few input-output pairs exemplifying the task given in the input prompt context without further gradient updates. For example, let us say we have an input text sequence X (where $X = x_{1:n}$) that we want to paraphrase, and we have two exemplar paraphrase pairs (X_1, Y_1) and (X_2, Y_2). As such the input prompt to a GPT3-like model can be of the form:

$$Z = [p_{1:s}; X_2; q_{1:t}; Y_2; p_{1:s}; X_1; q_{1:t}; Y_1; p_{1:s}; X; q_{1:t}], \quad (1)$$

where Z is the full input prompt sequence, $p_{1:s}$ is the prefix sequence for a sample input, and $q_{1:t}$ is the infix sequence marking the end of the sample input and the start of the sample output. $[.; .]$ denotes concatenation.

We hypothesize that similarly augmenting the input prompt with task exemplars can be still beneficial for smaller pre-trained language models (e.g., GPT2). Hu et al. (2021) show that full data fine tuning, LoRA, or prompt tuning on GPT3 can still outperform few-shot approaches. Thus, instead of applying and exploring example-augmented prompts in few shot settings, we use example-augmented prompts in our full data prompt tuning setup. That is, we can treat the embeddings of $p_{1:s}$ and $q_{1:t}$ (the prefix and infix embedding parameters as used in our prompt tuning setup as discussed before) as freely trainable parameters. This strategy removes the need for manually designing prefix and infix sequences to best describe the task and delimit the different example inputs and outputs.

However, simply using the prompt design format as in Eq. 1 is not without its problems. During prompt tuning the prefix length (k) and/or the infix length (l) can be quite large (e.g., 250). Since the prefix and infix are repeated for every example in Eq. 1, the total length of the input prompt can increase significantly. Long sequences can slow down training and cause memory issues. To solve this issue we revise the prompt design format in Eq. 1 as:

$$Z = [d_{1:m}; p_{1:s}; X_2; q_{1:t}; Y_2; p_{1:s}; X_1; q_{1:t}; Y_1; p_{1:s}; X; q_{1:t}]. \quad (2)$$

In this format, $d_{1:m}$ serves as a global prefix serving as a sort of task description. We try to maintain $s, t \ll m$ such that most of the tunable prompt embedding parameters are concentrated on the emebddings of $d_{1:m}$, allowing us to keep s and t small. Thus, given the small s and t, the overall input prompt sequence length does not increase as much when $p_{1:s}$ and $q_{1:t}$ are repeated.

Example Retrieval: Above, we described how to design prompts to incorporate input-output examples of a task serving as task exemplars. However, the question remains open: *where do we get the task examples?* We can always try to construct the task examples manually, but that would again require human involvement.

Instead, we consider an automated method. We hypothesize that given an input sequence X, input-output example pairs that are most similar to X would be the most suitable in serving as examples to guide paraphrasing of X. Thus, we use a k-Nearest-Neighbor (kNN) algorithm to retrieve the top-k example pairs that have the most semantically similar (to X) inputs from the training dataset.

More precisely, we first encode each input text in the training data into a single sentence vector using sentence-transformers[1] (Reimers and Gurevych 2019) (we use `paraphrase-mpnet-base-v2` as the pre-trained weights). Similarly, we also encode any given input X that we want to paraphrase. We select top k examples $((X_1, Y_1), (X_2, Y_2), \ldots, (X_k, Y_k))$ from the training set based on the cosine similarity scores between the embeddings of the training examples and that of the given input X. We add the retrieved examples to the prompt (as in Eq. 2) in the ascending order of their cosine similarity scores with the input X. In practice, we set $k = 2$ given the disadvantages of long sequences that we discussed.

Novelty Conditioned RAPT (NC-RAPT)

While there are prior works (Gupta et al. 2018; Kumar et al. 2019) that have focused on improving the novelty of paraphrase generation, there are not many works focusing on controlling the level of novelty. In this work, we use a simple model-agnostic framework to control the novelty level of the generation. This framework allows us at least two benefits.

1. The framework puts more control over the user in deciding the trade-offs they want in increasing the novelty of paraphrasing.
2. The framework can also enforce a model to generate highly novel paraphrases and thus compete with prior novelty-enhancing methods.

Our framework requires two broad steps. In the first step, we classify each sample in the dataset according to the novelty of the ground truth paraphrase. We use Translation Edit Rate (TER) (Olive 2005; Snover et al. 2006) between the input sequence and the ground truth sequence to quantify the novelty of the ground truth. We use three classes of novelty levels: `High`, `Medium`, and `Low`. We classify each sample among the three novelty level classes based on the TER values between the input and the ground truth. If the TER is ≥ 0.4, we classify the sample as `High`. If the TER is > 0.2 and < 0.4, we classify the sample as `Medium`. If the TER is ≤ 0.2, we classify the sample as `Low`. The thresholds were chosen intuitively based on qualitative analysis.

In the second step, during training, we use different prefix and infix embeddings for different novelty classes. Let us say $c_{(X,Y)}$ denotes the novelty-class of an input-output paraphrase-pair (X, Y), $p^c_{1:s}$ denotes the prefix sequence for novelty class c, and $q^c_{1:t}$ denotes the infix sequence for novelty class c. Given these notations, during training, we reformulate Eq. 2 for novelty-conditioned generation as:

$$\begin{aligned} Z = [d_{1:m}; p^{c_{(X_2,Y_2)}}_{1:s}; X_2; q^{c_{(X_2,Y_2)}}_{1:t}; Y_2; \\ p^{c_{(X_1,Y_1)}}_{1:s}; X_1; q^{c_{(X_1,Y_1)}}_{1:t}; Y_1; p^{c_{(X,Y)}}_{1:s}; X; q^{c_{(X,Y)}}_{1:t}]. \end{aligned} \quad (3)$$

Here Y is the ground truth paraphrase for X. We keep the same global prefix sequence $d_{1:m}$ for all novelty classes to save parameters. As can be understood from Eq. 3, the

[1] https://www.sbert.net/docs/pretrained_models.html

Input	*Why do all of my questions get marked as needing improvement*
LPT	why do my questions get marked as needing improvement
RAPT	why are my questions still being marked as needing improvement
NC-RAPT (High)	why is my question being marked as needs to be improved
NC-RAPT (Low)	why do all of my questions get marked as needing improvement

Table 1: Examples of generated paraphrases by different models from Quora Question Pairs dataset.

	Number of Parameters	
Model	**GPT2 Medium**	**GPT2 Large**
Fine Tuning	354,823,168	774,030,080
Adapter Tuning	25,303,040	47,437,312
LoRA Tuning	786,432	1,474,560
Prompt Tuning	270,336	337,920
LPT	1,056,768	1,812,480
RAPT	1,056,768	1,812,480
NC-RAPT	1,089,536	1,853,440

Table 2: Comparison on the number of trainable parameters for different adaptation methods.

model learns from a ground truth with novelty class c only when the corresponding prefix ($p^c_{1:s}$) and infix ($q^c_{1:t}$) sequences for the same novelty class c are used. As such, during inference, the model learns to generate a paraphrase of novelty class c if the prefix ($p^c_{1:s}$) and infix ($q^c_{1:t}$) sequences for novelty class c is used for the input. During inference, to generate a paraphrase of a given input X with novelty class c, we simply use $p^c_{1:s}$ instead of $p^{c(X,Y)}_{1:s}$ and $q^c_{1:t}$ instead of $q^{c(X,Y)}_{1:t}$ in Eq. 3. NC-RAPT is built upon RAPT. So every other details are the same as for RAPT. We show some example model generations in Table 1. In Table 2, we show the total trainable parameters for all the adaptation methods that we try over GPT2 medium and GPT2 large.

Experiments

Dataset

We use four datasets for our experiments as discussed below. Details of dataset split sizes are presented in Table 3.

- **Quora Question Pairs 50K split (QQP 50K)**[2]: Quora Question Paris (QQP) is a paraphrase detection dataset. We only use the true paraphrase pairs. We use the 50K dataset split as used in Gupta et al. (2018).[3]
- **Quora Question Pairs 140K split (QQP 140K)**: We also use QQP with a different split size (QQP 140K) as used by Hosking and Lapata (2021). [4]
- **Microsoft Research Paraphrase Corpus (MSRPC)**: MSRPC (Dolan, Quirk, and Brockett 2004) is another paraphrase detection corpus. We use only the true paraphrase pairs for paraphrase generation. We use the official train-test split.
- **ParaSCI-ACL**: ParaSCI-ACL (Dong, Wan, and Cao 2021) is a paraphrase generation dataset in the scientific domain. We use the official split.[5]

Dataset Name	**Training**	**Validation**	**Test**
QQP 50K	46,000	4,000	4,000
QQP 140K	134,206	5255	5255
MSRPC	2,203	550	1,147
ParaSCI ACL	28,883	2753	2,345

Table 3: Dataset split sizes

Evaluation Metrics

We use different evaluation metrics to account for different aspects of paraphrase quality as below:

- **BLEU**: Like most prior work, we use BLEU4 (Papineni et al. 2002) to measure similarity between prediction and ground truths.
- **Self-BLEU**: BLEU by itself does not explicitly measure for the novelty of the generated paraphrase. Often, it is possible to achieve high BLEU in paraphrase generation by simply copying the original input. Thus, we also use self-BLUE4, i.e. BLEU4 between the input and the prediction to account for the novelty of the prediction. Low Self-BLEU implies high novelty.
- **Self-TER**: To check for the novelty of paraphrases beyond simply checking n-gram overlaps as in Self-BLEU, we also measure the **T**ranslation **E**dit **R**ate (TER) (Snover et al. 2006) between the input and the prediction. High self-TER implies high novelty.
- **BERT**: We also want to account for the semantic fidelity of the generated texts. A paraphrase must retain the main semantic content of the input. To measure this, we use cosine-similarity between the sentence encodings of the input and the prediction. For sentence encoding we use Sentence-BERT (Reimers and Gurevych 2019) along with `paraphrase-mpnet-base-v2`[6] as the pre-trained weights.

[2]https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

[3]https://github.com/arvind385801/paraphraseGen

[4]https://github.com/tomhosking/separator

[5]https://github.com/dqxiu/ParaSCI

[6]https://www.sbert.net/docs/pretrained_models.html

	BERT	Self-TER	Self-BLEU	BLEU	iBLEU	SARI	BERT	Self-TER	Self-BLEU	BLEU	iBLEU	SARI
Method	**Dataset: QQP 140K**						**Dataset: QQP 50K**					
Copy	100.0	0.00	100.0	32.78	-7.05	14.98	100.0	0.00	100.0	30.36	-8.75	14.44
Ground Truth	89.05	54.31	30.98	100.0	60.71	83.88	88.53	56.31	30.34	100.0	60.90	87.51
GPT2 Baselines												
Fine Tuning	94.39	29.57	58.80	33.26	5.64	38.02	92.76	32.97	54.40	29.77	4.52	38.96
Adapter Tuning	93.77	35.89	56.75	30.20	4.12	36.64	91.90	36.68	52.89	27.90	3.66	37.61
LoRA	92.92	40.87	50.16	27.74	4.37	37.89	92.03	40.09	48.98	26.68	3.98	38.87
Prompt Tuning	94.81	25.96	59.94	31.56	4.11	36.45	91.79	34.23	47.63	27.34	4.85	40.49
LPT	94.99	27.40	59.16	33.21	5.50	38.67	92.62	34.84	47.79	28.70	5.75	41.58
Ours												
RAPT	87.88	50.27	31.78	34.09	14.33	42.40	90.04	44.99	38.39	**31.61**	10.61	43.91
NC-RAPT (High)	83.89	**62.13**	**19.55**	29.88	**15.05**	42.72	86.98	**58.27**	**23.15**	25.12	**10.64**	43.23
NC-RAPT (Med)	89.84	43.47	37.77	36.47	14.19	**44.67**	92.25	37.40	43.39	30.22	8.14	**45.16**
NC-RAPT (Low)	**96.97**	16.22	76.60	**38.36**	3.87	31.14	**96.78**	18.45	69.16	31.35	1.20	40.04
Method	**Dataset: ParaSCI ACL**						**Dataset: MSRPC**					
Copy	100.00	0.00	100.0	35.66	-5.04	15.01	100.0	0.00	100.0	47.75	3.43	19.91
Ground Truth	78.55	61.22	35.76	100.00	59.27	90.57	86.15	48.03	47.77	100.0	55.67	96.72
GPT2 Baselines												
Fine Tuning	89.94	28.56	72.34	33.32	1.62	33.66	97.35	9.09	89.63	45.09	4.67	29.78
Adapter Tuning	87.47	35.17	64.5	29.79	1.50	34.00	29.54	**73.15**	**8.85**	4.23	0.31	21.26
LoRA	92.43	24.28	74.37	31.99	0.08	33.22	96.85	10.7	86.95	44.57	5.12	32.81
Prompt Tuning	91.05	23.48	71.18	29.93	-0.40	31.88	95.76	14.59	80.68	42.13	5.29	34.51
LPT	92.83	19.6	77.06	33.08	0.04	31.64	95.57	15.05	80.21	41.7	5.13	34.99
Ours												
RAPT	84.86	41.48	55.37	35.34	8.13	39.00	94.54	16.93	79.84	45.08	7.60	35.58
NC-RAPT (High)	79.44	**55.88**	**39.29**	30.32	**9.44**	**41.30**	91.79	24.78	68.64	40.42	**7.70**	**38.6**
NC-RAPT (Med)	89.49	31.17	63.94	36.30	6.23	40.44	96.56	11.57	85.42	46.14	6.67	32.88
NC-RAPT (Low)	**94.62**	17.17	79.37	**39.16**	3.60	35.54	**97.74**	7.72	90.22	**47.65**	6.29	30.18

Table 4: Comparison of different approaches to adapt GPT2 Medium for paraphrase generation on different datasets. We bold the best results for each dataset excluding Copy and Ground Truth

- **iBLEU**: iBLEU (Sun and Zhou 2012) is designed to combine both BLEU and self-BLEU to measure the balance between closeness to ground truth and novelty:

$$\text{iBLEU} = \alpha \cdot \text{BLEU} - (1-\alpha) \cdot \text{self-BLEU} \quad (4)$$

  We use BLEU4 and self-BLEU4 with α being 0.7 similar to Hosking and Lapata (2021).
- **SARI**: SARI (Xu et al. 2016) explicitly checks for the goodness of edits made on the input by the model in its predictions by comparing against the edits in the ground truth references. West et al. (2021) show that it correlates well with human judgment for paraphrase generation.

Experimental Details

In the main experiments (Table 4), we explore different adaptation methods on GPT2 medium. We share some results (Table 5) on GPT2 large too. Typically, during prompt tuning, we initialized the prompt prefix and infix token embeddings with random natural language token embeddings from GPT2 vocabulary. Besides the language model adaptation techniques, we also compare a copy-baseline that simply copies the input text and poses it as a paraphrase. The copy model checks how far we can go in scores like BLEU without changing a single word. We also try using the ground truths as predictions themselves to get a better sense of natural novelty and semantic fidelity of the ground truths in a particular dataset. We share other experimental details and hyperparameters in a later section.

Experimental Results

As shown in Table 4, RAPT substantially outperforms the baselines on all metrics except BERT in general. Increased novelty may slightly hurt BERT-based semantic similarity score.

NC-RAPT behaves mostly as expected. We get high novelty (high self-TER, low self-BLEU) when using prefix and infix for novelty class `high`; we get low novelty when conditioned on `low` novelty. We also observe a trade-off among

	BERT	Self-TER	Self-BLEU	BLEU	iBLEU	SARI	BERT	Self-TER	Self-BLEU	BLEU	iBLEU	SARI
Method	**Dataset: QQP 50K**						**Dataset: MSRPC**					
GPT2 Baselines												
Fine Tuning	92.49	34.56	50.96	29.82	5.59	40.13	95.55	14.0	84.10	43.96	5.54	34.49
Adapter Tuning	92.35	36.51	49.55	29.16	5.55	40.04	93.91	13.84	81.81	42.82	5.43	33.96
LoRA	92.98	33.47	52.96	29.67	4.88	39.81	96.80	11.31	86.11	44.88	5.58	33.62
Prompt Tuning	93.51	29.94	54.82	29.62	4.29	39.39	96.73	10.82	86.46	44.94	5.52	32.82
LPT	92.94	33.63	50.32	29.60	5.63	40.91	97.20	8.55	88.26	45.52	5.38	32.15
Ours												
RAPT	90.56	42.83	41.11	31.16	9.48	43.27	96.50	11.34	86.06	47.80	7.64	34.57
NC-RAPT (High)	87.13	**56.45**	**25.04**	26.33	**10.92**	43.25	92.37	**23.9**	**69.33**	41.40	**8.18**	**39.50**
NC-RAPT (Med)	92.11	36.15	45.25	31.75	8.65	**44.99**	97.56	8.50	88.90	47.34	6.47	30.89
NC-RAPT (Low)	**96.69**	17.43	71.29	**32.03**	1.03	38.96	**98.85**	4.32	94.32	**48.86**	5.90	27.42

Table 5: Comparison of different approaches to adapt GPT2 Large for paraphrase generation on different datasets. We bold the best results for each dataset excluding Copy and Ground Truth.

Method	Self-BLEU	BLEU	iBLEU
DiPS*	32.45	18.47	3.19
SOW/REAP*	24.19	12.64	1.59
LBoW*	29.00	16.17	2.62
SEPARATOR*	**14.84**	14.70	5.84
RAPT	31.78	**34.09**	14.33
NC-RAPT (High)	19.55	29.88	**15.05**

Table 6: Comparison with prior work on QQP 140K. * indicates that the results are taken from Hosking and Lapata (2021). We bold the best results. DiPS refers to the model proposed by Kumar et al. (2019). SOW/REAP refers to the model proposed by Goyal and Durrett (2020). LBoW refers to the model proposed by Fu, Feng, and Cunningham (2019). Separator refers to the model proposed by Hosking and Lapata (2021). Our models are based on GPT2 medium.

novelty metrics, BLEU, and BERT in NC-RAPT. Conditioning on `high` novelty increases novelty but can reduce semantic fidelity (BERT) and BLEU compared to other models. However, the reduced semantic fidelity (BERT) score is still similar to or better than that achieved by ground truth paraphrases (Ground Truth Method). NC-RAPT conditioned on `high` generally gets better novelty scores and iBLEU compared to RAPT and the baselines. NC-RAPT conditioned on `medium` can sometimes gain the best SARI scores whereas NC-RAPT conditioned on `low` can get the best BERT and BLEU scores but at a significant cost to novelty. NC-RAPT allows the users to decide which trade-offs they are willing to take by choosing a specific novelty class.

The copy models can still achieve relatively high BLEU despite simply copying the original text. This phenomenon was also noted by Mao and Lee (2019). This shows the limitation of relying on metrics like BLEU alone.

In Table 5, we show that we still get fairly consistent results even when we use GPT2 large. In Table 6, we compare with some reported results of prior work (described in the table caption). Although SEPARATOR gets the highest novelty score (lowest self-BLEU), its BLEU score is quite low. Our proposed approaches based on GPT2 medium generally achieve a much better balance between BLEU and self-BLEU and thus, a substantially higher iBLEU.

Analysis

Random Retrieval Ablation: When using RAPT we use a kNN based example retrieval as discussed before. As an ablation, we try random retrieval instead of a kNN-based approach. As shown in Table 7, retrieving semantically similar examples using kNN gives us a much better performance. These results support our hypothesis that semantically similar example pairs can better guide paraphrase generation.

However, as a collolary these results may also suggest that RAPT-based methods would not perform as well on a testing dataset where the examples are semantically distant from anything in the training dataset. This can be one limitation of the method.

Prompt Tuning Variants: We also experimented with a few different variants of prompt tuning for paraphrase generation. We reported the results in Table 8. Here, Prompt Tuning (Lester, Al-Rfou, and Constant 2021) is the method that we use in our main experiments. Prompt Tuning Random is the same method but with random initialization of the prefix-infix embeddings instead of random selection from vocabulary embeddings. P-Tuning is similar to the method proposed by Liu et al. (2021b). It uses a BiLSTM to associate the prefix and infix tokens. Prefix Tuning is similar to the method proposed by Li and Liang (2021). It uses the same prefix, infix (transfromed by an MLP layer) in every GPT2 layer. Prefix-layer Tuning was used by Hu et al. (2021). It initializes and trains prefix-infix parameters for every layer instead of just the embedding layer as in simple prompt tuning (Lester, Al-Rfou, and Constant 2021). In the results, we

	BERT	Self-TER	Self-BLEU	BLEU	iBLEU	SARI	BERT	Self-TER	Self-BLEU	BLEU	iBLEU	SARI
Method	**Dataset: QQP 50K**						**Dataset: ParaSCI ACL**					
RAPT (kNN)	90.04	**44.99**	**38.39**	**31.61**	**10.61**	**43.91**	84.86	**41.48**	**55.37**	**35.34**	**8.13**	**39.00**
RAPT (random)	**92.48**	35.39	48.11	29.50	6.22	41.65	**92.45**	20.76	75.89	32.82	0.21	31.98

Table 7: Comparison of kNN-based RAPT and RAPT with random retrieval. The results are based on adapting GPT2 medium.

Method	BERT	Self-TER	iBLEU	SARI
Prompt Tuning (PT)	91.79	34.23	**4.85**	**40.49**
PT Random	92.22	33.56	4.42	39.70
P-Tuning	**93.90**	27.74	2.66	37.33
Prefix Tuning	89.25	43.98	4.34	38.69
Prefix-layer Tuning	88.64	**47.62**	4.63	38.58

Table 8: Comparison of Prompt Tuning variants when adapting GPT2 medium on QQP 50K.

see that prompt tuning gets the best iBLEU and SARI scores. Although Prefix and Prefix-layer Tuning increase the novelty it does so at the cost of other metrics. Thus, we chose prompt tuning for our main experiments.

Hyperparameter Details

We tune the hyperparameters on QQP 50K with GPT2 medium for all the approaches. We then use the same hyperparameters for other datasets and GPT2 large. We use random search with a maximum of 50 trials and 3 epochs per trial, and choose the hyperparameters based on validation loss. For all the approaches, we search the learning rate within $\{0.1, 0.01, 1e-3, 1e-4, 5e-5\}$. For adapter tuning, we search the adapter bottleneck hidden state dimension within $\{128, 256, 512\}$.

For LPT, we tune the learning rate for LoRA paramters and prompt tuning parameters separately (we use different learning rates to tune LoRA and prompt template embeddings). For LoRA, LPT, RAPT, and NC-RAPT (all approached involving LoRA), we fix r (matrix rank) as 8 because it worked well in Hu et al. (2021). We also use a weight decay of 0.01 for LoRA-based methods. We use the official code for LoRA. [7]

We set the infix length for all prompt tuning methods to 8 because it generally provided the best performance in Hu et al. (2021). We search the prefix length of prompt tuning random, prefix tuning, and prefix-layer tuning within $\{8, 64, 256\}$. We use the same prefix length as prompt tuning random for p-tuning because both similarly operates at the embedding level. For prompt tuning, we use the same hyperparameters as tuned for prompt tuning random.

For P-Tuning, we initialize the prompt template token embeddings with a dimension of b. The initialized prompt tokens are passed through a BiLSTM (separately for prefix and infix but with shared parameters) with a total hidden size of $2\times\lfloor\frac{d}{2}\rfloor$ (where d is the pre-trained model emebdding dimensions). We then use two affine layers with a GELU activation (Hendrycks and Gimpel 2016) in-between to transform the concatenation of the forward and backward hidden states (total dimension $2\times\lfloor\frac{d}{2}\rfloor$) into a dimension d. The intermediate layer also has a dimension of d. We search b within $\{d, \lfloor\frac{d}{2}\rfloor, \lfloor\frac{d}{4}\rfloor\}$.

Also, during prefix tuning, we use a 2-layered MLP with GELU activation in-between to transform the prompt template embeddings from some dimension b to dimension d. We use an intermediate layer dimension of d for the MLP. Again, for prefix tuning too, we search for b in $\{d, \lfloor\frac{d}{2}\rfloor, \lfloor\frac{d}{4}\rfloor\}$. We use an MLP layer because it was used by (Li and Liang 2021). We also tuned and trained a version without the MLP layer but did not observe better performance than prompt tuning on iBLEU and SARI.

For RAPT and NC-RAPT, we set the length of $d_{1:m}$ as $x-8$ (where the value of x is the same as the total prefix length which is searched and tuned in prompt tuning and prompt tuning random). We set the length of $p_{1:s}$ (in RAPT and NC-RAPT) as 8 (same as infix length). Thus, we keep both $p_{1:s}$ and $q_{1:t}$ small (length 8) in RAPT and NC-RAPT whereas the majority of the parameters are concentrated on the global prefix $d_{1:m}$ ($x-8$ length) as we discussed before.

In all cases, we use AdamW (Loshchilov and Hutter 2019) as the optimizer. We also use a linear schedule with warmup for 100 steps (we use the `get_linear_schedule_with_warmup()` function from Transformers library (Wolf et al. 2020)), a gradient norm clipping with a maximum of 1, a batch size of 32, and a maximum decoding length of $n+100$ where n is the size of the input prompt (which includes the prefix tokens, infix tokens, the input to be paraphrased, and all retrieved examples if any). The selected hyperparameters for each approaches from the search are provided in Table 9.

During training, we use a maximum epoch of 30 with early stopping. We set the early stopping patience as 3. Model selection during training is done based on validation loss. The models are trained and tuned on single Tesla V100 32GB GPUs. Gradient accumulation is used to maintain the effective batch size as 32.

Related Work

Paraphrase Generation - Traditionally rule-based systems were used for paraphrase generation (McKeown 1983; Kozlowski, McCoy, and Vijay-Shanker 2003; Hassan et al. 2007). Quirk, Brockett, and Dolan (2004); Zhao et al. (2008)

[7] https://github.com/microsoft/LoRA

	Learning Rate	Adapter Bottleneck	prefix length	Prompt Dimension (b)
Method	**Dataset: QQP 50K**			
Fine Tuning	$5e-5$	—	—	—
Adapter Tuning	$1e-4$	512	—	—
LoRA	$1e-3$	—	—	—
Prompt Tuning	0.1	—	256	—
Prompt Tuning Random	0.1	—	256	—
P-Tuning	0.01	—	256	d
Prefix Tuning	$1e-4$	—	256	d
Prefix-layer Tuning	0.01	—	64	—

Table 9: Selected Hyperparameters. d is the embedding dimension of the involved pre-trained language model.

used Statistical Machine Translation (SMT) methods for paraphrasing. More recent works typically utilize Seq2Seq models for paraphrase generation (Prakash et al. 2016; Cao et al. 2017; Zhao et al. 2018; Egonmwan and Chali 2019). Mallinson, Sennrich, and Lapata (2017) proposed bilingual pivoting for paraphrasing. Li et al. (2018) utilized deep reinforcement learning to advance paraphrase generation whereas Du and Ji (2019) utilized imitation learning. Gupta et al. (2018) incorporated a variational autoencoding strategy to generate multiple diverse paraphrases. Kumar et al. (2019) proposed a submodular optimization-based decoding method to generate diverse paraphrases. Cao and Wan (2020) also improved the novelty of paraphrases through a GAN augmented with a diversity loss. Park et al. (2019) used counterfactual debiasing to generate a diverse paraphrases. Lin and Wan (2021) proposed multi-round paraphrase generation for improved diversity. Fu, Feng, and Cunningham (2019) proposed a latent-variable model grounded by bag-of-words from target sentences for paraphrase generation. Witteveen and Andrews (2019) use GPT2 for paraphrase generation. Liu et al. (2020); West et al. (2021) proposed novel unsupervised paraphrasing strategies.

Similar to our novelty-controlled generation, there are a few approaches (Iyyer et al. 2018; Li et al. 2019; Chen et al. 2019; Kumar et al. 2020; Goyal and Durrett 2020; Huang and Chang 2021; Hosking and Lapata 2021) focusing on more controllable paraphrase generation. Although these approaches can provide extra control over different aspects (eg. granularity or syntactic templates) of paraphrase generation, they do not explicitly or directly help us in controlling novelty. Moreover, most of these approaches require specialized architectures which cannot be straightforwardly utilized in the adaptation of common pre-trained language models.

Prompt Tuning - Initial work on prompt tuning focused on discrete selection of prompt template tokens (Jiang et al. 2020; Schick and Schütze 2021a; Shin et al. 2020; Schick and Schütze 2021b). Some of the newer works (Li and Liang 2021; Liu et al. 2021b; Lester, Al-Rfou, and Constant 2021; Hu et al. 2021), instead, directly tuned the prompt template token embeddings and/or intermediate layer states in the continuous space; often achieving better performance than the former strategy (Liu et al. 2021b). Our approach follows the latter direction.

Retrieval Augmentation - Hashimoto et al. (2018) introduced a retrieve-and-edit framework for structured output prediction. Kazemnejad, Salehi, and Soleymani Baghshah (2020) built upon Hashimoto et al. (2018) by using retrieved examples to augment paraphrase generation. However, their approach is not integrated with prompt tuning and uses a specialized architectural which cannot be easily utilized in adapting a generic pre-trained language model without introducing pre-training-fine-tuning discrepancies. Similar to our work, Liu et al. (2021a); Gao, Fisch, and Chen (2021) augmented the prompts for pre-trained models using kNN-based retrieved examples. However, unlike our work, they either use manual prompts or a separate model for discrete prompt prediction (instead of tuning the prompts directly in a continuous space) while focusing on few-shot settings. Also, they did not explore paraphrase generation. To the best of our knowledge, our work is the first to integrate kNN-based example retrieval with prompt tuning, in a continuous space (Li and Liang 2021; Lester, Al-Rfou, and Constant 2021), for paraphrase generation in a standard (non-few-shot) supervised training setup while, at the same time, incorporating specialized prompts for novelty controlled generation.

Conclusion

In this paper, we propose RAPT as a parameter-efficient retrieval augmented prompt tuning setting for enhanced paraphrase generation. Building up on RAPT, we further introduce NC-RAPT for novelty-controlled paraphrase generation. Our experimental results from four datasets confirms the effectiveness of our methods. As future work, we are interested in applying our proposed approaches to other large pre-trained language models to investigate how performance varies with different model architectures and size. We would also like to explore RAPT for other downstream tasks like semantic parsing, natural language inference, named entity recognition etc. Another avenue for future research, would be testing the effectiveness of the paraphrases generated by the proposed approaches for data augmentation.

Acknowledgments

We would like to thank Edgar Meij, Srivas Prasad, Nimesh Ghelani and the anonymous reviewers for their constructive feedback and suggestions. We also thank Mounica Maddela for the valuable discussions.

References

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. In *Advances in NeurIPS*.

Cao, Y.; and Wan, X. 2020. DivGAN: Towards Diverse Paraphrase Generation via Diversified Generative Adversarial Network. In *Findings of EMNLP*.

Cao, Z.; Luo, C.; Li, W.; and Li, S. 2017. Joint Copying and Restricted Generation for Paraphrase. In Singh, S. P.; and Markovitch, S., eds., *Proceedings of AAAI*.

Chen, M.; Tang, Q.; Wiseman, S.; and Gimpel, K. 2019. Controllable Paraphrase Generation with a Syntactic Exemplar. In *Proceedings of ACL*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL*.

Dolan, B.; Quirk, C.; and Brockett, C. 2004. Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In *proceedings of COLING*.

Dong, Q.; Wan, X.; and Cao, Y. 2021. ParaSCI: A Large Scientific Paraphrase Dataset for Longer Paraphrase Generation. In *Proceedings of EACL*.

Du, W.; and Ji, Y. 2019. An Empirical Comparison on Imitation Learning and Reinforcement Learning for Paraphrase Generation. In *Proceedings of EMNLP-IJCNLP*.

Egonmwan, E.; and Chali, Y. 2019. Transformer and seq2seq model for Paraphrase Generation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, 249–255. Hong Kong: Association for Computational Linguistics.

Fu, Y.; Feng, Y.; and Cunningham, J. P. 2019. Paraphrase Generation with Latent Bag of Words. In *Advances in NeurIPS*.

Gao, T.; Fisch, A.; and Chen, D. 2021. Making Pre-trained Language Models Better Few-shot Learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 3816–3830. Online: Association for Computational Linguistics.

Goyal, T.; and Durrett, G. 2020. Neural Syntactic Preordering for Controlled Paraphrase Generation. In *Proceedings of ACL*.

Gupta, A.; Agarwal, A.; Singh, P.; and Rai, P. 2018. A Deep Generative Framework for Paraphrase Generation. *Proceedings of AAAI*.

Hashimoto, T. B.; Guu, K.; Oren, Y.; and Liang, P. 2018. A Retrieve-and-Edit Framework for Predicting Structured Outputs. In *Proceedings of NeurIPS*.

Hassan, S.; Csomai, A.; Banea, C.; Sinha, R.; and Mihalcea, R. 2007. UNT: SubFinder: Combining Knowledge Sources for Automatic Lexical Substitution. In *Proceedings of SemEval*.

Hendrycks, D.; and Gimpel, K. 2016. Gaussian Error Linear Units (GELUs). *arXiv: Learning*.

Hosking, T.; and Lapata, M. 2021. Factorising Meaning and Form for Intent-Preserving Paraphrasing. In *Proceedings of ACL*.

Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; and Gelly, S. 2019. Parameter-Efficient Transfer Learning for NLP. In *Proceedings of the ICML*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; and Chen, W. 2021. LoRA: Low-Rank Adaptation of Large Language Models. *ArXiv*.

Huang, K.-H.; and Chang, K.-W. 2021. Generating Syntactically Controlled Paraphrases without Using Annotated Parallel Pairs. In *Proceedings of EACL*.

Iyyer, M.; Wieting, J.; Gimpel, K.; and Zettlemoyer, L. 2018. Adversarial Example Generation with Syntactically Controlled Paraphrase Networks. In *Proceedings of ACL*.

Jiang, Z.; Xu, F. F.; Araki, J.; and Neubig, G. 2020. How can we know what language models know? *TACL*.

Kazemnejad, A.; Salehi, M.; and Soleymani Baghshah, M. 2020. Paraphrase Generation by Learning How to Edit from Samples. In *Proceedings of ACL*.

Kozlowski, R.; McCoy, K. F.; and Vijay-Shanker, K. 2003. Generation of Single-sentence Paraphrases from Predicate/Argument Structure using Lexico-grammatical Resources. In *Proceedings of the Second International Workshop on Paraphrasing*.

Kumar, A.; Ahuja, K.; Vadapalli, R.; and Talukdar, P. 2020. Syntax-Guided Controlled Generation of Paraphrases. *TACL*.

Kumar, A.; Bhattamishra, S.; Bhandari, M.; and Talukdar, P. 2019. Submodular Optimization-based Diverse Paraphrasing and its Effectiveness in Data Augmentation. In *Proceedings of NAACL*.

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. *ArXiv*.

Li, X. L.; and Liang, P. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of ACL*.

Li, Z.; Jiang, X.; Shang, L.; and Li, H. 2018. Paraphrase Generation with Deep Reinforcement Learning. In *Proceedings of EMNLP*.

Li, Z.; Jiang, X.; Shang, L.; and Liu, Q. 2019. Decomposable Neural Paraphrase Generation. In *Proceedings of ACL*.

Lin, Z.; and Wan, X. 2021. Pushing Paraphrase Away from Original Sentence: A Multi-Round Paraphrase Generation Approach. In *Findings of ACL-IJCNLP*.

Liu, J.; Shen, D.; Zhang, Y.; Dolan, B.; Carin, L.; and Chen, W. 2021a. What Makes Good In-Context Examples for GPT-3? *ArXiv*.

Liu, X.; Mou, L.; Meng, F.; Zhou, H.; Zhou, J.; and Song, S. 2020. Unsupervised Paraphrasing by Simulated Annealing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 302–312. Online: Association for Computational Linguistics.

Liu, X.; Zheng, Y.; Du, Z.; Ding, M.; Qian, Y.; Yang, Z.; and Tang, J. 2021b. GPT Understands, Too. *arXiv preprint arXiv:2103.10385*.

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. In *ICLR*.

Mallinson, J.; Sennrich, R.; and Lapata, M. 2017. Paraphrasing Revisited with Neural Machine Translation. In *Proceedings of EACL*.

Mao, H.-R.; and Lee, H.-Y. 2019. Polly Want a Cracker: Analyzing Performance of Parroting on Paraphrase Generation Datasets. In *Proceedings of EMNLP-IJCNLP*. Association for Computational Linguistics.

McKeown, K. 1983. Paraphrasing questions using given and new information. *American Journal of Computational Linguistics*, 9(1): 1–10.

Olive, J. 2005. Global Autonomous Language Exploitation (GALE). In *DARPA/IPTO Proposer Information Pamphlet*.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, 311–318.

Park, S.; Hwang, S.-w.; Chen, F.; Choo, J.; Ha, J.-W.; Kim, S.; and Yim, J. 2019. Paraphrase Diversification Using Counterfactual Debiasing. *Proceedings of AAAI*.

Prakash, A.; Hasan, S. A.; Lee, K.; Datla, V.; Qadir, A.; Liu, J.; and Farri, O. 2016. Neural Paraphrase Generation with Stacked Residual LSTM Networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2923–2934. Osaka, Japan: The COLING 2016 Organizing Committee.

Quirk, C.; Brockett, C.; and Dolan, W. 2004. Monolingual Machine Translation for Paraphrase Generation. In *Proceedings of EMNLP*.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language Models are Unsupervised Multitask Learners. *Technical Report, Open AI*.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of EMNLP 2019*.

Schick, T.; and Schütze, H. 2021a. Exploiting Cloze-Questions for Few-Shot Text Classification and Natural Language Inference. In *Proceedings of EACL*.

Schick, T.; and Schütze, H. 2021b. It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners. In *Proceedings of NAACL*.

Shin, T.; Razeghi, Y.; Logan IV, R. L.; Wallace, E.; and Singh, S. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of EMNLP*.

Shoeybi, M.; Patwary, M.; Puri, R.; LeGresley, P.; Casper, J.; and Catanzaro, B. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Snover, M.; Dorr, B.; Schwartz, R.; Micciulla, L.; and Makhoul, J. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. *Proceedings of Association for Machine Translation in the Americas*.

Sun, H.; and Zhou, M. 2012. Joint Learning of a Dual SMT System for Paraphrase Generation. In *Proceedings of ACL*.

West, P.; Lu, X.; Holtzman, A.; Bhagavatula, C.; Hwang, J. D.; and Choi, Y. 2021. Reflective Decoding: Beyond Unidirectional Generation with Off-the-Shelf Language Models. In *Proceedings of ACL*.

Witteveen, S.; and Andrews, M. 2019. Paraphrasing with Large Language Models. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, 215–220. Hong Kong: Association for Computational Linguistics.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Scao, T. L.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. M. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of EMNLP: System Demonstrations*.

Xu, W.; Napoles, C.; Pavlick, E.; Chen, Q.; and Callison-Burch, C. 2016. Optimizing statistical machine translation for text simplification. *TACL*, 4: 401–415.

Zhao, S.; Meng, R.; He, D.; Saptono, A.; and Parmanto, B. 2018. Integrating Transformer and Paraphrase Rules for Sentence Simplification. In *Proceedings of EMNLP*.

Zhao, S.; Niu, C.; Zhou, M.; Liu, T.; and Li, S. 2008. Combining Multiple Resources to Improve SMT-based Paraphrasing Model. In *Proceedings of ACL-08: HLT*.