Generating Training Data with Language Models: Towards Zero-Shot Language Understanding

Anonymous Author(s)
Affiliation
Address
email

Abstract

Pretrained language models (PLMs) have demonstrated remarkable performance in various natural language processing tasks: Unidirectional PLMs (*e.g.*, GPT) are well known for their superior text generation capabilities; bidirectional PLMs (*e.g.*, BERT) have been the prominent choice for natural language understanding (NLU) tasks. While both types of models have achieved promising few-shot learning performance, their potential for zero-shot learning has been underexplored. In this paper, we present a simple approach that uses both types of PLMs for fully zero-shot learning of NLU tasks without requiring any task-specific data: A unidirectional PLM generates class-conditioned texts guided by prompts, which are used as the training data for fine-tuning a bidirectional PLM. With quality training data selected based on the generation probability and regularization techniques (label smoothing and temporal ensembling) applied to the fine-tuning stage for better generalization and stability, our approach demonstrates strong performance across seven classification tasks of the GLUE benchmark (*e.g.*, 72.3/73.8 on MNLI-m/mm and 92.8 on SST-2), significantly outperforming zero-shot prompting methods and achieving even comparable results to strong few-shot approaches using 32 training samples per class[1].

1 Introduction

Pretrained language models (PLMs) [5, 8, 11, 19, 33, 37] have achieved human-level performance on natural language understanding (NLU) tasks [62, 63] when fine-tuned on a large amount of task-specific training data. However, such a supervised fine-tuning paradigm is drastically different from how humans perform these tasks: We barely need to see many task-specific training samples to perform well. Recently, many studies have revealed the intriguing few-shot learning potential of PLMs: By converting task descriptions to natural language prompts and injecting them into PLMs, prompt-based approaches [5, 13, 51, 52, 55] leverage task-specific information for better training data efficiency and have achieved remarkable few-shot results.

When prompt-based methods are applied to the zero-shot setting, however, the PLMs' predictions are much less accurate. For example, GPT-3's zero-shot performance is much degraded relative to its few-shot performance [5], especially on challenging tasks like natural language inference (NLI). Without any task-specific samples, it is indeed challenging for PLMs to effectively interpret the prompts that come in different formats and are unseen in the pretraining data. To familiarize PLMs with various prompts for zero-shot generalization to unseen tasks, a recent study proposes instruction tuning [66], which fine-tunes PLMs on a large collection of different tasks described by instructions. Despite its strong performance, its success is grounded in the large number of cross-task annotated datasets (*e.g.*, train on many non-NLI tasks and transfer to NLI tasks) and the gigantic model size (*e.g.*, hundreds of billions of parameters), posing great challenges for training and using them.

[1]Code is shared in the supplementary material.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

In this work, we study zero-shot learning of PLMs on NLU tasks without any task-specific or cross-task data. Motivated by the strong text generation power of recent PLMs [5, 23, 29, 48], we propose SuperGen, a **Super**vision **Gen**eration approach, wherein training data are created via a unidirectional PLM (*i.e.*, the generator) which generates class-conditioned texts guided by label-descriptive prompts. A bidirectional PLM (*i.e.*, the classifier) is then fine-tuned on the generated texts to perform the corresponding task. Both PLMs can be of moderate size to fit in typical research hardware (*e.g.*, a GPT-2-sized [47] generator and a $\text{RoBERTa}_{\text{Large}}$-sized [33] classifier). With supervision automatically created by the generator, SuperGen eliminates the need for task-specific annotations and provides the classifier PLM with a larger amount of training data than in few-shot scenarios. SuperGen is compatible with any PLM as the classifier and any fine-tuning method. We note that the generator creates synthetic samples in a zero-shot manner, and the classifier is fine-tuned on the synthetic data (the classifier is thus not zero-shot, but there are no task-specific data required in such a process).

Across seven classification tasks of the GLUE benchmark [62], SuperGen significantly outperforms the prompt-based zero-shot method and even achieves an overall better result in both average performance and stability than strong few-shot approaches that use 32 annotated samples per class. We identify several key factors to the strong performance of SuperGen through ablation studies: (1) selecting quality training data based on their generated probability, and (2) using label smoothing and temporal ensembling to regularize fine-tuning on generated data.

2 Related Work

2.1 Few-Shot and Zero-Shot Learning with PLMs

Instead of using a large amount of annotated training data for fine-tuning PLMs on downstream tasks, few-shot learning studies how to better leverage only a small amount of task-specific training data, a more realistic scenario in many applications. The most strict few-shot learning setting does not assume access to any unlabeled data or large validation sets for hyperparameter tuning [44], where prompt-based methods [5, 13, 32, 34, 51–53, 55, 59, 80] are prominently deployed to inject task descriptions into PLMs and make effective use of their language modeling capability for improved training data efficiency in low-data regimes. More broadly, semi-supervised learning additionally leverages unlabeled task-specific data, where data augmentation [7, 69], regularization [39] and bootstrapping [52] methods are commonly used.

Zero-shot learning, on the other hand, is a much more challenging setting with absolutely no access to any task-specific data. When prompt-based methods are directly used to obtain predictions from PLMs without any training, their zero-shot performance can be much worse [5, 13]—difficult NLU tasks can be barely formulated as prompts that resemble the format of pretraining data, posing great challenges for PLMs to accurately interpret and leverage the prompts without given any training samples. The current mainstream of zero-shot learning is based on transfer learning: By converting a set of tasks with abundant annotations into instruction templates [38, 50, 66, 70], entailment pairs [75, 76] or question-answer formats [46, 82] and fine-tuning PLMs on them, the PLMs acquire the cross-task transfer ability [74] to execute unseen tasks when they are formulated in a similar format. Our work proposes a different approach from these studies: We use a unidirectional PLM to generate training data for fine-tuning another PLM on the target task. This not only removes the need for a large amount of cross-task annotations, but also eliminates the task difference in training and inference. Moreover, different from previous studies [1, 72] that rely on labeled data to fine-tune the generative PLM, we directly use prompts to guide data generation without fine-tuning.

2.2 Controlled Text Generation with PLMs

Controlled text generation [22] aims to steer the generated texts of language models towards desired contents, styles or domains. Through fine-tuning PLMs on attribute-specific data, high-level control (*e.g.*, generating certain topics or sentiments [84]), fine-grained control (*e.g.*, generating specific words or phrases [6]) or both [24] can be achieved. Adapting PLMs to generate texts of specific attributes can also be realized at inference time without any further training of the PLMs [10, 25, 26, 31, 43, 64, 71]. Different text attributes can also be represented during pretraining time as control codes [23] which later can serve as explicit guidance for generating domain/attribute-specific texts.

Along another line of controlling text generation, the idea of using prompts as guidance has emerged recently—Since natural language generation is largely based on contexts, providing certain prompts as the beginning of a sequence can effectively steer the subsequent texts to be generated. The prompts

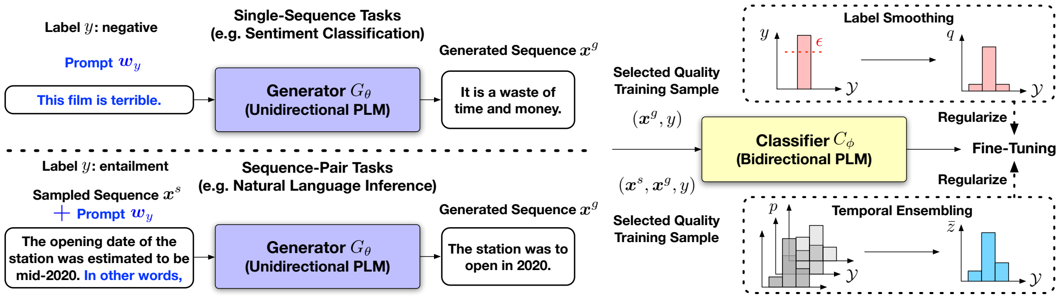


Figure 1: Overview of SuperGen for zero-shot learning of NLU tasks. A unidirectional PLM is used as the generator for creating training data guided by label-descriptive prompts. Quality training samples are selected based on average log generation probability. A bidirectional PLM is fine-tuned on the selected training set with label smoothing and temporal ensembling as regularization to perform the classification task.

can be either in natural language [53] or as learnable parameters [30]. In this work, we also guide text generation via prompts, but for the novel purpose of creating training data for NLU tasks. There have been studies with similar purposes, such as generating similar/dissimilar sentences for training sentence embeddings [54] and using labeled samples as demonstrations to prompt gigantic PLMs [77] for creating novel training data. In this work, we explore generating training data *without using any labeled samples* for a wide range of different NLU tasks. The similar setting is also explored in a concurrent study [73]. Compared to annotated task-specific data, the generated texts may contain noise and have domain difference from the downstream task. We will introduce several important strategies for effective fine-tuning of PLMs on generated data.

3 Method

3.1 Preliminaries

Problem Formulation. We consider solving a classification problem[2] where we are only given the label space $\mathcal{Y}$ and a mapping $\mathcal{M}: \mathcal{Y} \to \mathcal{W}$ that converts each label $y \in \mathcal{Y}$ into a label-descriptive prompt (*i.e.*, a short phrase) $\boldsymbol{w}_y \in \mathcal{W}$. We assume access to a unidirectional PLM G_θ as the generator and a bidirectional PLM C_ϕ which will be fine-tuned as the classifier[3]. We also assume the pretraining corpus $\mathcal{D}$ (*e.g.*, Wikipedia) is available.

Text Generation with Unidirectional PLMs. A unidirectional PLM G_θ is pretrained to maximize the generation probability of each token in a sequence $\boldsymbol{x} = [x_1, x_2, \dots, x_n]$ conditioned on previous tokens:

$$\max_\theta \prod_{i=1}^{n} p_\theta(x_i|\boldsymbol{x}_{<i}), \quad \text{where} \quad p_\theta(x_i|\boldsymbol{x}_{<i}) = \frac{\exp(\boldsymbol{e}_i^\top \boldsymbol{h}_i)}{\sum_{j=1}^{|V|} \exp(\boldsymbol{e}_j^\top \boldsymbol{h}_i)}.$$

Here, $p_\theta(\cdot)$ is usually parameterized using token embeddings $\boldsymbol{e}$ and contextualized embeddings $\boldsymbol{h}$ given by a Transformer [61] encoder.

After pretraining, G_θ can be directly used to generate new texts by recursively sampling tokens from its output probability distribution. Typically, a temperature hyperparameter $\tau > 0$ is introduced during sampling [20] to adjust the sharpness of the probability distribution:

$$p_\theta(x_i|\boldsymbol{x}_{<i}) = \frac{\exp(\boldsymbol{e}_i^\top \boldsymbol{h}_i/\tau)}{\sum_{j=1}^{|V|} \exp(\boldsymbol{e}_j^\top \boldsymbol{h}_i/\tau)}, \tag{1}$$

[2]We do not consider regression tasks in this work due to the difficulty of generating texts conditioned on a continuous label space. However, there exist approaches [14, 49] that solve regression tasks by training on classification tasks. We leave the integration of SuperGen with these methods as future work for solving regression tasks.

[3]We assume the classifier to be bidirectional PLMs since they generally work better than unidirectional PLMs in NLU tasks; we can in principle use any PLM as the classifier.

where $\tau \to 0$ approximates greedily picking the most probable next token; $\tau \to \infty$ induces a uniform distribution. Additionally, sampled tokens can be confined to the top-k most probable ones to avoid low-quality tokens. In this work, we find such top-k sampling with temperature is sufficient to produce coherent and meaningful texts as training data for NLU tasks. Exploring more sophisticated sampling strategies [21] is left for future work.

3.2 Training Data Generation

When given a label-descriptive prompt such as "Write a negative review:", humans are able to produce texts pertaining to the corresponding class. We aim to leverage the strong text generation power of a unidirectional PLM G_θ for the same purpose of creating class-conditioned training data. We note that G_θ is directly used for generation without any parameter updates. The prompts used for different NLU tasks in GLUE are summarized in Table 1.

Table 1: Prompts used to generate class-conditioned texts for different GLUE tasks. SST-2 is a single-sequence classification task and the rest are sequence-pair classification tasks. Generation for CoLA does not use prompts but by varying sampling temperatures. $\boldsymbol{x}^s$ denotes a sequence randomly sampled from the pretraining corpus; $\boldsymbol{x}^g$ denotes the sequence to be generated by G_θ; $\ldots$ denotes skipping at least one sequence. See Appendix B for more details.

Task	Label	Prompt
SST-2	positive	Rating: 5.0 $\boldsymbol{x}^g$
	negative	Rating: 1.0 $\boldsymbol{x}^g$
MNLI	entailment	$\boldsymbol{x}^s$. In other words, $\boldsymbol{x}^g$
	neutral	$\boldsymbol{x}^s$. Furthermore, $\boldsymbol{x}^g$
	contradiction	There is a rumor that $\boldsymbol{x}^s$. However, the truth is: $\boldsymbol{x}^g$
QNLI	entailment	$\boldsymbol{x}^s$? $\boldsymbol{x}^g$
	not entailment	$\boldsymbol{x}^s$? $\ldots \boldsymbol{x}^g$
RTE	entailment	$\boldsymbol{x}^s$. In other words, $\boldsymbol{x}^g$
	not entailment	$\boldsymbol{x}^s$. Furthermore, $\boldsymbol{x}^g$
MRPC	equivalent	$\boldsymbol{x}^s$. In other words, $\boldsymbol{x}^g$
	not equivalent	$\boldsymbol{x}^s$. Furthermore, $\boldsymbol{x}^g$
QQP	equivalent	$\boldsymbol{x}^s$? In other words, $\boldsymbol{x}^g$
	not equivalent	$\boldsymbol{x}^s$? Furthermore, $\boldsymbol{x}^g$

Generating Single Sequences. For single-sequence NLU tasks such as sentiment classification (*e.g.*, SST-2), we simply use a prompt $\boldsymbol{w}_y$ corresponding to label y as the beginning of the sequence and let G_θ generate the remaining sequence:

$$\boldsymbol{x}^g \leftarrow G_\theta(\boldsymbol{w}_y),$$

where $G_\theta(\boldsymbol{w}_y)$ denotes using $\boldsymbol{w}_y$ as the input to G_θ and recursively sampling tokens from the distribution in Eq. (1) until a full sequence is generated; $\boldsymbol{x}^g$ denotes the *generated* sequence (*i.e.*, excluding the prompt), which will be paired with y to form one training sample $(\boldsymbol{x}^g, y)$.

For syntactic tasks like linguistic acceptability classification (*e.g.*, CoLA) which requires generating both linguistically acceptable and unacceptable sequences, we start the sequence with random stop words and use varying sampling temperatures for generating different sequences. A smaller temperature (*e.g.*, $\tau = 0.1$ in Equation (1)) sharpens the sampling probability distribution towards the most probable tokens, thus the resulting sequence will more likely to be linguistically acceptable. Using a larger temperature (*e.g.*, $\tau = 10$ in Equation (1)) flattens the sampling probability distribution to be more uniform, and the generated tokens will be nearly random, which can create linguistically incorrect sequences.

Generating Sequence Pairs. Sequence-pair classification tasks require generating two sequences of specific relationships (*e.g.*, entailment, contradiction). We sample[4] the first sequence $\boldsymbol{x}^s$ from the pretraining corpus $\mathcal{D}$, concatenate the prompt $\boldsymbol{w}_y$ with $\boldsymbol{x}^s$, and let G_θ generate the second sequence $\boldsymbol{x}^g$:

$$\boldsymbol{x}^g \leftarrow G_\theta\left([\boldsymbol{x}^s; \boldsymbol{w}_y]\right), \ \boldsymbol{x}^s \sim \mathcal{D}.$$

The sequence pair training sample will then be formed as $(\boldsymbol{x}^s, \boldsymbol{x}^g, y)$.

Rewarding and Penalizing Repetitions for Sequence Pair Generation. A common issue in text generation is degenerate repetition [21, 23, 47, 67] where generated texts can get stuck in repetition loops. To address this issue, one approach is to discourage repetition by reducing the logits of tokens that are already in the sequence before performing sampling [23]. In sequence pair generation, however, it is sometimes desirable to encourage the second sequence to repeat some words in the first sentence (*e.g.*, for generating an entailment or a paraphrase). Therefore, we propose a simple

[4] In principle, we can also generate the first sequence using G_θ, but we find sampling from $\mathcal{D}$ improves the diversity of texts.

modification of Eq. (1) that rewards/penalizes repetition based on whether the token has appeared in $\boldsymbol{x}^s$/$\boldsymbol{x}^g$:

$$p_\theta(x_i|\boldsymbol{x}_{<i}) = \frac{\exp(\boldsymbol{e}_i^\top \boldsymbol{h}_i/\omega)}{\sum_{j=1}^{|V|}\exp(\boldsymbol{e}_j^\top \boldsymbol{h}_i/\omega)}, \quad \text{where} \quad \omega = \begin{cases} \tau\alpha & x_i \in \boldsymbol{x}^s \wedge x_i \notin \boldsymbol{x}^g \\ \tau\beta & x_i \in \boldsymbol{x}^g \\ \tau & \text{else} \end{cases}, \tag{2}$$

and $\alpha > 0, \beta > 0$ are hyperparameters. By setting $\alpha < 1$ and $\beta > 1$, we can promote tokens in $\boldsymbol{x}^s$ that have not appeared in $\boldsymbol{x}^g$ to have a higher chance of being generated, and discourage the generation of repetitive tokens in $\boldsymbol{x}^g$ to mitigate degenerate repetition. The parameters used for different tasks are listed in Appendix C Table 8.

3.3 Effective Fine-Tuning on Generated Texts

With the generated training data, one can fine-tune a bidirectional PLM C_ϕ as the classifier to perform the NLU task. However, training C_ϕ via standard supervised training on all generated texts is likely to yield suboptimal performance on downstream tasks because (1) the generated texts may contain noise as G_θ may not always produce texts pertaining to the desired class, especially for challenging sequence pair tasks with subtle semantic relationships; and (2) the generated texts can be considered as originated from the domain of G_θ's pretraining data, with a potentially different distribution from the downstream task; straightforward application of supervised training will result in overfitting to the pretraining domain and diminishing generalization ability, a common challenge in transfer learning [60, 83]. To address these challenges, we next introduce several simple and important strategies for more effective and stable fine-tuning on generated texts.

Selecting Quality Training Data. We aim to select generated texts $\boldsymbol{x}^g$ that are most likely to pertain to the desired label y (*i.e.*, with the highest $p(\boldsymbol{x}^g|y)$). The true probability $p(\boldsymbol{x}^g|y)$ is unknown and we estimate it via the generation probability given by G_θ conditioned on the prompt $\boldsymbol{w}_y$:

$$p(\boldsymbol{x}^g|y) \approx p_\theta(\boldsymbol{x}^g|\boldsymbol{w}_y) = \prod_{i=1}^{n} p_\theta\left(x_i \big| [\boldsymbol{w}_y; \boldsymbol{x}^g_{<i}]\right).$$

Since the above measure is biased towards shorter sequences, we instead use the geometric mean of the above conditional generation probability (or equivalently, the average log probability) of all tokens in $\boldsymbol{x}^g$ as the ranking score, following [78]:

$$r = \frac{1}{n}\sum_{i=1}^{n} \log p_\theta\left(x_i \big| [\boldsymbol{w}_y; \boldsymbol{x}^g_{<i}]\right). \tag{3}$$

To construct a training set consisting of N samples per class, we will generate more samples (*e.g.*, $10N$), and select training data based on the score r in Eq. (3): For all tasks except CoLA, the top-N ones of each class are selected; for CoLA, the top-N ones are used as the training sample as linguistically acceptable sequences, and the bottom-N ones are as linguistically unacceptable sequences.

Regularization for Better Generalization and Stability. Even with the above training data selection procedure, the resulting training set may still contain noise and there exists domain difference from the downstream tasks. We apply two regularization techniques, *label smoothing* [58] and *temporal ensembling* [27] for better fine-tuning stability and generalization.

Given a training sample $(\boldsymbol{x}^g, y)$, *label smoothing* trains the classifier C_ϕ to minimize the standard cross-entropy loss between the label and the classifier's prediction $p_\phi(\boldsymbol{x}^g)$, except that the label is a weighted average of the one-hot vector and a uniform distribution over all labels:

$$\min_\phi - \sum_{j=1}^{|\mathcal{Y}|} q_j \log(p_\phi(\boldsymbol{x}^g)_j), \tag{4}$$

where $q_j = \mathbb{1}(j = y)(1-\epsilon) + \epsilon/|\mathcal{Y}|$ and ϵ is the smoothing weight. By forcing the classifier to be less confident on training data, label smoothing improves robustness to label noise [35] and prevents overfitting to the training set [40], thus improving generalization to different domains.

The motivation for *temporal ensembling* is that neural networks usually first pick up easy and general patterns in the data before learning more sophisticated and dataset-specific features [79], and thus the earlier states of the network offer better generalizability to different domains. We therefore record the predictions $\boldsymbol{p}_\phi = p_\phi(\boldsymbol{x}^g)$ of C_ϕ on each training sample $(\boldsymbol{x}^g, y)$ at different training steps, and use the accumulated moving-average predictions $\bar{\boldsymbol{z}}$ to regularize the latest model training. This also helps suppress the fluctuation in model predictions due to data noise, offering better noise-robustness [41]. We update ensembled predictions $\bar{\boldsymbol{z}}$ once every B batches:

$$\hat{\boldsymbol{z}} \leftarrow \gamma \hat{\boldsymbol{z}} + (1-\gamma)\boldsymbol{p}_\phi, \; \bar{\boldsymbol{z}} \leftarrow \hat{\boldsymbol{z}}/(1-\gamma^t), \tag{5}$$

where $\hat{\boldsymbol{z}}$ has a zero initialization; γ is the momentum parameter; t is the number of updates $\bar{\boldsymbol{z}}$ has received; the division $(1-\gamma^t)$ is for bias correction [27]. We also use the ensembled prediction $\bar{\boldsymbol{z}}$ as a reliable signal to filter out noisy training samples: Only those samples on which $\bar{\boldsymbol{z}}$ strongly agrees with the label y (*i.e.*, $\bar{z}_y > \delta$ where $\delta > 0$ is a threshold parameter) will be used for training.

We regularize model training by extending Eq. (4) to add a KL divergence regularization term from the model prediction to the ensembled prediction weighed by λ:

$$\min_\phi - \sum_{j=1}^{|\mathcal{Y}|} q_j \log(p_\phi(\boldsymbol{x}^g)_j) - \lambda \sum_{j=1}^{|\mathcal{Y}|} \bar{z}_j \log \frac{p_\phi(\boldsymbol{x}^g)_j}{\bar{z}_j}. \tag{6}$$

We follow [27] to slowly ramp-up λ during training.

3.4 Overall Algorithm

We summarize SuperGen for single-sequence NLU tasks in Algorithm 1. Solving sequence-pair problems follows the same algorithm except the pretraining corpus $\mathcal{D}$ is needed for sampling the first sequence $\boldsymbol{x}^s$.

Algorithm 1: SuperGen for Zero-Shot Learning.

Input: $\mathcal{Y}$: Label space; $\mathcal{P}$: Label-descriptive prompts; G_θ: Unidirectional PLM; C_ϕ: Bidirectional PLM.
Parameter: N: Number of training samples per class to generate; $M (\gg N)$: Number of total training samples to generate; T: Number of training steps; B: Ensemble prediction update interval; δ: Threshold parameter.
Output: C_ϕ^*: Classifier that classifies input texts into $\mathcal{Y}$.

for $y \in \mathcal{Y}$ **do**
  $\mathcal{T}_y \leftarrow \{\}$
  `// Class y train set init.`
  for $i \in [1, 2, \ldots, M]$ **do**
    $\boldsymbol{x}^g \leftarrow G_\theta(\boldsymbol{w}_y)$
    $\mathcal{T}_y \leftarrow \mathcal{T}_y \bigcup \{(\boldsymbol{x}^g, y)\}$
  end
end
$\mathcal{T} \leftarrow \{\}$
`// Selected train set.`
for $y \in \mathcal{Y}$ **do**
  Sort $\mathcal{T}_y$ in descending order by Eq. (3)
  $\mathcal{T} \leftarrow \mathcal{T} \bigcup \mathcal{T}_y[:N]$
end
$\hat{\boldsymbol{z}} \leftarrow \boldsymbol{0}$
`// Ensembled prediction init.`
$\mathcal{T}^* \leftarrow \mathcal{T}$
`// Filtered train set.`
for $i \in [1, 2, \ldots, T]$ **do**
  Fine-tune C_ϕ via Eq. (6) on a minibatch of $\mathcal{T}^*$
  if $i \% B = 0$ **then**
    Update $\hat{\boldsymbol{z}}, \bar{\boldsymbol{z}}$ via Eq. (5)
    $\mathcal{T}^* \leftarrow \{(\boldsymbol{x}^g, y) | \bar{z}_y > \delta, (\boldsymbol{x}^g, y) \in \mathcal{T}\}$
  end
end
return $C_\phi^* = C_\phi$

4 Experimental Setup

Downstream Tasks and Metrics. We use all the tasks included in GLUE [62] except STS-B which is a regression task. Please refer to Appendix A for more details about GLUE tasks. We follow the evaluation protocol of [13]: We use F1 score as the metric for QQP and MRPC, Matthews correlation for CoLA, and accuracy for the rest of the tasks. The original development sets of these tasks are used for testing. For all reported results, we include the average and standard deviation over 5 different random seeds.

Models. Unless specified otherwise, we use CTRL (1.63B parameters) [23] as the generator G_θ and COCO-LM$_{\text{Large}}$ (367M parameters) [37] as the classifier C_ϕ. We also show the results using similar-sized PLMs (GPT-2 [47]/RoBERTa [33]) as the generator/classifier in Appendix D.

Fine-Tuning Settings and Hyperparameters. We note that SuperGen is compatible with any fine-tuning method; while using more sophisticated methods may grant further performance improvement, we use the basic prompt-based fine-tuning with manual templates

Table 2: Results on seven GLUE classification tasks. We report average and standard deviation (as subscripts) performance over 5 different random seeds. †: Results from LM-BFF [13].

Method	MNLI-(m/mm) (Acc.)	QQP (F1)	QNLI (Acc.)	SST-2 (Acc.)	CoLA (Matt.)	RTE (Acc.)	MRPC (F1)	AVG
Zero-Shot Setting: No task-specific data (neither labeled nor unlabeled).								
Prompting†	$50.8_{0.0}$/$51.7_{0.0}$	$49.7_{0.0}$	$50.8_{0.0}$	$83.6_{0.0}$	$2.0_{0.0}$	$51.3_{0.0}$	$61.9_{0.0}$	50.1
SuperGen	**72.3**$_{0.5}$/**73.8**$_{0.5}$	**66.1**$_{1.1}$	**73.3**$_{1.9}$	**92.8**$_{0.6}$	**32.7**$_{5.5}$	**65.3**$_{1.2}$	$82.2_{0.5}$	**69.4**
- data selection	$63.7_{1.5}$/$64.2_{1.6}$	$62.3_{2.2}$	$63.9_{3.2}$	$91.3_{2.0}$	$30.5_{8.8}$	$62.4_{1.5}$	$81.6_{0.2}$	65.1
- label smooth	$70.7_{0.8}$/$72.1_{0.7}$	$65.1_{0.9}$	$71.4_{2.5}$	$91.0_{0.9}$	$9.5_{1.0}$	$64.8_{1.1}$	**83.0**$_{0.7}$	65.2
- temporal ensemble	$62.0_{4.6}$/$63.6_{4.8}$	$63.9_{0.3}$	$72.4_{2.0}$	$92.5_{0.9}$	$23.5_{7.0}$	$63.5_{1.0}$	$78.8_{2.2}$	65.3
Few-Shot Setting: Use 32 labeled samples/class (half for training and half for development).								
Fine-tuning†	$45.8_{6.4}$/$47.8_{6.8}$	$60.7_{4.3}$	$60.2_{6.5}$	$81.4_{3.8}$	**33.9**$_{14.3}$	$54.4_{3.9}$	$76.6_{2.5}$	59.1
Manual prompt†	$68.3_{2.3}$/$70.5_{1.9}$	$65.5_{5.3}$	$64.5_{4.2}$	$92.7_{0.9}$	$9.3_{7.3}$	$69.1_{3.6}$	$74.5_{5.3}$	63.6
+ demonstration†	**70.7**$_{1.3}$/**72.0**$_{1.2}$	**69.8**$_{1.8}$	**69.2**$_{1.9}$	$92.6_{0.5}$	$18.7_{8.8}$	$68.7_{2.3}$	$77.8_{2.0}$	66.9
Auto prompt†	$68.3_{2.5}$/$70.1_{2.6}$	$67.0_{3.0}$	$68.3_{7.4}$	$92.3_{1.0}$	$14.0_{14.1}$	**73.9**$_{2.2}$	$76.2_{2.3}$	65.8
+ demonstration†	$70.0_{3.6}$/$72.0_{3.1}$	$67.7_{5.8}$	$68.5_{5.4}$	**93.0**$_{0.6}$	$21.8_{15.9}$	$71.1_{5.3}$	**78.1**$_{3.4}$	**67.3**
Fully supervised†	*89.8/89.5*	*81.7*	*93.3*	*95.0*	*62.6*	*80.9*	*91.4*	*84.9*

Table 3: Results with different groups of prompts. CoLA does not use prompts for generation. The number of prompt groups is equal to the number of the task labels.

Prompt Group	MNLI-(m/mm)	QQP	QNLI	SST-2	RTE	MRPC
# 0 (Original)	**72.3**$_{0.5}$/**73.8**$_{0.5}$	$66.1_{1.1}$	**73.3**$_{1.9}$	**92.8**$_{0.6}$	$65.3_{1.2}$	**82.2**$_{0.5}$
# 1	$70.7_{1.4}$/$72.4_{1.2}$	$65.5_{1.4}$	$71.9_{1.7}$	$92.2_{0.9}$	$64.4_{1.6}$	$81.9_{0.4}$
# 2	$70.8_{0.6}$/$72.1_{0.8}$	$65.6_{1.1}$	$72.2_{2.2}$	$92.4_{0.8}$	$64.7_{1.8}$	$81.8_{0.8}$
# 3	$70.9_{1.4}$/$72.2_{1.4}$	-	-	-	-	-
Mixed	$72.2_{0.7}$/$73.4_{0.6}$	**66.9**$_{1.5}$	$73.0_{1.7}$	**92.8**$_{0.9}$	**66.3**$_{1.0}$	$81.3_{2.0}$

approach for simplicity and clarity. For all tasks, we use the same templates and label words as in [13]. Under the zero-shot learning setting, it is not possible to tune hyperparameters due to the lack of validation sets. Therefore, we keep all fine-tuning hyperparameters (*e.g.*, learning rate, batch size, training epochs, number of generated training samples, label smoothing and temporal ensembling hyperparameters) same across all tasks. See Appendix C Table 9 for details.

Compared Methods and Ablations. We include the results of zero-shot prompting, standard few-shot fine-tuning and the four few-shot prompt-based fine-tuning methods proposed in [13]. We also conduct ablation studies by removing the following three techniques from SuperGen one at a time: (1) not using Eq. (3) for training data selection but randomly selecting the same amount of training data (- data selection); (2) not using label smoothing (- label smooth) but using one-hot labels; and (3) not using temporal ensembling (*i.e.*, using Eq. (4) instead of Eq. (6) as the training objective) (- temporal ensemble). Lastly, we report the fully supervised fine-tuning results trained on the entire training sets.

5 Evaluation

5.1 Main Results

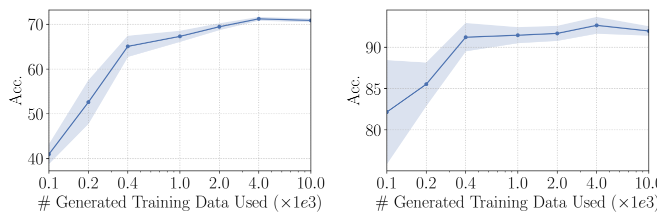
We present the results of SuperGen, its ablations and compared methods in Table 2. Overall, SuperGen significantly outperforms zero-shot prompting and achieves an overall better result than all few-shot methods. Notably, SuperGen results in much smaller variance over different random seeds than few-shot approaches on most tasks—with access to more training data, fine-tuning of PLMs becomes much more stable. The ablation results demonstrate that all three strategies (*i.e.*, quality training data selection, label smoothing and temporal ensembling) play important roles in improving and stabilizing the final performance, especially on challenging tasks like MNLI.

5.2 Using Different Prompts

One important factor of SuperGen is the choice of label-descriptive prompts as they directly influence the quality of generated training samples. To study the impact of different prompt choices on the final model performance, we create different groups of prompts other than the original ones. We replace the prompt for one label used in Table 1 with a synonymous one and keep other prompts unchanged when forming a different prompt group (Please refer to Appendix B Table 7 for details). We also experiment with mixing the generated data by different prompt groups (mixed). The results are shown in Table 3. Overall, the model performance under different prompts is quite close, except on RTE whose test set is very small, potentially resulting in the higher variance. In this work, we manually choose simple prompts that make intuitive sense, and we leave the automatic searching of optimal prompts as future work.

5.3 Results with Different Amount of Generated Data

With training data automatically created by the generator, we can have a virtually infinite amount of training samples. We show the results of using different amount of generated data (after quality data selection) for fine-tuning the classifier C_ϕ in Fig. 2 on MNLI-m and SST-2. When the number of training data is small (*e.g.*, 100), the fine-tuning variance is high, resulting in the similar instability issue with few-shot settings. With more generated data used, both average performance and training stability improve, yielding comparable results (with smaller variance) to fine-tuning using few-shot task-specific data. However, when too many generated data (*e.g.*, $10,000$) are used, the classifier's performance slightly drops, probably due to increased label noise—recall that the training data are selected based on the ranking score in Eq. (3), so using more data results in the inclusion of more lower-ranking texts in the training set and reduced data quality. One way to address this issue is to use a fixed selection ratio and increase the total number of generated texts to obtain a larger number of high-quality training data. However, this comes at a greater computation cost in the generation step. An important future direction is thus to develop better data selection strategies.



(a) MNLI-m. (b) SST-2.

Figure 2: Results with different amount of generated training data used. Dots and error bars are the average performance and the standard deviation over 5 seeds, respectively.

5.4 Using Generators for Knowledge Distillation

Apart from using unidirectional PLMs G_θ for training data generation, one could also directly apply them to unlabeled data formulated as prompts to obtain zero-shot predictions (*i.e.*, prompting [5, 13]), which can then be used as soft labels to train the classifier C_ϕ. In Table 4, we show (1) the zero-shot prediction accuracy of CTRL (the best out of three different prompts, details in Appendix F) and (2) the classifier performance trained from CTRL's predictions on the entire unlabeled training set as soft labels (*i.e.*, knowledge distillation). Similar to the observations in previous studies [5, 66, 81], the zero-shot predictions of unidirectional PLMs are quite inaccurate and directly using them as soft labels to train classifiers does not yield good results. We hypothesize that the advantages of using unidirectional PLMs for training data generation over using them for zero-shot predictions are twofold: (1) Better flexibility in prompt formats. When unidirectional PLMs are used for zero-shot predictions, the prompts have to be designed so that the label word is the last token in the sequence to be predicted, as unidirectional PLMs cannot attend to subsequent tokens. Such constraints may result in the prompt being dissimilar to the pretraining data distribution and worsen the prediction quality of the PLMs. On the contrary, using unidirectional PLMs for generation is not subject to any prompt

Table 4: Comparisons with using CTRL for zero-shot prompting and for knowledge distillation. †: The entire training set is used as unlabeled data.

Method	**MNLI-(m/mm)**	**SST-2**
SuperGen	$\mathbf{72.3}_{0.5}$/$\mathbf{73.8}_{0.5}$	$\mathbf{92.8}_{0.6}$
CTRL Prompting	$38.5_{0.0}$/$39.2_{0.0}$	$72.5_{0.0}$
Knowledge Distill†	$40.8_{0.5}$/$41.5_{0.6}$	$73.6_{0.8}$

format constraints. (2) More direct uses of PLMs' language modeling ability. Using unidirectional PLMs for training data generation *directly* leverages the PLMs' output token probability. Applying PLMs for zero-shot prediction, however, requires an additional step to convert token predictions to label predictions (*i.e.*, the verbalizer [52]), and such a mapping process usually necessitates manual curation and can hardly be optimal [13] especially without abundant task-specific data.

5.5 Using SuperGen in Few-Shot Settings

Table 5: Using SuperGen for few-shot learning. The few-shot setup follows [13].

Method	MNLI-(m/mm)	QQP	CoLA
Zero-Shot SuperGen	$72.3_{0.5}$/$73.8_{0.5}$	$66.1_{1.1}$	$32.7_{5.5}$
Few-Shot Manual Prompt	$68.3_{2.3}$/$70.5_{1.9}$	$65.5_{5.3}$	$9.3_{7.3}$
+ SuperGen	$\mathbf{73.1}_{0.9}$/$\mathbf{74.3}_{0.6}$	$\mathbf{69.9}_{2.5}$	$\mathbf{45.5}_{6.5}$

We present a simple extension of SuperGen to few-shot settings and show that the generated data of SuperGen may further improve the few-shot performance. When few-shot samples are available, we first fine-tune the classifier on the few-shot training set (standard prompt-based fine-tuning without regularization), and then continue fine-tuning the classifier on the generated data by SuperGen as described in Section 3.3. This allows the classifier to effectively leverage the knowledge from the few-shot training set to filter out noisy samples in the generated data, as temporal ensembling regularizes the classifier to remember the predictions learned previously and only keeps samples on which the model predictions agree with the label. As shown in Table 5, such a simple approach that applies SuperGen to few-shot settings improves both zero-shot SuperGen and few-shot prompt-based fine-tuning. We note that few-shot samples are not used in the training data generation stage, and we expect the results to be even better if they are leveraged to generate training data closer to the task-specific distribution. Possible ways to use few-shot samples for generation include using them as demonstrations [5], for creating augmentations [28] and for tuning the generators. We leave the explorations of generating higher quality data by leveraging few-shot samples for future work.

6 Discussions and Conclusions

Ethical Considerations. While PLMs have demonstrated remarkable text generation and understanding capability, they can come with potential risks or harms [2, 3, 5] such as generating misinformation [42] or amplifying harmful biases [45]. The focus of our work is on utilizing existing PLMs to generate training data for NLU tasks instead of developing new PLMs or generation methods. Therefore, our method can be used in company with any bias reduction and correction techniques [15, 36] to mitigate the risks of PLMs.

Limitations. One inherent limitation with zero-shot learning is the lack of access to task-specific samples for hyperparameter tuning, whereas the performance of neural networks is usually heavily dependent on the choice of hyperparameters even when the training algorithm and training set are fixed [44]. Also, without access to any labeled data, the generated training data quality may not be high enough to achieve good performance on challenging tasks, especially when the task distribution is significantly different from the pretraining data distribution (*e.g.*, the "linguistically incorrect" label of CoLA requires generating sequences with grammar mistakes – a different distribution from the one used to train PLMs). A promising direction to address the above limitations is extending SuperGen to few-shot settings and leveraging a small amount of labeled data for generating better quality data and for hyperparameter tuning.

Conclusions. We propose SuperGen, an automatic supervision generation approach for zero-shot learning of NLU tasks. By providing label-descriptive prompts as guidance to a unidirectional PLM, training data can be automatically created for fine-tuning a bidirectional PLM. Our framework differs from previous transfer-learning-based zero-shot methods in that SuperGen does not rely on cross-task annotations and eliminates the task difference in training and inference. We show that several strategies are important for effective and stable fine-tuning on generated data, including quality training data selection, label smoothing and temporal ensembling. SuperGen achieves strong performance on seven classification tasks of the GLUE benchmark, even yielding comparable or better results than sophisticated few-shot learning methods and offering better stability. There is large room for future work, including but not limited to: Extension to few-shot learning settings, exploring larger generator models, better fine-tuning techniques to leverage generated data and better strategies for selecting quality training data.

References

[1] Ateret Anaby-Tavor, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. Do not have enough data? deep learning to the rescue! In *AAAI*, 2020.

[2] Emily M. Bender and Alexander Koller. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *ACL*, 2020.

[3] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *ACM Conference on Fairness, Accountability, and Transparency*, 2021.

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

[5] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

[6] Alvin Chan, Y. Ong, Bill Tuck Weng Pung, Aston Zhang, and Jie Fu. CoCon: A self-supervised approach for controlled text generation. In *ICLR*, 2021.

[7] Jiaao Chen, Zichao Yang, and Diyi Yang. Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *ACL*, 2020.

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

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

[10] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. In *ICLR*, 2020.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

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

[13] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. In *ACL*, 2021.

[14] Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple contrastive learning of sentence embeddings. In *EMNLP*, 2021.

[15] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of EMNLP*, 2020.

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

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

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

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

[20] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015.

[21] Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *ICLR*, 2020.

[22] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. Toward controlled generation of text. In *ICML*, 2017.

[23] Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. CTRL: A conditional transformer language model for controllable generation. *ArXiv*, abs/1909.05858, 2019.

[24] Muhammad Khalifa, Hady ElSahar, and Marc Dymetman. A distributional approach to controlled text generation. In *ICLR*, 2021.

[25] Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq R. Joty, Richard Socher, and Nazneen Rajani. GeDi: Generative discriminator guided sequence generation. In *EMNLP*, 2021.

[26] Sachin Kumar, Eric Malmi, Aliaksei Severyn, and Yulia Tsvetkov. Controlled text generation as continuous optimization with multiple constraints. In *NeurIPS*, 2021.

[27] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *ICLR*, 2017.

[28] Kenton Lee, Kelvin Guu, Luheng He, Tim Dozat, and Hyung Won Chung. Neural data augmentation via example extrapolation. *arXiv preprint arXiv:2102.01335*, 2021.

[29] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, 2020.

[30] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, 2021.

[31] Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. DExperts: Decoding-time controlled text generation with experts and anti-experts. In *ACL*, 2021.

[32] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. GPT understands, too. *ArXiv*, abs/2103.10385, 2021.

[33] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[34] Robert L Logan IV, Ivana Balažević, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. Cutting down on prompts and parameters: Simple few-shot learning with language models. *arXiv preprint arXiv:2106.13353*, 2021.

[35] Michal Lukasik, Srinadh Bhojanapalli, Aditya Krishna Menon, and Surinder Kumar. Does label smoothing mitigate label noise? In *ICML*, 2020.

[36] Xinyao Ma, Maarten Sap, Hannah Rashkin, and Yejin Choi. PowerTransformer: Unsupervised controllable revision for biased language correction. In *EMNLP*, 2020.

[37] Yu Meng, Chenyan Xiong, Payal Bajaj, Saurabh Tiwary, Paul Bennett, Jiawei Han, and Xia Song. COCO-LM: Correcting and contrasting text sequences for language model pretraining. In *NeurIPS*, 2021.

[38] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*, 2021.

[39] Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. Adversarial training methods for semi-supervised text classification. In *ICLR*, 2017.

[40] Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. When does label smoothing help? In *NeurIPS*, 2019.

[41] Duc Tam Nguyen, Chaithanya Kumar Mummadi, Thi-Phuong-Nhung Ngo, Thi Hoai Phuong Nguyen, Laura Beggel, and Thomas Brox. SELF: Learning to filter noisy labels with self-ensembling. In *ICLR*, 2020.

[42] Artidoro Pagnoni, Vidhisha Balachandran, and Yulia Tsvetkov. Understanding factuality in abstractive summarization with FRANK: A benchmark for factuality metrics. In *NAACL*, 2021.

[43] Damian Pascual, Béni Egressy, Clara Meister, Ryan Cotterell, and Roger Wattenhofer. A plug-and-play method for controlled text generation. In *Findings of EMNLP*, 2021.

[44] Ethan Perez, Douwe Kiela, and Kyunghyun Cho. True few-shot learning with language models. In *NeurIPS*, 2021.

[45] Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W. Black. Style transfer through back-translation. In *ACL*, 2018.

[46] Raul Puri and Bryan Catanzaro. Zero-shot text classification with generative language models. *ArXiv*, abs/1912.10165, 2019.

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

[48] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 2019.

[49] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *EMNLP*, 2019.

[50] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang A. Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M SAIFUL BARI, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Stella Rose Biderman, Leo Gao, T. G. Owe Bers, Thomas Wolf, and Alexander M. Rush. Multitask prompted training enables zero-shot task generalization. *ArXiv*, abs/2110.08207, 2021.

[51] Teven Le Scao and Alexander M. Rush. How many data points is a prompt worth? In *NAACL*, 2021.

[52] Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *EACL*, 2021.

[53] Timo Schick and Hinrich Schütze. Few-shot text generation with natural language instructions. In *EMNLP*, 2021.

[54] Timo Schick and Hinrich Schütze. Generating datasets with pretrained language models. In *EMNLP*, 2021.

[55] Timo Schick and Hinrich Schütze. It's not just size that matters: Small language models are also few-shot learners. In *NAACL*, 2021.

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

[57] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, 2013.

[58] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[59] Derek Tam, Rakesh R Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. Improving and simplifying pattern exploiting training. In *EMNLP*, 2021.

[60] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. 2010.

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

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

[63] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS*, 2019.

[64] Zirui Wang, Adams Wei Yu, Orhan Firat, and Yuan Cao. Towards zero-label language learning. *arXiv preprint arXiv:2109.09193*, 2021.

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

[66] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. In *ICLR*, 2022.

[67] Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. In *ICLR*, 2020.

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

[69] Qizhe Xie, Zihang Dai, Eduard H. Hovy, Minh-Thang Luong, and Quoc V. Le. Unsupervised data augmentation for consistency training. In *NeurIPS*, 2020.

[70] Hanwei Xu, Yujun Chen, Yulun Du, Nan Shao, Yanggang Wang, Haiyu Li, and Zhilin Yang. ZeroPrompt: Scaling prompt-based pretraining to 1,000 tasks improves zero-shot generalization. *ArXiv*, abs/2201.06910, 2022.

[71] Kevin Yang and Dan Klein. FUDGE: Controlled text generation with future discriminators. In *NAACL*, 2021.

[72] Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. Generative data augmentation for commonsense reasoning. In *Findings of EMNLP*, 2020.

[73] Jiacheng Ye, Jiahui Gao, Qintong Li, Hang Xu, Jiangtao Feng, Zhiyong Wu, Tao Yu, and Lingpeng Kong. Zerogen: Efficient zero-shot learning via dataset generation. *ArXiv*, abs/2202.07922, 2022.

[74] Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. Crossfit: A few-shot learning challenge for cross-task generalization in nlp. In *EMNLP*, 2021.

[75] Wenpeng Yin, Jamaal Hay, and Dan Roth. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In *EMNLP*, 2019.

[76] Wenpeng Yin, Nazneen Rajani, Dragomir Radev, Richard Socher, and Caiming Xiong. Universal natural language processing with limited annotations: Try few-shot textual entailment as a start. In *EMNLP*, 2020.

[77] Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyeong Park. Gpt3mix: Leveraging large-scale language models for text augmentation. In *EMNLP Findings*, 2021.

[78] Weizhe Yuan, Graham Neubig, and Pengfei Liu. BARTScore: Evaluating generated text as text generation. In *NeurIPS*, 2021.

[79] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

[80] Ningyu Zhang, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen. Differentiable prompt makes pre-trained language models better few-shot learners. In *ICLR*, 2022.

[81] Tony Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *ICML*, 2021.

[82] Ruiqi Zhong, Kristy Lee, Zheng Zhang, and Dan Klein. Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections. In *Findings of EMNLP*, 2021.

[83] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 2020.

[84] Daniel M. Ziegler, Nisan Stiennon, Jeff Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *ArXiv*, abs/1909.08593, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A GLUE Tasks

We provide the details of the seven classification tasks included in the GLUE benchmark.

MNLI: Multi-genre Natural Language Inference [68] aims to predict whether a given premise sentence entails, contradicts or neutral with respect to a given hypothesis sentence.

QQP: Quora Question Pairs [56] aims to determine whether a pair of questions asked are semantically equivalent.

QNLI: Question Natural Language Inference aims to predict whether a given sentence contains the answer to a given question sentence.

SST-2: Stanford Sentiment Treebank [57] aims to determine if a movie review has positive or negative sentiment.

CoLA: Corpus of Linguistic Acceptability [65] aims to determine whether a given sentence is linguistically acceptable or not.

RTE: Recognizing Textual Entailment [4, 9, 16, 18] aims to predict whether a given premise sentence entails a given hypothesis sentence or not.

MRPC: Microsoft Research Paraphrase Corpus [12] aims to predict whether two sentences are semantically equivalent or not.

B Details of Prompts Used for Different Tasks

Table 6: Extensions of Table 1 with more details of prompts used to generate class-conditioned texts for different GLUE tasks. SST-2 and CoLA are single-sequence classification tasks and the rest are sequence-pair classification tasks. Generation for CoLA does not use prompts but by varying sampling temperatures. Text generation with CTRL [23] requires starting with control codes, and we use the ones that correspond to the pretraining corpus where the first sequence is sampled: For MNLI, RTE and MRPC, the first sequence is sampled from Wikipedia; for QNLI and QQP, the first sequence is sampled from OpenWebText [17]. $\boldsymbol{x}^s$ denotes a sequence randomly sampled from the pretraining corpus; $\boldsymbol{x}^g$ denotes the sequence to be generated by G_θ; $\ldots$ denotes skipping at least one sequence. The prompts used for SST-2 are part of the CTRL [23] codes.

Task	Task Type	Control Code	Label	Prompt
SST-2	single-sequence	Reviews	positive negative	Rating: 5.0 $\boldsymbol{x}^g$ Rating: 1.0 $\boldsymbol{x}^g$
CoLA	single-sequence	Links	grammatical not grammatical	$\boldsymbol{x}^g$ $\boldsymbol{x}^g$
MNLI	sequence-pair	Wikipedia	entailment neutral contradiction	$\boldsymbol{x}^s$. In other words, $\boldsymbol{x}^g$ $\boldsymbol{x}^s$. Furthermore, $\boldsymbol{x}^g$ There is a rumor that $\boldsymbol{x}^s$. However, the truth is: $\boldsymbol{x}^g$
QNLI	sequence-pair	Links	entailment not entailment	$\boldsymbol{x}^s$? $\boldsymbol{x}^g$ $\boldsymbol{x}^s$? $\ldots \boldsymbol{x}^g$
RTE	sequence-pair	Wikipedia	entailment not entailment	$\boldsymbol{x}^s$. In other words, $\boldsymbol{x}^g$ $\boldsymbol{x}^s$. Furthermore, $\boldsymbol{x}^g$
MRPC	sequence-pair	Wikipedia	equivalent not equivalent	$\boldsymbol{x}^s$. In other words, $\boldsymbol{x}^g$ $\boldsymbol{x}^s$. Furthermore, $\boldsymbol{x}^g$
QQP	sequence-pair	Links	equivalent not equivalent	$\boldsymbol{x}^s$? In other words, $\boldsymbol{x}^g$ $\boldsymbol{x}^s$? Furthermore, $\boldsymbol{x}^g$

We present more details about the prompts used for different tasks in Table 6 which is an extended version of Table 1.

For SST-2, we fix the beginning of the generated sequence $\boldsymbol{x}^g$ to be "The/this film/movie" to make sure the generated texts are related to movie reviews.

For CoLA, we start the generated sequence $\boldsymbol{x}^g$ with a random stop word.

Table 7: Different prompt groups used in the experiments of Section 5.2. We replace the original prompt for each label with an alternative one and keep other prompts unchanged when forming a different prompt group.

Task	Label	Original	Alternative
SST-2	positive	Rating: 5.0 $\boldsymbol{x}^g$	Rating: 4.0 $\boldsymbol{x}^g$
	negative	Rating: 1.0 $\boldsymbol{x}^g$	Rating: 2.0 $\boldsymbol{x}^g$
MNLI	entailment	$\boldsymbol{x}^s$. In other words, $\boldsymbol{x}^g$	$\boldsymbol{x}^s$. To put it another way, $\boldsymbol{x}^g$
	neutral	$\boldsymbol{x}^s$. Furthermore, $\boldsymbol{x}^g$	$\boldsymbol{x}^s$. In addition, $\boldsymbol{x}^g$
	contradiction	There is a rumor that $\boldsymbol{x}^s$. However, the truth is: $\boldsymbol{x}^g$	People believe that $\boldsymbol{x}^s$. However, the truth is: $\boldsymbol{x}^g$
QNLI	entailment	$\boldsymbol{x}^s$? $\boldsymbol{x}^g$	Question: $\boldsymbol{x}^s$? Answer: $\boldsymbol{x}^g$
	not entailment	$\boldsymbol{x}^s$? $\ldots\boldsymbol{x}^g$	Question: $\boldsymbol{x}^s$? Answer: $\ldots\boldsymbol{x}^g$
RTE	entailment	$\boldsymbol{x}^s$. In other words, $\boldsymbol{x}^g$	$\boldsymbol{x}^s$. To put it another way, $\boldsymbol{x}^g$
	not entailment	$\boldsymbol{x}^s$. Furthermore, $\boldsymbol{x}^g$	$\boldsymbol{x}^s$. In addition, $\boldsymbol{x}^g$
MRPC	equivalent	$\boldsymbol{x}^s$. In other words, $\boldsymbol{x}^g$	$\boldsymbol{x}^s$. To put it another way, $\boldsymbol{x}^g$
	not equivalent	$\boldsymbol{x}^s$. Furthermore, $\boldsymbol{x}^g$	$\boldsymbol{x}^s$. In addition, $\boldsymbol{x}^g$
QQP	equivalent	$\boldsymbol{x}^s$? In other words, $\boldsymbol{x}^g$	$\boldsymbol{x}^s$? To put it another way, $\boldsymbol{x}^g$
	not equivalent	$\boldsymbol{x}^s$? Furthermore, $\boldsymbol{x}^g$	$\boldsymbol{x}^s$? In addition, $\boldsymbol{x}^g$

For QNLI and QQP, the first sequence is always a question, and we require the sampled sequence $\boldsymbol{x}^s$ to end with a question mark and begin with one of the following words: “how”, “what”, “why”, “who”, “which”, “where”, “when”, “whom”, “whose”.

For QNLI, the generated sequence $\boldsymbol{x}^g$ for the “entailment” label is the one that immediately follows the sampled sequence $\boldsymbol{x}^s$; the generated sequence $\boldsymbol{x}^g$ for the “not entailment” label is randomly sampled from the paragraph following $\boldsymbol{x}^g$ excluding the first sequence that immediately follows $\boldsymbol{x}^g$.

We also show the different prompt groups used in the experiments of Section 5.2 in Table 7.

C Hyperparameters and Reproducibility

Hyperparameters for Generating Training Data. Table 8 lists the hyperparameters used in the training data generation stage. For sequence-pair tasks, we use greedy sampling for better reproducibility. For labels that require generating entailment, paraphrase, or equivalent sequence pairs, we set $\alpha \leq 1$ to encourage word overlapping between the second sequence and the first sequence; otherwise, we set $\alpha = \beta > 1$ to discourage word repetition.

To construct a training set consisting of N samples per class, we will generate M samples per class, and select training data based on the score r in Eq. (3): For all tasks except CoLA and the “neutral” label of MNLI, the top-N ones of each class are selected; for CoLA, the top-N ones are used as the training sample as linguistically acceptable sequences, and the bottom-N ones are as linguistically unacceptable sequences; for the “neutral” label of MNLI, we find it better to randomly select N samples from

Table 8: Hyperparameters for generating training data of different tasks. τ: Temperature during sampling ($\tau = 0$ means using greedy sampling); α and β: Repetition rewarding/penalizing parameters; M: Number of total generated texts per label. The top-k sampling (if $\tau > 0$) uses $k = 10$.

Task	Label	τ	α	β	M
SST-2	positive	0.2	-	1.2	25,000
	negative		-	1.2	25,000
CoLA	grammatical	[0.1, 10]	-	1.2	10,000
	not grammatical		-	1.2	10,000
MNLI	entailment	0	0.8	1.1	25,000
	neutral		1.3	1.3	25,000
	contradiction		1.1	1.1	25,000
QNLI	entailment	0	0.9	1.2	25,000
	not entailment		0.9	1.2	25,000
RTE	entailment	0	0.8	1.1	30,000
	not entailment		1.1	1.1	30,000
MRPC	equivalent	0	0.8	1.1	30,000
	not equivalent		1.1	1.1	30,000
QQP	equivalent	0	1.0	1.2	25,000
	not equivalent		1.2	1.2	25,000

Table 9: Hyperparameters used for fine-tuning on different tasks (they are kept same for all tasks). Fine-tuning-related hyperparameters (*e.g.*, learning rate, batch size) follow the default values (when the validation set is not available) in Appendix A of [13]; regularization-related hyperparameters follow the default values in label smoothing and temporal ensembling. lr: Learning rate; bs: Batch size; $N|\mathcal{Y}|$: Total number of selected generated data (*i.e.*, training set size); B: Ensemble prediction update interval; T: Number of training steps; ϵ: Label smoothing parameter; γ: Temporal ensembling momentum parameter; δ: Threshold for filtering out noisy data; λ_{max}: Maximum weight (after ramp-up) of temporal ensembling regularization.

lr	bs	$N\|\mathcal{Y}\|$	B	T	ϵ	γ	δ	λ_{max}
1e-5	16	6,000	100	1,125	0.15	0.8	0.8	10

Table 10: Results with different generator/classifier PLMs.

PLMs	**MNLI-(m/mm)**	**SST-2**
G_θ: CTRL, C_ϕ: COCO-LM	$\mathbf{72.3}_{0.5}$/$\mathbf{73.8}_{0.5}$	$92.8_{0.6}$
G_θ: CTRL, C_ϕ: RoBERTa	$69.0_{0.8}$/$70.6_{0.9}$	$\mathbf{93.3}_{1.5}$
G_θ: GPT-2, C_ϕ: COCO-LM	$69.5_{1.2}$/$71.3_{1.3}$	$88.2_{1.8}$
G_θ: GPT-2, C_ϕ: RoBERTa	$68.3_{0.9}$/$69.7_{0.7}$	$88.6_{0.8}$

the total M samples instead of using the ranking score, probably because a neutral hypothesis with respect to the premise has a wide range of possibilities (*i.e.*, any hypothesis that is not entailed by or contradicts with the premise will be neutral), and random selection improves the diversity in generated hypotheses of the neutral label.

Hyperparameters for Fine-Tuning. Table 9 lists the hyperparameters used in the fine-tuning stage. We keep them same across all tasks except CoLA which uses $\delta = 0$ because half of the training data for CoLA are intentionally made to be of low quality (*i.e.*, as linguistically unacceptable sequences) and there is no need to filter them out. We follow [27] to slowly ramp-up λ in Equation (6) during the first 10 ensembles: $\lambda(t) = \lambda_{\text{max}} \exp(-5(1 - t/10)^2)$ where t is the number of prediction ensembles performed.

Computation Environment. All experiments are conducted on NVIDIA GeForce RTX 3090 GPUs. SuperGen can be run on typical research hardware (*e.g.*, with $>$ 10GB GPU memory). The generator PLM G_θ does not need to be trained so a relatively large generator can be used (*e.g.*, a 1.63B-parameter CTRL model).

D Using Different PLMs

The final performance is also relevant to the choice of PLMs as the generator/classifier. Apart from the default PLM choice, we report the results of using GPT-2_{XLarge} (1.54B parameters) [47] as the generator and $\text{RoBERTa}_{\text{Large}}$ (356M parameters) [33] as the classifier in Table 10 with everything else unchanged. When using GPT-2, we change the prompt used for SST-2 to "The film is bad/terrible/awful." for the negative label and "The film is good/great/excellent." for the positive label, since the original prompts used for SST-2 in Table 1 are a part of the control codes of CTRL and cannot be effectively leveraged by GPT-2. Overall, both CTRL and GPT-2 are able to generate quality training data for good fine-tuned classifier performance; CTRL consistently yields better results than GPT-2 regardless of the choice of the classifier PLM, probably because CTRL is pretrained with control codes which provide explicit guidance for generating texts of certain domains and attributes. We also observe that the generated text quality is strongly correlated to the generator's model size—using a smaller version of GPT-2 (*e.g.*, with 117M parameters) results in significantly less coherent texts and can hardly serve as training data. An interesting future direction is to try larger generator PLMs (*e.g.*, GPT-3) which may create training data of better quality.

Table 11: Example generated texts for SST-2, MNLI and QQP. *Sampled sequences* from pretraining corpus ($\boldsymbol{x}^s$) are italicized; generated sequences ($\boldsymbol{x}^g$) are underlined; **prompts** ($\boldsymbol{w}^y$) are in bold.

Task	Label	Generated Text
SST-2	positive	**Rating: 5.0** The film is a great example of the kind of movie that you can watch over and over.
	negative	**Rating: 1.0** The film was a total waste of time. I would not recommend this movie to anyone.
MNLI	entailment	*The construction of the station began in 2016, when the opening date was estimated to mid-2020.* **In other words**, the station was to open in 2020.
	neutral	*The construction of the station began in 2016, when the opening date was estimated to mid-2020.* **Furthermore**, it is expected that a new bus terminal will be built at this station.
	contradiction	**There is a rumor that** *The construction of the station began in 2016, when the opening date was estimated to mid-2020.* **However, the truth is:** The construction started in 2017, and the official opening date was set for March 31, 2018.
QQP	equivalent	*What are the most wear resistant steels?* **In other words,** what are the most durable steels?
	not equivalent	*What are the most wear resistant steels?* **Furthermore,** what is the best way to clean them?

Table 12: Negative example generated texts for MNLI that do not pertain to the desired label. *Sampled sequences* from pretraining corpus ($\boldsymbol{x}^s$) are italicized; generated sequences ($\boldsymbol{x}^g$) are underlined; **prompts** ($\boldsymbol{w}^y$) are in bold. In example #1, the generated text corresponding to the "entailment" label actually pertains to the "neutral" label. In example #2, the generated text corresponding to the "contradiction" label actually pertains to the "neutral" label. In example #3, the generated text corresponding to the "neutral" label actually pertains to the "contradition" label.

Example	Label	Generated Text
#1	entailment	*Directed by Daniel J. Sullivan, the cast features Kelly AuCoin, Annie Parisse, Lee Tergesen and Alex Wolff.* **In other words**, the cast features a lot of people who are not in the original Broadway cast.
	neutral	*Directed by Daniel J. Sullivan, the cast features Kelly AuCoin, Annie Parisse, Lee Tergesen and Alex Wolff.* **Furthermore**, it was announced that a sequel to "The Last Airbender" will be released in 2019.
	contradiction	**There is a rumor that** *Directed by Daniel J. Sullivan, the cast features Kelly AuCoin, Annie Parisse, Lee Tergesen and Alex Wolff.* **However, the truth is:** The cast includes only two actors from the original Broadway production.
#2	entailment	*As a result, minor problems begin to look serious and rare events begin to seem common.* **In other words**, a minor problem can begin to look serious and rare events seem common.
	neutral	*As a result, minor problems begin to look serious and rare events begin to seem common.* **Furthermore**, the number of people who are affected by these changes is increasing.
	contradiction	**There is a rumor that** *As a result, minor problems begin to look serious and rare events begin to seem common.* **However, the truth is:** The only thing that can cause such an effect is a major change in the weather.
#3	entailment	*The character was announced on 20 September 2019, but casting wasn't revealed until 10 December 2019.* **In other words**, the casting wasn't revealed until 10 December 2019.
	neutral	*The character was announced on 20 September 2019, but casting wasn't revealed until 10 December 2019.* **Furthermore**, it is unknown whether the role will be played by a male or female actor.
	contradiction	**There is a rumor that** *The character was announced on 20 September 2019, but casting wasn't revealed until 10 December 2019.* **However, the truth is:** The character was announced on 19 October 2018.

E Case Studies

We present concrete examples of generated texts guided by prompts of different labels in Table 11. The generated sequences are not only coherent, but also pertain to the corresponding labels. For easier tasks like SST-2, the generated texts almost always correctly reflect the desired sentiment polarity specified by the prompt. For more difficult tasks like MNLI, sometimes the generated texts are not of the correct label. Table 12 demonstrates more concrete examples where the generated texts may not correctly pertain to the label of the prompt. The existence of such label noise motivates our use of the regularization techniques in the fine-tuning stage.

We believe that larger generator PLMs (*e.g.*, GPT-3 [5]) can bring about better text generation quality and improve the accuracy in producing texts that pertain to the desired class. Furthermore, better filtering strategies can be developed in the future to select training data with the correct labels.

F Knowledge Distillation Baseline Details

We show the concrete prompts used for the knowledge distillation baseline in Tables 13 and 14 on MNLI and SST-2, respectively. We use the best prompt (prompt # 1 in both tables) out of the three according to the zero-shot test set prediction accuracy for generating soft labels to train the classification model (*i.e.*, knowledge distillation). The classifier is trained with Kullback–Leibler

Table 13: Different prompts used on MNLI for CTRL zero-shot prompting and knowledge distillation baselines. $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ denote the first and second input sequence, respectively.

Prompt	Template	Label name
# 1	Sentence 1: $\boldsymbol{x}_1$ Sentence 2: $\boldsymbol{x}_2$ Does Sentence 1 entail Sentence 2? The answer is:	entailment: Yes neutral: Maybe contradiction: No
# 2	Premise: $\boldsymbol{x}_1$ Hypothesis: $\boldsymbol{x}_2$ Does the premise entail the hypothesis? Options: Yes. No. Maybe. The answer is:	entailment: Yes neutral: Maybe contradiction: No
# 3	Premise: $\boldsymbol{x}_1$ Hypothesis: $\boldsymbol{x}_2$ What is the relation between the premise and the hypothesis? Options: Entailment. Neutral. Contradiction. The answer is:	entailment: Entailment neutral: Neutral contradiction: Contradiction

Table 14: Different prompts used on SST-2 for CTRL zero-shot prompting and knowledge distillation baselines. $\boldsymbol{x}$ denotes the input sequence.

Prompt	Template	Label name
# 1	$\boldsymbol{x}$ This is	positive: good; negative: bad
# 2	$\boldsymbol{x}$ It was	positive: good; negative: bad
# 3	Review: $\boldsymbol{x}$ Sentiment:	positive: Positive; negative: Negative

(KL) divergence as the objective to approximate the soft labels generated by CTRL on the entire training set.