Tuning Language Models as Training Data Generators for Augmentation-Enhanced Few-Shot Learning

Yu Meng [1] **Martin Michalski** [1] **Jiaxin Huang** [1] **Yu Zhang** [1] **Tarek Abdelzaher** [1] **Jiawei Han** [1]

Abstract

Recent studies have revealed the intriguing few-shot learning ability of pretrained language models (PLMs): They can quickly adapt to a new task when fine-tuned on a small amount of labeled data formulated as prompts, without requiring abundant task-specific annotations. Despite their promising performance, most existing few-shot approaches that only learn from the small training set still underperform fully supervised training by nontrivial margins. In this work, we study few-shot learning with PLMs from a different perspective: We first tune an autoregressive PLM on the few-shot samples and then use it as a generator to synthesize a large amount of novel training samples which augment the original training set. To encourage the generator to produce label-discriminative samples, we train it via weighted maximum likelihood where the weight of each token is automatically adjusted based on a discriminative meta-learning objective. A classification PLM can then be fine-tuned on both the few-shot and the synthetic samples with regularization for better generalization and stability. Our approach FewGen achieves an overall better result across seven classification tasks of the GLUE benchmark than existing few-shot learning methods, improving no-augmentation methods by $5+$ average points, and outperforming augmentation methods by $3+$ average points.

1. Introduction

Recent research has demonstrated the appealing few-shot learning potential of pretrained language models (PLMs) (Brown et al., 2020; Clark et al., 2020; Devlin et al., 2019; He et al., 2021; Liu et al., 2019; Meng et al., 2021a; 2022b) on natural language understanding (NLU) tasks (Wang et al., 2019; 2018): Instead of relying on abundant task-specific annotations, PLMs can effectively leverage a small set of training samples to quickly learn a new task. Such training data efficiency is usually achieved by formulating downstream tasks as prompts (Brown et al., 2020; Gao et al., 2021; Scao & Rush, 2021; Schick & Schütze, 2021a;d), allowing the PLM to adapt its language modeling ability acquired through pretraining to downstream tasks.

The success of prompt-based methods has stimulated numerous explorations along the line of effective few-shot learning with PLMs: The training samples converted to natural language prompts can be used to directly fine-tune PLMs (Gao et al., 2021; Schick & Schütze, 2021a) or as in-context demonstrations to facilitate better inference (Liu et al., 2022b; Min et al., 2022b). Recent approaches aim to automate the design of prompts by gradient-based searching (Shin et al., 2020) or parameterizing prompts as continuous learnable embeddings (Lester et al., 2021; Zhang et al., 2022; Zhong et al., 2021). Other studies investigate and address specific issues in prompt-based few-shot learning (Liu et al., 2022a; Tam et al., 2021; Zhao et al., 2021). While remarkable, the model performance still has a nontrivial gap from fully supervised models trained on massive labeled data. Indeed, training deep models is inherently data demanding—model generalization usually benefits from more training samples (Baum & Haussler, 1988).

In this work, we study few-shot learning with PLMs from a different perspective: Instead of proposing new methods for fine-tuning on few-shot samples, we focus on the generation of quality training data based on few-shot samples and using these synthesized training samples to fine-tune the classification models. Motivated by the strong text generation power of autoregressive PLMs (Brown et al., 2020; Keskar et al., 2019; Raffel et al., 2019), a few previous studies enlarge the training set by generating new texts as training samples. They either fine-tune the generator on the initial training set with the standard maximum likelihood objective (Anaby-Tavor et al., 2020; Kumar et al., 2020) or use the training samples as demonstrations (Yoo et al., 2021). However, these methods do not explicitly model the distinction across different labels and may struggle to

[1]University of Illinois Urbana-Champaign. Correspondence to: Yu Meng <yumeng5@illinois.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

generate accurate training samples pertaining to the desired labels for challenging NLU tasks.

In this paper, we explore how to effectively use few-shot samples to tune PLMs for generating high quality label-discriminative training samples. Our contributions are as follows: (1) We analyze the issues of using standard maximum likelihood for tuning the generator and propose a meta-weighted maximum likelihood objective by automatically learning token weights that emphasize label discriminativeness. (2) We propose a simple and effective training procedure for fine-tuning classification PLMs on generated data by mitigating label noise. (3) Under the same few-shot learning setting, our method FewGen outperforms existing methods by 3+ average points on seven classification tasks of the GLUE benchmark (Wang et al., 2018). Ablation studies validate the effectiveness of our proposed meta-weighted training objective and classifier fine-tuning method.[1]

2. Related Work

Few-Shot Learning with PLMs. Few-shot learning has gained much attention recently due to its minimal resource assumption—Without requiring massive annotated data but only leveraging a few training samples (*e.g.*, 16 per label), few-shot methods can be widely adopted in many practical scenarios where obtaining large-scale annotations is unaffordable. Standard fine-tuning of PLMs for few-shot learning usually performs poorly because the limited training samples may not be sufficient for optimizing the parameters in the newly introduced classification head. To reuse the language modeling ability of PLMs without introducing randomly initialized parameters, prompt-based approaches (Brown et al., 2020; Gao et al., 2021; Hu et al., 2022; Logan IV et al., 2021; Min et al., 2022a; Schick & Schütze, 2021a;b;d; Tam et al., 2021) formulate training samples as natural language prompt templates so that various downsteam tasks can be solved as a token prediction problem. They enjoy improved training data efficiency over standard fine-tuning in low-data regimes (Scao & Rush, 2021) and achieve remarkable few-shot learning performance. Later developments in prompt-based methods replace the manual design of prompt templates with automatic search or learning (Cui et al., 2022; Hambardzumyan et al., 2021; Lester et al., 2021; Liu et al., 2021b; Zhang et al., 2022; Zhong et al., 2021). There are also studies focusing on specific issues (Liu et al., 2022a; Tam et al., 2021; Zhao et al., 2021) in prompt-based methods. Instead of proposing fine-tuning methods for few-shot learning, we study how to generate quality training samples as augmentations by learning from the few-shot samples.

[1]Code can be found at `https://github.com/yumeng5/FewGen`.

Data Augmentation. Data augmentation methods (Chen et al., 2020; Huang et al., 2022; Lee et al., 2021; Meng et al., 2021b; Miyato et al., 2017; Xie et al., 2020) aim to create similar samples to the existing ones so that the enlarged training set can benefit model generalization. Early approaches simply use manually designed rules (*e.g.*, swapping or inserting tokens) for word-level alterations over the given samples to create new ones (Wei & Zou, 2019). Later methods leverage the strong generation power of PLMs to synthesize novel samples from scratch. Given a training set, the PLMs can be either fine-tuned on the labeled samples to learn label-conditioned generation probability (Kumar et al., 2020; Lee et al., 2021; Yang et al., 2020) or take the labeled data as demonstrations (Wang et al., 2021; Yoo et al., 2021) to generate similar samples pertaining to the same label. In this work, we study how to effectively tune generators on few-shot training data for creating new data—standard fine-tuning of PLMs on a small set of training data is prone to overfitting, and the resulting model may struggle to generate accurate, diverse and novel training data. We address this challenge by leveraging prefix-tuning and proposing a new meta-weighted generator tuning objective that emphasizes label-distinctive tokens.

Controlled Text Generation. Generating training samples for different labels can be viewed as a form of controlled text generation (Hu et al., 2017), whose goal is to generate textual contents of desired semantics, styles or attributes. Such control can be realized through different stages of PLM training and deployment: During pretraining, control codes (Keskar et al., 2019) can be used as explicit guidance for training the model to generate domain/attribute-specific texts; fine-tuning PLMs with attribute-specific data can also grant high-level control (*e.g.*, certain topics or sentiments (Ziegler et al., 2019)), fine-grained control (*e.g.*, specific words or phrases (Chan et al., 2021)) or both (Khalifa et al., 2021); at inference time, control over desired attributes can also be enforced without updating the PLM parameters (Dathathri et al., 2020; Krause et al., 2021; Kumar et al., 2021; Liu et al., 2021a; Pascual et al., 2021; Yang & Klein, 2021). More specifically related to the idea of generating training data with language models, early methods in text classification use bag-of-words or LSTM-based language models (Meng et al., 2018; 2019) to generate class-conditioned texts as training data. Recently, a few studies explore fine-tuning autoregressive PLMs (Anaby-Tavor et al., 2020; Yang et al., 2020) with the standard language modeling objective on the training set or using label-specific prompts (Gao et al., 2023; Meng et al., 2022a; Schick & Schütze, 2021c; Wang et al., 2021; Ye et al., 2022) to steer text generation towards the desired label. In this work, we analyze issues with directly tuning PLMs on few-shot samples with the standard maximum likelihood objective and propose a weighted variant of the objective that encourages

the PLM to focus on label-discriminative tokens.

Meta-Learning for Sample Weighting. The idea of weighting training samples in the loss calculation originates from the class imbalance (Wang et al., 2017) and noisy label (Hendrycks et al., 2018) learning scenarios—By assigning higher weights to the samples from minority classes or lower weights to the noisy samples, the learning process is less impacted by the imbalance/label noise issues. Meta-learning (Andrychowicz et al., 2016; Finn et al., 2017; Franceschi et al., 2018; Wu et al., 2018) is one way to automatically learn the weight for each sample. Specifically, a meta objective, usually defined as the loss on a clean unbiased validation set (Ren et al., 2018; Shu et al., 2019), can be used to learn the sample weights which become hyperparameters that control the optimization of model parameters. Our work has a different motivation and formulation of the meta objective for token-wise weighted training: Not all tokens in a training sample are equally label-discriminative. We thus design a meta objective to emphasize distinction across different labels (instead of using the validation loss as the meta objective) for learning the token weights.

3. Method

3.1. Preliminaries

Overview. We consider the strict few-shot learning setting (Perez et al., 2021): The training set $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}, y)_i\}$ consists of K training samples per label where $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$ is a text sequence with n tokens. The development set $\mathcal{D}_{\text{dev}}$ is of the same size as $\mathcal{D}_{\text{train}}$. There is no access to additional task-specific unlabeled data. The number of training samples K is assumed to be very small (*e.g.*, $K = 16$), making it challenging to train a classification model $C_{\boldsymbol{\phi}}$ that generalizes well to unseen data. To mitigate the training data scarcity issue, we first train an autoregressive PLM on $\mathcal{D}_{\text{train}}$, and then use it as a generator $G_{\boldsymbol{\theta}}$ to synthesize more novel samples $\mathcal{D}_{\text{gen}} = \{(\tilde{\boldsymbol{x}}, \tilde{y})_i\}$ that augment the original training set. Finally, a classification PLM $C_{\boldsymbol{\phi}}$ is fine-tuned on both $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{gen}}$ to perform the task. An overview of FewGen is shown in Fig. 1.

Text Generation with Autoregressive PLMs. In standard fine-tuning for text generation, an autoregressive PLM $G_{\boldsymbol{\theta}}$ is trained via the maximum likelihood generation loss of each token in a sequence $\boldsymbol{x}$ conditioned on previous tokens:

$$\min_{\boldsymbol{\theta}} -\frac{1}{n}\sum_{j=1}^{n} \log p_{\boldsymbol{\theta}}(x_j|\boldsymbol{x}_{<j}),$$

$$p_{\boldsymbol{\theta}}(x_j|\boldsymbol{x}_{<j}) = \frac{\exp(\boldsymbol{e}_j^\top \boldsymbol{h}_j)}{\sum_{j'=1}^{|V|}\exp(\boldsymbol{e}_{j'}^\top \boldsymbol{h}_j)}.$$

where the token generation probability $p_{\boldsymbol{\theta}}(\cdot)$ is usually parameterized using token embeddings $\boldsymbol{e}$ and hidden states $\boldsymbol{h}$ of a Transformer (Vaswani et al., 2017) model. After training, $G_{\boldsymbol{\theta}}$ can be used to generate novel texts by iteratively sampling tokens from its generation probability distribution.

Prefix-Tuning. Unlike fine-tuning which updates all model parameters $\boldsymbol{\theta}$ of a PLM, prefix-tuning (Li & Liang, 2021) freezes all pretrained Transformer parameters and only optimizes prefix vectors $\boldsymbol{\theta}_p$ that are prepended to each Transformer layer. We use prefix-tuning for training $G_{\boldsymbol{\theta}_p}$ on $\mathcal{D}_{\text{train}}$ because (1) it offers better effectiveness than fine-tuning for small datasets (Li & Liang, 2021) and (2) the generation models for different labels can share the same backbone Transformer parameters with only the prefix vectors being different, significantly reducing the memory requirement for multi-class classification tasks.

3.2. Label-Discriminative Text Generator Tuning

Motivation. To model the conditional text generation probability $p(\boldsymbol{x}|y_l)$ on different labels, a straightforward way is to parameterize a generation model $G_{\boldsymbol{\theta}_{p_l}}$ for each label y_l via a set of prefix vectors $\boldsymbol{\theta}_p = \{\boldsymbol{\theta}_{p_l}\}|_{l=1}^{L}$ so that $p(\boldsymbol{x}|y_l) = p_{\boldsymbol{\theta}_{p_l}}(\boldsymbol{x})$, and then tune $\boldsymbol{\theta}_{p_l}$ on the training samples $\boldsymbol{x}$ with label y_l:

$$\min_{\boldsymbol{\theta}_{p_l}} \mathcal{L}_{\text{gen}}, \quad \mathcal{L}_{\text{gen}}(\boldsymbol{\theta}_{p_l}) = -\frac{1}{n}\sum_{j=1}^{n} \log p_{\boldsymbol{\theta}_{p_l}}(x_j|\boldsymbol{x}_{<j}). \quad (1)$$

However, such an approach only optimizes the *generative* likelihood $p(\boldsymbol{x}|y_l)$ without accounting for *label discriminativeness* $p(y_l|\boldsymbol{x})$ which is essential for generating unambiguous training samples to benefit the final classification task. Challenging NLU tasks can have largely similar distributions across different labels, with very nuanced differences reflected by a few key tokens. For example, a negative review text "*a movie where the ending feels like a cop-out*" may immediately become a positive one by just changing the last word "cop-out" to "revelation". Indeed, we find that such subtle distinctions over different labels may not be effectively captured using the standard generation objective in Eq. (1) where each token contributes *equally* to the overall loss. As shown in Fig. 2, a discriminative loss $\mathcal{L}_{\text{disc}}$ (defined in Eq. (2)) can even increase during training—It is possible that the dominating patterns in the training samples are label-*indiscriminate* (*e.g.*, a movie review dataset may frequently mention "the movie"), making the generators of different labels eventually converge to similar distributions, especially when there are limited training samples per label.

To promote the generation of label-discriminative texts, we encourage each token x_j to be more likely generated under the corresponding label y_l instead of other labels (*i.e.*, maximize $p_{\boldsymbol{\theta}_{p_l}}(x_j|\boldsymbol{x}_{<j})$ and minimize $p_{\boldsymbol{\theta}_{p_{l'}}}(x_j|\boldsymbol{x}_{<j})$ for $l' \neq l$)

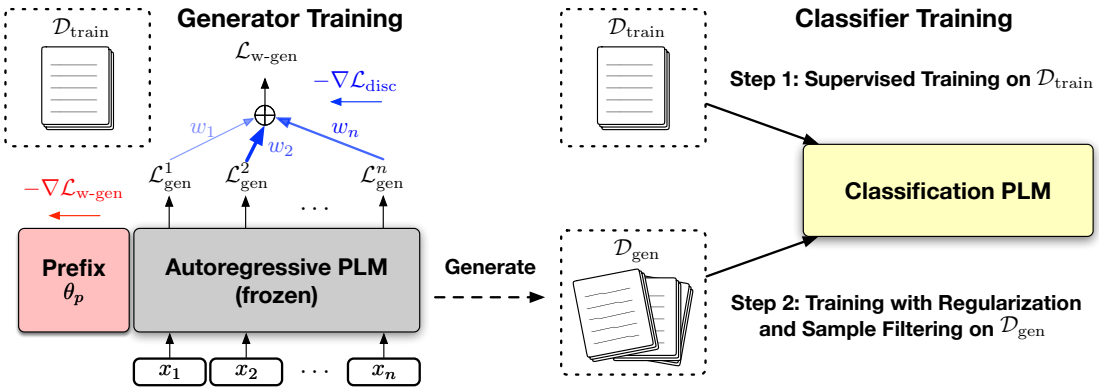
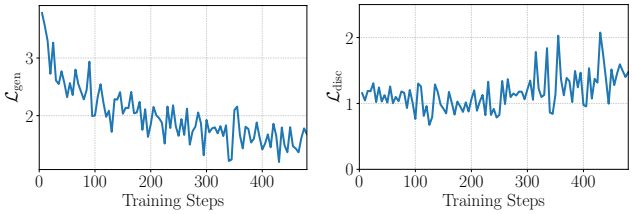


Figure 1: Overview of FewGen. A generator PLM is first tuned on the few-shot samples with our proposed meta-weighted training objective and then used to synthesize new training samples. A classification PLM is finally trained on both the few-shot and the generated samples.



(a) $\mathcal{L}_{\text{gen}}$ during training (b) $\mathcal{L}_{\text{disc}}$ during training

Figure 2: (On MNLI) Training the generator via $\mathcal{L}_{\text{gen}}$ does not automatically decrease $\mathcal{L}_{\text{disc}}$.

via a discriminative loss $\mathcal{L}_{\text{disc}}$:

$$\mathcal{L}_{\text{disc}}(\boldsymbol{\theta}_p) = -\frac{1}{n}\sum_{j=1}^{n}\mathcal{L}^j_{\text{disc}}(\boldsymbol{\theta}_p),$$
$$\mathcal{L}^j_{\text{disc}}(\boldsymbol{\theta}_p) = \frac{p_{\boldsymbol{\theta}_{p_l}}(x_j|\boldsymbol{x}_{<j})}{\sum_{l'=1}^{L} p_{\boldsymbol{\theta}_{p_{l'}}}(x_j|\boldsymbol{x}_{<j})}. \quad (2)$$

Although one can directly combine $\mathcal{L}_{\text{disc}}$ with $\mathcal{L}_{\text{gen}}$ to train $G_{\boldsymbol{\theta}_p}$ to enforce distinction across different labels, doing so will result in two undesirable consequences: (1) A hyperparameter needs to be introduced to balance the weights of the two losses, whose optimal value is likely to vary by task; and (2) directly updating generator parameters with the discriminative loss $\mathcal{L}_{\text{disc}}$ will worsen the language modeling quality of the generator, making it prone to generating less fluent and coherent texts after training.

Weighted Maximum Likelihood Generator Tuning. To preserve the generative learning of $G_{\boldsymbol{\theta}_p}$ while emphasizing label-discriminative tokens, we assume each token is associated with a weight in the maximum likelihood loss. Intuitively, when our goal is to generate distinctive texts across different labels as training samples, not all tokens should contribute equally to generator training. For example, for sentiment classification tasks, one would expect "good/bad" to be more label-discriminative than "the movie", and the former should be paid more attention to during training. It is thus natural to generalize $\mathcal{L}_{\text{gen}}$ in Eq. (1) to $\mathcal{L}_{\text{w-gen}}$ as follows by assuming a weight w_j is given for each token.

$$\min_{\boldsymbol{\theta}_{p_l}} \mathcal{L}_{\text{w-gen}}, \quad \mathcal{L}_{\text{w-gen}}(\boldsymbol{\theta}_{p_l};\boldsymbol{w}) = -\sum_{j=1}^{n} w_j \mathcal{L}^j_{\text{gen}}(\boldsymbol{\theta}_{p_l}), \quad (3)$$
$$\mathcal{L}^j_{\text{gen}}(\boldsymbol{\theta}_{p_l}) = \log p_{\boldsymbol{\theta}_{p_l}}(x_j|\boldsymbol{x}_{<j}).$$

Note that in $\mathcal{L}_{\text{w-gen}}$, $\boldsymbol{w}$ is assumed to be the *hyperparameter* under which $\boldsymbol{\theta}_{p_l}$ is optimized. When w_j is the same for every token, Eq. (3) will be equivalent to Eq. (1). While it is possible to manually design weighting rules for setting $\boldsymbol{w}$ to promote label-discriminative learning, they will likely necessitate task-specific knowledge and nontrivial tuning. To facilitate the automatic learning of these weights $\boldsymbol{w}$, we propose to parameterize them as learnable hyperparameters using the idea of meta-learning.

Meta Weight Learning Setup. To automatically learn token weights as hyperparameters, we formulate a bi-level optimization problem using the idea of meta-learning. The inner objective $\mathcal{L}_{\text{w-gen}}$ optimizes the generator parameters $\boldsymbol{\theta}_p$ given the token weights w_j:

$$\mathcal{L}_{\text{w-gen}}(\boldsymbol{\theta}_p;\boldsymbol{\omega}) = -\sum_{j=1}^{n} w_j(\boldsymbol{\omega})\mathcal{L}^j_{\text{gen}}(\boldsymbol{\theta}_p),$$
$$\boldsymbol{\theta}^*_p(\boldsymbol{\omega}) = \operatorname*{argmin}_{\boldsymbol{\theta}_p} \mathcal{L}_{\text{w-gen}},$$

where the token weights $w_j(\boldsymbol{\omega})$ are parameterized and learned via a weighting network $g_{\boldsymbol{\omega}}$ (details about its implementation are in Appendix A). The weighting network

Algorithm 1 Meta-Weighted Generator Tuning.

Input: $\mathcal{D}_{\text{train}}$: Few-shot training set.
Parameter: T: Number of training steps.
Output: $\boldsymbol{\theta}_p$: Prefix parameters for all labels.
Initialize $\boldsymbol{\theta}_p^{(0)}$ (with task-descriptive prompts) and $\boldsymbol{\omega}^{(0)}$
for $t \in [0, 1, \ldots, T-1]$ **do**
  $\mathcal{B} \leftarrow$ Sample a minibatch from $\mathcal{D}_{\text{train}}$
  $\hat{\boldsymbol{\theta}}_p^{(t)}\left(\boldsymbol{\omega}^{(t)}\right) \leftarrow$ Take one gradient step to descend $\mathcal{L}_{\text{w-gen}}\left(\boldsymbol{\theta}_p^{(t)}; \boldsymbol{\omega}^{(t)}\right)$ on $\mathcal{B}$
  $\boldsymbol{\omega}^{(t+1)} \leftarrow$ Take one gradient step to descend $\mathcal{L}_{\text{disc}}\left(\hat{\boldsymbol{\theta}}_p^{(t)}\left(\boldsymbol{\omega}^{(t)}\right)\right)$ on $\mathcal{B}$
  $\boldsymbol{\theta}_p^{(t+1)} \leftarrow$ Take one gradient step to descend $\mathcal{L}_{\text{w-gen}}\left(\boldsymbol{\theta}_p^{(t)}; \boldsymbol{\omega}^{(t+1)}\right)$ on $\mathcal{B}$
end
return $\boldsymbol{\theta}_p = \boldsymbol{\theta}_p^{(T)}$

parameters $\boldsymbol{\omega}$ are trained with an outer objective $\mathcal{L}_{\text{disc}}$:

$$\mathcal{L}_{\text{disc}}(\boldsymbol{\theta}_p^*(\boldsymbol{\omega})) = -\frac{1}{n}\sum_{j=1}^{n}\mathcal{L}_{\text{disc}}^j(\boldsymbol{\theta}_p^*(\boldsymbol{\omega})), \quad \boldsymbol{\omega}^* = \operatorname*{argmin}_{\boldsymbol{\omega}} \mathcal{L}_{\text{disc}}. \tag{4}$$

Under the above bi-level optimization formulation, the discriminative loss $\mathcal{L}_{\text{disc}}$ is not used to directly update generator parameters, but to automatically learn token weights that are used as hyperparameters by the inner objective $\mathcal{L}_{\text{w-gen}}$. As the token weights are trained to minimize $\mathcal{L}_{\text{disc}}$, the generator focuses more on label-discriminative tokens.

We use an online optimization strategy (Shu et al., 2019) instead of nested optimization loops to optimize $\boldsymbol{\omega}^*$ and $\boldsymbol{\theta}_p^*$ for training efficiency. It also guarantees convergence to the critical points of both $\mathcal{L}_{\text{w-gen}}$ and $\mathcal{L}_{\text{disc}}$ under mild conditions. We initialize the prefix parameters $\boldsymbol{\theta}_p$ using natural language prompts, and the details can be found in Appendix B. The overall training procedure is shown in Algorithm 1.

Analysis of Meta Weight Learning. To study how the token weights are learned during training, we analyze the gradients of the weighting network parameters $\boldsymbol{\omega}$ which are optimized via Eq. (4) (detailed derivation in Appendix C):

$$-\left.\frac{\partial \mathcal{L}_{\text{disc}}\left(\hat{\boldsymbol{\theta}}_p^{(t)}(\boldsymbol{\omega})\right)}{\partial \boldsymbol{\omega}}\right|_{\boldsymbol{\omega}=\boldsymbol{\omega}^{(t)}} \propto \sum_{j=1}^{n} d_j \left.\frac{\partial w_j(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}}\right|_{\boldsymbol{\omega}=\boldsymbol{\omega}^{(t)}},$$

$$d_j = \left.\frac{\partial \mathcal{L}_{\text{disc}}\left(\hat{\boldsymbol{\theta}}_p\right)}{\partial \hat{\boldsymbol{\theta}}_p}\right|_{\hat{\boldsymbol{\theta}}_p=\hat{\boldsymbol{\theta}}_p^{(t)}} \left.\frac{\partial \mathcal{L}_{\text{gen}}^j(\boldsymbol{\theta}_p)}{\partial \boldsymbol{\theta}_p}\right|_{\boldsymbol{\theta}_p=\boldsymbol{\theta}_p^{(t)}}^{\top}.$$

Algorithm 2 Classifier fine-tuning on $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{gen}}$.

Input: $\mathcal{D}_{\text{train}}$: Few-shot training set; $\mathcal{D}_{\text{gen}}$: Synthesized training set.
Parameter: T: Number of training steps.
Output: $\boldsymbol{\phi}$: Trained classification model parameters.
$\boldsymbol{\phi}^{(0)} \leftarrow$ Train on $\mathcal{D}_{\text{train}}$ with standard supervised learning
$\bar{\boldsymbol{z}} \leftarrow \mathbf{0}$ `// Initialize ensembled prediction`
for $t \in [0, 1, \ldots, T-1]$ **do**
  $\mathcal{B} \leftarrow$ Sample a minibatch from $\mathcal{D}_{\text{gen}}$
  $\boldsymbol{\phi}^{(t+1)} \leftarrow$ Take one gradient step to descend $\mathcal{L}_{\text{class}}$ in Eq. (5) on $\mathcal{B}$
  $\bar{\boldsymbol{z}} \leftarrow$ Accumulate the current model prediction
  Update $\mathcal{D}_{\text{gen}}$ to exclude noisy samples based on $\bar{\boldsymbol{z}}$
end
return $\boldsymbol{\phi} = \boldsymbol{\phi}^{(T)}$

It can be seen that the gradient descent direction of $\boldsymbol{\omega}$ is determined by a sum of token weight gradient ascent directions (*i.e.*, $\frac{\partial w_j(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}}$) weighted by a scalar d_j, where d_j characterizes the similarity between the gradient of the discriminative objective and the gradient of the generative objective on the jth token. Therefore, the meta weights will be higher on those tokens where optimizing their generative objective is more beneficial for minimizing the discriminative objective, so that label-distinctive information is better emphasized.

3.3. Classifier Fine-Tuning

With the trained generator $G_{\boldsymbol{\theta}_p}$, we can synthesize novel training samples $\mathcal{D}_{\text{gen}}$ that augment $\mathcal{D}_{\text{train}}$ for fine-tuning a classification PLM $C_{\boldsymbol{\phi}}$. The major challenge to effectively leverage $\mathcal{D}_{\text{gen}}$ is that the label noise (*i.e.*, some generated samples may not accurately pertain to the corresponding label) may deteriorate model performance if standard supervised learning is directly used. We propose a simple noise-robust training procedure to improve the generalization and stability of training: First fine-tune $C_{\boldsymbol{\phi}}$ on $\mathcal{D}_{\text{train}}$ with standard supervised training, and then continue fine-tuning it on $\mathcal{D}_{\text{gen}}$ by applying *label smoothing* (Szegedy et al., 2016) and *temporal ensembling* (Laine & Aila, 2017) as regularization, following (Meng et al., 2022a). Specifically, given a training sample $(\tilde{\boldsymbol{x}}, \tilde{y}) \in \mathcal{D}_{\text{gen}}$, we minimize the following classification loss:

$$\mathcal{L}_{\text{class}}(\boldsymbol{\phi}) = -\sum_{l=1}^{L} q_l \log(p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{x}})_l) - \lambda \sum_{l=1}^{L} \bar{z}_l \log \frac{p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{x}})_l}{\bar{z}_l}, \tag{5}$$

where $q_l = \mathbb{1}(l = \tilde{y})(1-\epsilon) + \epsilon/L$ and ϵ is the label smoothing weight; $p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{x}})$ is the model prediction on $\tilde{\boldsymbol{x}}$; λ is a regularization weight for temporal ensembling; and $\bar{\boldsymbol{z}}$ is the accumulated moving-average model predictions. We also use the ensembled prediction $\bar{\boldsymbol{z}}$ to filter out noisy synthesized samples: We only include those samples for training

where $\bar{\boldsymbol{z}}$ strongly agrees with the label $\tilde{y}$ (*i.e.*, $\bar{z}_{\tilde{y}} > \delta$ where $\delta > 0$ is a threshold parameter). In Eq. (5), the first classification term is the cross-entropy loss with smoothed labels; the second regularization term corresponds to temporal ensembling, which requires the current model prediction to be close to its past accumulated predictions. This not only neutralizes the fluctuation in model predictions for better training stability when label noise is present (Nguyen et al., 2020) but also helps prevent catastrophic forgetting (Kirkpatrick et al., 2017) of the information learned previously from the few-shot training set $\mathcal{D}_{\text{train}}$. Please refer to Appendix B for details about the temporal ensembling implementation. The overall procedure of classifier fine-tuning is summarized in Algorithm 2.

4. Experimental Setup

Downstream Tasks and Metrics. We conduct evaluation on all tasks of the GLUE benchmark (Wang et al., 2018) (more details in Appendix D) except STS-B which is a regression task. We follow the same data split and evaluation protocol as (Gao et al., 2021): Both $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{dev}}$ contain 16 samples per label and are sampled from the original training set with 5 different random seeds. The original development sets are used for testing. For all reported results, we include the average and standard deviation over the 5 different $\mathcal{D}_{\text{train}}/\mathcal{D}_{\text{dev}}$ splits. F1 score is used as the metric for QQP and MRPC, Matthews correlation for CoLA, and accuracy for the remaining tasks.

Models and Training Settings. FewGen is a training data generation method and can be used with any fine-tuning method on any classification model. We use moderate-sized PLMs to ensure our results are reproducible on typical research hardware: CTRL (1.6B parameters) (Keskar et al., 2019) as the generator G_θ and RoBERTa$_{\text{Large}}$ (356M parameters) (Liu et al., 2019) as the classifier C_ϕ. We use prefix-tuning for training G_θ and prompt-based fine-tuning for training C_ϕ. For simplicity, we use the most basic manual prompt version of LM-BFF (Gao et al., 2021). The only exception is CoLA for which we use the standard fine-tuning since the input data might be out of the distribution of C_ϕ (Gao et al., 2021). The hyperparameter tuning is performed on $\mathcal{D}_{\text{dev}}$. More details are in Appendix B.

Compared Methods. No-augmentation baselines include zero-shot prompting, standard fine-tuning, in-context learning, and the following strong few-shot learning methods: Four versions of LM-BFF (Gao et al., 2021), P-Tuning (Liu et al., 2021b) and DART (Zhang et al., 2022). We also compare with data augmentation methods for few-shot learning: MixText (Chen et al., 2020), using back translation systems to generate paraphrases (UDA-style (Xie et al., 2020) augmentation), a few-shot demonstration method GPT3Mix (Yoo et al., 2021), and standard fine-tuning of generator on the few-shot samples with prompts. For fair comparisons, all augmentation methods use LM-BFF (Man.) to fine-tune a RoBERTa$_{\text{Large}}$ classifier. We also include the results of fully-supervised fine-tuning. More details about augmentation baselines are in Appendix E.

5. Evaluation

5.1. Main Results

We present the results of FewGen and baselines in Table 1. FewGen achieves overall better performance across the GLUE tasks, on average $5+$ points higher than the previous best few-shot method without augmentation, and $3+$ points better than GPT3Mix[2] (Yoo et al., 2021) which uses a 100 times larger generator model (175B) than FewGen.

Comparison with Back Translation. Using back translation to paraphrase the few-shot samples does not improve the results—this is probably because it does not produce samples that are sufficiently different from the few-shot training set. The success of UDA (Xie et al., 2020) is grounded in the augmentations from abundant unlabeled data that improve the classifier generalization. However, under the strict few-shot learning setup, there is no access to additional task-specific unlabeled data (Gao et al., 2021), making it challenging for paraphrase-based methods to create sufficiently diverse training samples only based on the small few-shot set. The new training samples produced by our FewGen method are not limited to the paraphrases of the few-shot samples, as the generator is trained via prefix-tuning to preserve the PLM's pretraining knowledge, based on which novel training samples can be synthesized.

Comparison with GPT3Mix. The gigantic size of GPT3 makes it challenging for tuning on few-shot samples. Therefore, GPT3Mix (Yoo et al., 2021) uses few-shot samples as demonstrations for creating the augmentations. Such an approach suffers from two limitations: (1) Without any parameter update to the PLM, its learning ability is not fully leveraged to adapt to the few-shot training set. (2) The PLM can only use a small subset of the few-shot samples at a time for creating each augmentation, as the number of demonstrations received by the model is bounded by its maximum input sequence length. This makes the quality of the created augmentations more sensitive to the randomly drawn training samples. Our FewGen method, on the other hand, can use the entire few-shot set for tuning the PLM and achieves overall even better classification results with a much smaller PLM ($< 1\%$ the size of the GPT3 model)

[2]The original GPT3Mix paper uses accuracy as the metric instead of Matthews correlation for CoLA; our reimplemented GPT3Mix achieves $79.4_{0.6}$ on CoLA if measured by accuracy.

Table 1: Results on seven classification tasks of the GLUE benchmark. We report average and standard deviation (as subscripts) performance over 5 different $\mathcal{D}_{\text{train}}/\mathcal{D}_{\text{dev}}$ splits defined in (Gao et al., 2021). †: Results from (Gao et al., 2021). ‡: Results from (Zhang et al., 2022). Methods that use additional models apart from the final classification model are marked.

Method	**MNLI-(m/mm)** (Acc.)	**QQP** (F1)	**QNLI** (Acc.)	**SST-2** (Acc.)	**CoLA** (Matt.)	**RTE** (Acc.)	**MRPC** (F1)	**AVG**
Methods without Augmentation: Few-shot samples are directly used for classifier tuning or as demonstrations for inference								
Prompting†	50.8/51.7	49.7	50.8	83.6	2.0	51.3	61.9	50.1
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
In-Context†	$52.0_{0.7}/53.4_{0.6}$	$36.1_{5.2}$	$53.8_{0.4}$	$84.8_{1.3}$	$-1.5_{2.4}$	$60.4_{1.4}$	$45.7_{6.0}$	47.4
LM-BFF (Man.)†	$68.3_{2.3}/70.5_{1.9}$	$65.5_{5.3}$	$64.5_{4.2}$	$92.7_{0.9}$	$9.3_{7.3}$	$69.1_{3.6}$	$74.5_{5.3}$	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
LM-BFF (Auto)† (w. 2.9B T5)	$68.3_{2.5}/70.1_{2.6}$	$67.0_{3.0}$	$68.3_{7.4}$	$92.3_{1.0}$	$14.0_{14.1}$	$\mathbf{73.9}_{2.2}$	$76.2_{2.3}$	65.8
+ demonstration† (w. 2.9B T5)	$70.0_{3.6}/72.0_{3.1}$	$67.7_{5.8}$	$68.5_{5.4}$	$93.0_{0.6}$	$21.8_{15.9}$	$71.1_{5.3}$	$78.1_{3.4}$	67.3
P-Tuning‡	$61.5_{2.1}/-$	$65.6_{3.0}$	$64.3_{2.8}$	$92.2_{0.4}$	–	–	$74.5_{7.6}$	–
DART‡	$67.5_{2.6}/-$	$67.8_{3.2}$	$66.7_{3.7}$	$93.5_{0.5}$	–	–	$78.3_{4.5}$	–
Methods with Augmentation: Few-shot samples are used for creating synthesized samples and for classifier tuning								
MixText	$65.1_{2.6}/66.2_{2.8}$	$60.6_{3.9}$	$68.4_{5.1}$	$89.1_{2.3}$	$12.8_{9.2}$	$66.5_{4.1}$	$64.6_{7.6}$	61.1
Back Translation (w. trained Marian)	$66.9_{4.6}/68.3_{3.8}$	$59.8_{4.6}$	$67.8_{4.9}$	$91.1_{1.9}$	$7.5_{3.7}$	$62.4_{5.3}$	$68.0_{11.2}$	60.6
GPT3Mix (w. 175B GPT3)	$61.5_{3.2}/62.6_{2.2}$	$70.4_{1.9}$	$69.2_{0.3}$	$\mathbf{93.6}_{0.6}$	$\mathbf{48.9}_{1.9}$	$70.4_{10.0}$	$69.9_{12.4}$	69.2
Generator Fine-Tuning (w. 1.6B CTRL)	$68.9_{5.1}/70.8_{5.3}$	$60.4_{8.7}$	$70.9_{4.1}$	$91.2_{1.2}$	$18.8_{10.0}$	$66.1_{4.4}$	$60.8_{15.4}$	62.6
FewGen (w. 1.6B CTRL)	$\mathbf{75.7}_{1.6}/\mathbf{77.1}_{1.0}$	$\mathbf{71.5}_{1.7}$	$\mathbf{76.3}_{4.4}$	$93.1_{0.8}$	$40.0_{7.5}$	$71.2_{2.4}$	$\mathbf{81.1}_{2.5}$	**72.8**
Fully Supervised Fine-Tuning†	*89.8/89.5*	*81.7*	*93.3*	*95.0*	*62.6*	*80.9*	*91.4*	*84.9*

Table 2: Ablation studies by removing (−) or switching (w.) one component of FewGen.

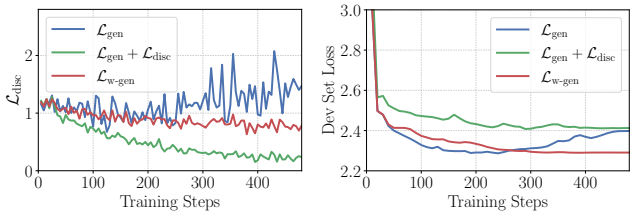
Method	**MNLI-(m/mm)**	**QQP**	**QNLI**	**SST-2**	**CoLA**	**RTE**	**MRPC**
FewGen	$75.7_{1.6}/77.1_{1.0}$	$71.5_{1.7}$	$76.3_{4.4}$	$93.1_{0.8}$	$40.0_{7.5}$	$71.2_{2.4}$	$81.1_{2.5}$
w. $\mathcal{L}_{\text{gen}}$	$74.9_{1.0}/76.2_{1.0}$	$70.7_{1.9}$	$75.0_{4.8}$	$92.5_{0.7}$	$37.8_{8.2}$	$69.5_{2.2}$	$80.8_{3.0}$
w. $\mathcal{L}_{\text{gen}} + \mathcal{L}_{\text{disc}}$	$74.6_{1.6}/76.0_{1.5}$	$68.8_{2.1}$	$76.1_{4.3}$	$92.4_{0.8}$	$41.2_{9.0}$	$70.1_{2.2}$	$79.6_{2.4}$
− label smooth	$75.0_{1.3}/76.2_{1.0}$	$71.1_{1.8}$	$76.5_{3.5}$	$92.7_{0.7}$	$39.3_{8.6}$	$69.4_{1.9}$	$81.3_{2.8}$
− temporal ensemble	$72.2_{2.5}/74.0_{2.2}$	$65.8_{2.1}$	$75.1_{2.7}$	$92.1_{1.7}$	$33.9_{4.4}$	$66.6_{2.4}$	$80.4_{3.2}$
w. fine-tune on $\mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{gen}}$	$68.9_{1.8}/70.6_{1.9}$	$64.3_{1.5}$	$71.1_{4.1}$	$91.8_{1.3}$	$34.0_{3.2}$	$59.6_{1.0}$	$80.4_{3.5}$

which can be deployed much more easily in practice.

5.2. Ablation Studies

The overall performance gain brought by FewGen over a no-augmentation counterpart can be seen by comparing FewGen with LM-BFF (Man.) which uses the same classifier and fine-tuning method on $\mathcal{D}_{\text{train}}$ only. We further analyze the effectiveness of each important component in FewGen via the following ablations: (1) Using the standard $\mathcal{L}_{\text{gen}}$ in Eq. (1) instead of our proposed $\mathcal{L}_{\text{w-gen}}$ in Eq. (3) for generator tuning (w. $\mathcal{L}_{\text{gen}}$); (2) using the directly combined $\mathcal{L}_{\text{gen}}$ and $\mathcal{L}_{\text{disc}}$ for generator tuning (w. $\mathcal{L}_{\text{gen}} + \mathcal{L}_{\text{disc}}$); (3) without applying label smoothing in Eq. (5) (− label smooth); (4) without applying temporal ensembling in Eq. (5) (− temporal ensemble); (5) directly fine-tuning the classification model on the combination of $\mathcal{D}_{\text{gen}}$ and $\mathcal{D}_{\text{train}}$ (w. fine-tune on $\mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{gen}}$)[3]. As shown in Table 2, (1) & (2) using the standard maximum likelihood loss or the combination of generative and discriminative losses to tune the generator both yield lower-quality training data and lead to degraded classification performance; (3) & (4) not applying regularization techniques for fine-tuning the classifier is more prone to label noise in the generated samples; (5) fine-tuning the classifier on the combination of $\mathcal{D}_{\text{gen}}$ and $\mathcal{D}_{\text{train}}$ significantly underperforms our two-step fine-tuning method.

[3]For this ablation, we upsample $\mathcal{D}_{\text{train}}$ by $\times 100$ so that its size is comparable with $\mathcal{D}_{\text{gen}}$; otherwise, the result is much worse.



(a) $\mathcal{L}_{\text{disc}}$ during training (b) Dev set loss during training

Figure 3: With different generator tuning objectives, (a) $\mathcal{L}_{\text{disc}}$ and (b) language modeling loss on the dev set.

5.3. Analyses of Loss Functions for Generator Tuning

As shown in Table 2, the choice of generator loss has a significant impact on the synthesized data quality and thus

Table 3: Quantitative evaluation of generator training objectives. We use two metrics: Generated data accuracy (Acc; higher is better) and generator's perplexity on the test set (PPL; lower is better). The results are averaged over 5 $\mathcal{D}_{\text{train}}/\mathcal{D}_{\text{dev}}$ splits.

Objective	MNLI		QQP		QNLI		SST-2		CoLA		RTE		MRPC	
	Acc. (↑)	PPL (↓)	Acc. (↑)	PPL (↓)	Acc. (↑)	PPL (↓)	Acc. (↑)	PPL (↓)	Acc. (↑)	PPL (↓)	Acc. (↑)	PPL (↓)	Acc. (↑)	PPL (↓)
$\mathcal{L}_{\text{gen}}$	69.4	13.1	87.5	10.9	57.0	23.4	91.5	43.8	59.1	85.6	82.9	9.3	87.6	5.0
$\mathcal{L}_{\text{gen}} + \mathcal{L}_{\text{disc}}$	70.2	13.5	87.3	11.2	57.2	24.8	92.0	49.5	59.2	87.0	82.8	9.6	86.3	5.3
$\mathcal{L}_{\text{w-gen}}$	**72.3**	**11.9**	**89.5**	**10.7**	**60.1**	**23.2**	**93.2**	**43.5**	**60.7**	**83.8**	**83.4**	**8.9**	**90.5**	**4.9**

Table 4: (For MNLI) Examples of generated second sequence (hypothesis) by generators tuned with three different objectives conditioned on a given first sequence (premise) "*In 2009, hair samples from 1,137 Taiji residents were tested for mercury by the National Institute for Minamata Disease (NIMD)*". The true label of the generated sequence is marked at the end of the sequence (if the generated sequence correctly pertains to the target label, it is marked in blue; otherwise, it is in red).

Objective	Label	Generated Second Sequence
$\mathcal{L}_{\text{gen}}$	entailment	The NIMD found that hair samples from 1,137 Taiji residents had mercury in their hair. *(neutral)*
	neutral	The NIMD found that there was no evidence of a link between exposure to high levels of mercury and thyroid cancer. *(neutral)*
	contradiction	There was no evidence of mercury in hair samples from Taiji. *(neutral)*
$\mathcal{L}_{\text{gen}} + \mathcal{L}_{\text{disc}}$	entailment	The number of hairs in a sample is equal to the number of people who lived in Taiji. *(neutral)*
	neutral	The results showed that there was no significant difference in levels of mercury. *(neutral)*
	contradiction	Hair samples from 1,137 Taiji residents were not tested. *(contradiction)*
$\mathcal{L}_{\text{w-gen}}$	entailment	The NIMD tested hair samples from 1,137 residents of Taiji. *(entailment)*
	neutral	There was no significant difference in levels between people who lived near a nickel mine and those living far away. *(neutral)*
	contradiction	The NIMD did not test any of the hair samples. *(contradiction)*

the final model performance. We conduct further analyses to compare the training processes of the generator under the following three loss functions and the resulting generated samples: (1) $\mathcal{L}_{\text{gen}}$ which is the standard language modeling loss; (2) $\mathcal{L}_{\text{gen}} + \mathcal{L}_{\text{disc}}$ which directly adds the discriminative loss to generator training; and (3) $\mathcal{L}_{\text{w-gen}}$ which is our meta-weighted objective. Fig. 3 shows the discriminative loss $\mathcal{L}_{\text{disc}}$ and the standard language modeling loss on the held-out development set throughout training. Although using $\mathcal{L}_{\text{gen}} + \mathcal{L}_{\text{disc}}$ helps reduce the discriminative loss, it comes at the cost of hindering language modeling—the generator loss on the development set is high. Using our meta-weighted objective $\mathcal{L}_{\text{w-gen}}$ not only encourages discriminativeness but also mitigates overfitting, yielding the lowest validation set loss. This is likely because the model receives contrastive information from other labels which facilitates more accurate modeling of the texts with the target label.

Quantitative Analyses. Apart from the final classification model performance which indirectly reflects the synthetic data quality, we additionally conduct more direct quantitative analyses of different generator training objectives. We use two metrics: (1) The accuracy of generated texts, which is judged by fully-supervised RoBERTa$_{\text{Large}}$ models fine-tuned on the original training sets of each task. We choose to adopt such an automatic evaluation instead of human evaluation because it is efficient and reliable—fully-supervised RoBERTa$_{\text{Large}}$ models have comparable or better accuracy than human baselines according to the GLUE benchmark[4]. (2) The generator's perplexity on the test sets, which reflects how well the generator models the task distribution. As shown in Table 3, using $\mathcal{L}_{\text{w-gen}}$ for generator training consistently outperforms using $\mathcal{L}_{\text{gen}}$ or $\mathcal{L}_{\text{gen}} + \mathcal{L}_{\text{disc}}$, both in generated text accuracy and in language modeling ability.

Comparing $\mathcal{L}_{\text{w-gen}}$ with $\mathcal{L}_{\text{gen}}$, the meta weights automatically learned emphasize discriminative tokens in generator training and help the generator capture subtle semantic differences across different labels, resulting in better language modeling quality and more distinctive synthetic data.

Comparing $\mathcal{L}_{\text{w-gen}}$ with $\mathcal{L}_{\text{gen}} + \mathcal{L}_{\text{disc}}$, the generator training objective is not directly impacted by the discriminative objective, thus avoiding the gradient interference issue in multi-task learning (Standley et al., 2019)—the gradient for optimizing the generative probability $p(\boldsymbol{x}|y_l)$ will be interfered by the gradient optimizing the discriminative probability $p(y_l|\boldsymbol{x})$ if $\mathcal{L}_{\text{gen}} + \mathcal{L}_{\text{disc}}$ is used. Therefore, using $\mathcal{L}_{\text{w-gen}}$ results in better language modeling quality and more fluent and coherent generation results.

Qualitative Analyses. We showcase concrete generation results for the three labels of MNLI by models trained with the three different loss functions in Table 4. The model trained with $\mathcal{L}_{\text{gen}}$ produces fluent and coherent sentences, but the generated sentences do not accurately pertain to

[4] https://gluebenchmark.com/leaderboard

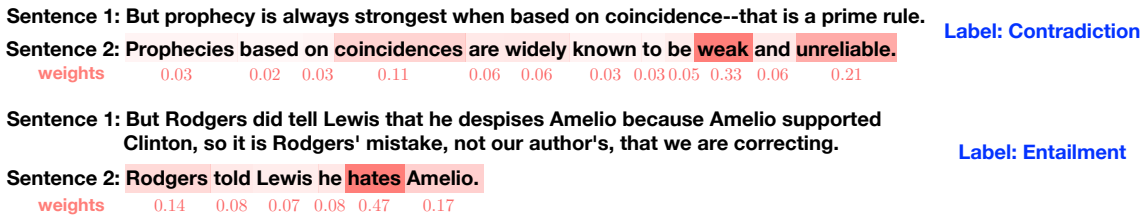


Figure 4: Visualization of learned token weights on two samples from MNLI's few-shot training set. The generator is trained given the first sentence to generate the second. The tokens associated with higher weights are more label indicative.

the desired label (*i.e.*, the "entailment" and "contradiction" generation results are in fact neutral with respect to the given sentence), lacking label discriminativeness. When $\mathcal{L}_{\text{gen}} + \mathcal{L}_{\text{disc}}$ is used, the generated samples of different labels are more distinctive, but also become less natural and coherent due to the model's language modeling ability being hampered. The generator tuned with $\mathcal{L}_{\text{w-gen}}$ produces both coherent and label-discriminative samples. More concrete generation results for each task can be found in Appendix F.

5.4. Visualization of Learned Token Weights

To understand how token weights are automatically learned during generator tuning, we visualize the learned weights in Fig. 4. The tokens with higher weights (*e.g.*, "weak" in the first example and "hates" in the second example) are learned to be important tokens that decide the relation of the second sentence to the first sentence (*i.e.*, the label of the training sample). With such tokens emphasized during training, the generator is encouraged to capture label-discriminative information that facilitates the generation of unambiguous training samples.

6. Discussions and Conclusions

Ethical Considerations. Despite the impressive text generation and representation power of PLMs, they can also come with the risk (Bender et al., 2021; Bender & Koller, 2020; Brown et al., 2020) of generating disinformation (Pagnoni et al., 2021) or exacerbating biases (Prabhumoye et al., 2018). Instead of improving upon PLM architectures or generation techniques, our work focuses on using existing PLMs to create training data for NLU tasks. In practice, our method can be combined with any bias reduction and correction strategies (Gehman et al., 2020; Ma et al., 2020) to reduce the adverse effects of PLMs.

Limitations. Compared to few-shot learning methods that directly train classification models on the small training set, FewGen requires tuning a generator PLM and using it to synthesize novel training samples, resulting in higher computation costs and longer running time. Still, we believe that our method may bring more good than harm—when the small training data size becomes the performance bottleneck for NLU tasks, a simple yet costly solution is to obtain more human annotations. Our method may replace or reduce the human efforts in such training data creation processes.

Conclusions. In this work, we propose FewGen, which leverages few-shot training samples to tune a generator PLM for synthesizing novel training data. The generated data can be then used in combination with few-shot samples to fine-tune a classification model for better generalization. To emphasize label-discriminative information during generator tuning, we propose a weighted maximum likelihood objective where the token weights are automatically learned via a discriminative meta objective. Since the generated samples may contain label noise, we propose a simple training procedure that first trains classifiers on the few-shot training set and then on the generated set by applying regularization for noise-robustness. Across seven classification tasks from the GLUE benchmark, FewGen significantly outperforms existing approaches under the same few-shot learning setting. The effectiveness of each important component in FewGen is validated via ablation studies. Future directions may include: Using larger PLMs as the generator and the classifier, jointly training both models with each other's high-confident predictions, improving the robustness of models trained on synthetic data, and developing systematic metrics to evaluate the quality of generated training samples.

Acknowledgments

Research was supported in part by US DARPA KAIROS Program No. FA8750-19-2-1004 and INCAS Program No. HR001121C0165, National Science Foundation IIS-19-56151, IIS-17-41317, and IIS 17-04532, and the Molecule Maker Lab Institute: An AI Research Institutes program supported by NSF under Award No. 2019897, and the Institute for Geospatial Understanding through an Integrative Discovery Environment (I-GUIDE) by NSF under Award No. 2118329. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily represent the views, either expressed or implied, of DARPA or the U.S. Government. Yu Meng was supported by the Google PhD Fellowship. We thank anonymous reviewers for valuable and insightful feedback.

References

Anaby-Tavor, A., Carmeli, B., Goldbraich, E., Kantor, A., Kour, G., Shlomov, S., Tepper, N., and Zwerdling, N. Do not have enough data? deep learning to the rescue! In *AAAI*, 2020.

Andrychowicz, M., Denil, M., Colmenarejo, S. G., Hoffman, M. W., Pfau, D., Schaul, T., and de Freitas, N. Learning to learn by gradient descent by gradient descent. In *NIPS*, 2016.

Baum, E. and Haussler, D. What size net gives valid generalization? In *NIPS*, 1988.

Bender, E. M. and Koller, A. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *ACL*, 2020.

Bender, E. M., Gebru, T., McMillan-Major, A., and Shmitchell, S. On the dangers of stochastic parrots: Can language models be too big? In *ACM Conference on Fairness, Accountability, and Transparency*, 2021.

Bentivogli, L., Clark, P., Dagan, I., and Giampiccolo, D. The fifth pascal recognizing textual entailment challenge. In *TAC*, 2009.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T. J., Child, R., Ramesh, A., Ziegler, D. M., 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. Language models are few-shot learners. In *NeurIPS*, 2020.

Chan, A., Ong, Y., Pung, B. T. W., Zhang, A., and Fu, J. CoCon: A self-supervised approach for controlled text generation. In *ICLR*, 2021.

Chen, J., Yang, Z., and Yang, D. MixText: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *ACL*, 2020.

Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020.

Cui, G., Hu, S., Ding, N., Huang, L., and Liu, Z. Prototypical verbalizer for prompt-based few-shot tuning. In *ACL*, 2022.

Dagan, I., Glickman, O., and Magnini, B. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, 2005.

Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., Yosinski, J., and Liu, R. Plug and play language models: A simple approach to controlled text generation. In *ICLR*, 2020.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

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

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., and Pontil, M. Bilevel programming for hyperparameter optimization and meta-learning. In *ICML*, 2018.

Gao, J., Pi, R., Lin, Y., Xu, H., Ye, J., Wu, Z., Zhang, W., Liang, X., Li, Z., and Kong, L. Self-guided noise-free data generation for efficient zero-shot learning. In *ICLR*, 2023.

Gao, T., Fisch, A., and Chen, D. Making pre-trained language models better few-shot learners. In *ACL*, 2021.

Gehman, S., Gururangan, S., Sap, M., Choi, Y., and Smith, N. A. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *EMNLP Findings*, 2020.

Giampiccolo, D., Magnini, B., Dagan, I., and Dolan, B. The third pascal recognizing textual entailment challenge. In *ACL-PASCAL workshop on textual entailment and paraphrasing*, 2007.

Haim, R. B., Dagan, I., Dolan, B., Ferro, L., Giampiccolo, D., Magnini, B., and Szpektor, I. The second pascal recognising textual entailment challenge. In *PASCAL Challenges Workshop on Recognising Textual Entailment*, 2006.

Hambardzumyan, K., Khachatrian, H., and May, J. WARP: Word-level adversarial reprogramming. In *ACL*, 2021.

He, P., Liu, X., Gao, J., and Chen, W. DeBERTa: Decoding-enhanced BERT with disentangled attention. In *ICLR*, 2021.

Hendrycks, D., Mazeika, M., Wilson, D., and Gimpel, K. Using trusted data to train deep networks on labels corrupted by severe noise. In *NeurIPS*, 2018.

Hu, S., Ding, N., Wang, H., Liu, Z., Li, J.-Z., and Sun, M. Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification. In *ACL*, 2022.

Hu, Z., Yang, Z., Liang, X., Salakhutdinov, R., and Xing, E. P. Toward controlled generation of text. In *ICML*, 2017.

Huang, J., Gu, S. S., Hou, L., Wu, Y., Wang, X., Yu, H., and Han, J. Large language models can self-improve. *ArXiv*, abs/2210.11610, 2022.

Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H. T., Heafield, K., Neckermann, T., Seide, F., Germann, U., Aji, A. F., Bogoychev, N., Martins, A. F. T., and Birch, A. Marian: Fast neural machine translation in C++. In *ACL System Demo*, 2018.

Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., and Socher, R. CTRL: A conditional transformer language model for controllable generation. *ArXiv*, abs/1909.05858, 2019.

Khalifa, M., ElSahar, H., and Dymetman, M. A distributional approach to controlled text generation. In *ICLR*, 2021.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 2017.

Krause, B., Gotmare, A. D., McCann, B., Keskar, N. S., Joty, S. R., Socher, R., and Rajani, N. GeDi: Generative discriminator guided sequence generation. In *EMNLP*, 2021.

Kumar, S., Malmi, E., Severyn, A., and Tsvetkov, Y. Controlled text generation as continuous optimization with multiple constraints. In *NeurIPS*, 2021.

Kumar, V., Choudhary, A., and Cho, E. Data augmentation using pre-trained transformer models. In *Workshop on Life-long Learning for Spoken Language Systems*, 2020.

Laine, S. and Aila, T. Temporal ensembling for semi-supervised learning. In *ICLR*, 2017.

Lee, K., Guu, K., He, L., Dozat, T., and Chung, H. W. Neural data augmentation via example extrapolation. *arXiv preprint arXiv:2102.01335*, 2021.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, 2021.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, 2021.

Liu, A., Sap, M., Lu, X., Swayamdipta, S., Bhagavatula, C., Smith, N. A., and Choi, Y. DExperts: Decoding-time controlled text generation with experts and anti-experts. In *ACL*, 2021a.

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *NeurIPS*, 2022a.

Liu, J., Shen, D., Zhang, Y., Dolan, B., Carin, L., and Chen, W. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out*, 2022b.

Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., and Tang, J. GPT understands, too. *ArXiv*, abs/2103.10385, 2021b.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Logan IV, R. L., Balažević, I., Wallace, E., Petroni, F., Singh, S., and Riedel, S. Cutting down on prompts and parameters: Simple few-shot learning with language models. *arXiv preprint arXiv:2106.13353*, 2021.

Ma, X., Sap, M., Rashkin, H., and Choi, Y. PowerTransformer: Unsupervised controllable revision for biased language correction. In *EMNLP*, 2020.

Meng, Y., Shen, J., Zhang, C., and Han, J. Weakly-supervised neural text classification. In *CIKM*, 2018.

Meng, Y., Shen, J., Zhang, C., and Han, J. Weakly-supervised hierarchical text classification. In *AAAI*, 2019.

Meng, Y., Xiong, C., Bajaj, P., Tiwary, S., Bennett, P., Han, J., and Song, X. COCO-LM: Correcting and contrasting text sequences for language model pretraining. In *NeurIPS*, 2021a.

Meng, Y., Zhang, Y., Huang, J., Wang, X., Zhang, Y., Ji, H., and Han, J. Distantly-supervised named entity recognition with noise-robust learning and language model augmented self-training. In *EMNLP*, 2021b.

Meng, Y., Huang, J., Zhang, Y., and Han, J. Generating training data with language models: Towards zero-shot language understanding. In *NeurIPS*, 2022a.

Meng, Y., Xiong, C., Bajaj, P., Tiwary, S., Bennett, P., Han, J., and Song, X. Pretraining text encoders with adversarial mixture of training signal generators. In *ICLR*, 2022b.

Min, S., Lewis, M., Hajishirzi, H., and Zettlemoyer, L. Noisy channel language model prompting for few-shot text classification. In *ACL*, 2022a.

Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., and Zettlemoyer, L. Rethinking the role of demonstrations: What makes in-context learning work? In *EMNLP*, 2022b.

Miyato, T., Dai, A. M., and Goodfellow, I. J. Adversarial training methods for semi-supervised text classification. In *ICLR*, 2017.

Nguyen, D. T., Mummadi, C. K., Ngo, T.-P.-N., Nguyen, T. H. P., Beggel, L., and Brox, T. SELF: Learning to filter noisy labels with self-ensembling. In *ICLR*, 2020.

Pagnoni, A., Balachandran, V., and Tsvetkov, Y. Understanding factuality in abstractive summarization with FRANK: A benchmark for factuality metrics. In *NAACL*, 2021.

Pascual, D., Egressy, B., Meister, C., Cotterell, R., and Wattenhofer, R. A plug-and-play method for controlled text generation. In *EMNLP Findings*, 2021.

Perez, E., Kiela, D., and Cho, K. True few-shot learning with language models. In *NeurIPS*, 2021.

Prabhumoye, S., Tsvetkov, Y., Salakhutdinov, R., and Black, A. W. Style transfer through back-translation. In *ACL*, 2018.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 2019.

Ren, M., Zeng, W., Yang, B., and Urtasun, R. Learning to reweight examples for robust deep learning. In *ICML*, 2018.

Scao, T. L. and Rush, A. M. How many data points is a prompt worth? In *NAACL*, 2021.

Schick, T. and Schütze, H. Exploiting cloze-questions for few-shot text classification and natural language inference. In *EACL*, 2021a.

Schick, T. and Schütze, H. Few-shot text generation with natural language instructions. In *EMNLP*, 2021b.

Schick, T. and Schütze, H. Generating datasets with pretrained language models. In *EMNLP*, 2021c.

Schick, T. and Schütze, H. It's not just size that matters: Small language models are also few-shot learners. In *NAACL*, 2021d.

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

Shin, T., Razeghi, Y., IV, R. L. L., Wallace, E., and Singh, S. Eliciting knowledge from language models using automatically generated prompts. In *EMNLP*, 2020.

Shu, J., Xie, Q., Yi, L., Zhao, Q., Zhou, S., Xu, Z., and Meng, D. Meta-weight-net: Learning an explicit mapping for sample weighting. In *NeurIPS*, 2019.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, 2013.

Standley, T. S., Zamir, A. R., Chen, D., Guibas, L. J., Malik, J., and Savarese, S. Which tasks should be learned together in multi-task learning? In *ICML*, 2019.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

Tam, D., Menon, R. R., Bansal, M., Srivastava, S., and Raffel, C. Improving and simplifying pattern exploiting training. In *EMNLP*, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *EMNLP Workshop BlackboxNLP*, 2018.

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS*, 2019.

Wang, Y.-X., Ramanan, D., and Hebert, M. Learning to model the tail. In *NIPS*, 2017.

Wang, Z., Yu, A. W., Firat, O., and Cao, Y. Towards zero-label language learning. *ArXiv*, abs/2109.09193, 2021.

Warstadt, A., Singh, A., and Bowman, S. R. Neural network acceptability judgments. In *TACL*, 2019.

Wei, J. and Zou, K. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *EMNLP*, 2019.

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

Wu, L., Tian, F., Xia, Y., Fan, Y., Qin, T., Lai, J., and Liu, T.-Y. Learning to teach with dynamic loss functions. In *NeurIPS*, 2018.

Xie, Q., Dai, Z., Hovy, E. H., Luong, M.-T., and Le, Q. V. Unsupervised data augmentation for consistency training. In *NeurIPS*, 2020.

Yang, K. and Klein, D. FUDGE: Controlled text generation with future discriminators. In *NAACL*, 2021.

Yang, Y., Malaviya, C., Fernandez, J., Swayamdipta, S., Bras, R. L., ping Wang, J., Bhagavatula, C., Choi, Y., and Downey, D. G-daug: Generative data augmentation for commonsense reasoning. In *EMNLP Findings*, 2020.

Ye, J., Gao, J., Li, Q., Xu, H., Feng, J., Wu, Z., Yu, T., and Kong, L. ZeroGen: Efficient zero-shot learning via dataset generation. In *EMNLP*, 2022.

Yoo, K. M., Park, D.-H., Kang, J., Lee, S.-W., and Park, W. GPT3Mix: Leveraging large-scale language models for text augmentation. In *EMNLP Findings*, 2021.

Zhang, N., Li, L., Chen, X., Deng, S., Bi, Z., Tan, C., Huang, F., and Chen, H. Differentiable prompt makes pre-trained language models better few-shot learners. In *ICLR*, 2022.

Zhao, T., Wallace, E., Feng, S., Klein, D., and Singh, S. Calibrate before use: Improving few-shot performance of language models. In *ICML*, 2021.

Zhong, Z., Friedman, D., and Chen, D. Factual probing is [mask]: Learning vs. learning to recall. In *NAACL*, 2021.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. Fine-tuning language models from human preferences. *ArXiv*, abs/1909.08593, 2019.

A. Details of Weighting Network Implementation

Since the token weights $\boldsymbol{w}$ used in Eq. (4) need to characterize the discriminativeness of each token, we use the value of discriminative objective at each token $\mathcal{L}^j_{\text{disc}}$ as the input to the weighting network, and we use softmax to normalize the weights:

$$w_j(\boldsymbol{\omega}) = \frac{\exp\left(g_{\boldsymbol{\omega}}(\mathcal{L}^j_{\text{disc}})\right)}{\sum_{j'=1}^{n}\exp\left(g_{\boldsymbol{\omega}}(\mathcal{L}^{j'}_{\text{disc}})\right)}.$$

Following (Shu et al., 2019), we instantiate $g_{\boldsymbol{\omega}}$ to be a feedforward network (FFN) with only one 100-dimension hidden layer by default.

B. Implementation Details

Table 5: Prompts used for initializing the prefix vectors and control codes (required by CTRL (Keskar et al., 2019)) used in generator training. The control codes are selected to approximiate the task domain. For single-sequence tasks, $\boldsymbol{x}$ denotes the training sample; for sequence-pair tasks, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ denote the first and second sequence in the training sample, respectively.

Task	Task Type	Control Code	Label	Initialization Prompt
SST-2	single-sequence	Reviews	positive negative	Rating: 5.0 positive movie review: $\boldsymbol{x}$ Rating: 1.0 negative movie review: $\boldsymbol{x}$
CoLA	single-sequence	Links	grammatical not grammatical	Linguistically correct sentence: $\boldsymbol{x}$ Linguistically incorrect sentence: $\boldsymbol{x}$
MNLI	sequence-pair	Wikipedia	entailment neutral contradiction	Sentence 1 implies Sentence 2. Sentence 1: $\boldsymbol{x}_1$ Sentence 2: $\boldsymbol{x}_2$ Sentence 2 supplements Sentence 1. Sentence 1: $\boldsymbol{x}_1$ Sentence 2: $\boldsymbol{x}_2$ Sentence 2 contradicts Sentence 1. Sentence 1: $\boldsymbol{x}_1$ Sentence 2: $\boldsymbol{x}_2$
QNLI	sequence-pair	Links	entailment not entailment	Paragraph is relevant to Question. Question: $\boldsymbol{x}_1$ Paragraph: $\boldsymbol{x}_2$ Paragraph is irrelevant to Question. Question: $\boldsymbol{x}_1$ Paragraph: $\boldsymbol{x}_2$
RTE	sequence-pair	Wikipedia	entailment not entailment	Sentence 1 implies Sentence 2. Sentence 1: $\boldsymbol{x}_1$ Sentence 2: $\boldsymbol{x}_2$ Sentence 2 supplements Sentence 1. Sentence 1: $\boldsymbol{x}_1$ Sentence 2: $\boldsymbol{x}_2$
MRPC	sequence-pair	Wikipedia	equivalent not equivalent	Sentence 1 is equivalent to Sentence 2. Sentence 1: $\boldsymbol{x}_1$ Sentence 2: $\boldsymbol{x}_2$ Sentence 1 is different from Sentence 2. Sentence 1: $\boldsymbol{x}_1$ Sentence 2: $\boldsymbol{x}_2$
QQP	sequence-pair	Links	equivalent not equivalent	Question 1 is equivalent to Question 2. Question 1: $\boldsymbol{x}_1$ Question 2: $\boldsymbol{x}_2$ Question 1 is different from Question 2. Question 1: $\boldsymbol{x}_1$ Question 2: $\boldsymbol{x}_2$

Details of Initialization Prompts Used for Generator Tuning on Different Tasks. For generator tuning, we find it beneficial to initialize the prefix vectors with task-descriptive prompts, similar to the observations in (Li & Liang, 2021). The prefix lengths (*i.e.*, number of trained prefix token positions) are equal to the number of tokens in the prompts. We present details about the prompts used for initializing the prefix vectors for different tasks in Table 5. For sequence-pair tasks, an additional infix prompt is used between the two sequences, and we also tune the embeddings of the infix (*i.e.*, prompt-tuning (Lester et al., 2021)) for generator training.

Details of Generator Tuning. The meta-weighted generator tuning procedure (Algorithm 1) involves three forward and backward passes, and thus its time complexity is approximately 3 times of standard generator training without meta learning. However, since the few-shot training sets have a small amount of training data, the extra time cost is usually affordable. In practice, our generator tuning with meta weight learning takes 10 minutes to train on each task (the standard generator training time without meta-learning is 3.5 minutes). We use a fixed set of hyperparamters for all tasks without task-specific hyperparamter tuning: In Algorithm 1, we set the batch size to be 2, the learning rate for optimizing $\hat{\boldsymbol{\theta}}_p$ to be $2e-2$, the learning rate for optimizing $\boldsymbol{\omega}$ to be $1e-2$, the learning rate for optimizing $\boldsymbol{\theta}_p$ to be $5e-3$, and training epoch to be 20. We also experiment with larger batch sizes (*e.g.*, 16/32) and/or training for more epochs, but they result in worse language modeling quality than the default hyperparamters.

Details of Generating Training Data. Following (Meng et al., 2022a), for sequence-pair tasks (MNLI, QQP, QNLI, RTE and MRPC), we randomly sample the first sequence from the pretraining corpus (*e.g.*, Wikipedia) and use greedy sampling

for generating the second sequence. For single-sequence tasks (SST-2 and CoLA), we use top-k sampling with temperature to generate training data from scratch where $k = 10$. For all tasks, we generate $5,000$ samples per label.

For SST-2, we use one of the following tokens to start generation: "a", "one", "the", "this", "that", "i", "you", "it", "what". For CoLA, we use a random stop word to start generation.

We apply repetition penalty (Keskar et al., 2019) to the logits of tokens that have already appeared in the sequence. Overall, the token probability distribution is post-processed as follows before conducting sampling:

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

where τ is the temperature hyperparameter, and α is the repetition penalty hyperparameter. For labels that favor token repetitions between the first and the second sequences (*e.g.*, paraphrase or entailment), we set α to be a smaller value (*e.g.*, 1.0), and vice versa.

The hyperparameter values for training data generation on all tasks can be found in Table 6.

Table 6: Hyperparameters for generating training data for different tasks. τ: Temperature during sampling ($\tau = 0$ means greedy sampling); α: Repetition penalty.

Task	Label	τ	α
SST-2	positive	0.5	1.1
	negative		1.1
CoLA	grammatical	0.3	1.1
	not grammatical	10	1.1
MNLI	entailment	0	1.1
	neutral		1.5
	contradiction		1.1
QNLI	entailment	0	1.0
	not entailment		1.5
RTE	entailment	0	1.0
	not entailment		1.5
MRPC	equivalent	0	1.0
	not equivalent		1.5
QQP	equivalent	0	1.0
	not equivalent		1.5

Hyperparameters for Fine-Tuning Classifier PLMs. For fine-tuning on the few-shot training samples $\mathcal{D}_{\text{train}}$, we search among the following hyperparameter ranges based on development set ($\mathcal{D}_{\text{dev}}$) model performance and pick the best performing model for futher fine-tuning on synthesized data: Learning rate in $[1e-5, 2e-5]$ and batch size in $[4, 8]$. The number of training steps is fixed to be 1000. We also find it beneficial to apply label smoothing (smoothing weight set to 0.15) for fine-tuning on the few-shot training set.

For fine-tuning on the synthesized training samples $\mathcal{D}_{\text{gen}}$, we use the following hyperparameters: $5e-6$ as the learning rate; 16 as the batch size; label smoothing weight $\epsilon = 0.15$; temporal ensemble momentum $\gamma = 0.9$; temporal ensemble loss weight $\lambda = 20$; training steps $T = 6,000$.

Details of Temporal Ensembling for Fine-Tuning Classifier PLMs on Synthetic Data. We update ensembled predictions $\bar{\boldsymbol{z}}$ as follows where $\boldsymbol{p}_\phi$ is the current model prediction, γ is the momentum parameter, $\hat{\boldsymbol{z}}$ is the accumulated model prediction before bias correction, $\bar{\boldsymbol{z}}$ is the accumulated model prediction after bias correction, and t is the number of updates $\bar{\boldsymbol{z}}$ has received:

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

The accumulated model prediction $\hat{\boldsymbol{z}}$ has a zero initialization; the division $(1-\gamma^t)$ is for bias correction (Laine & Aila, 2017). After each update of $\hat{\boldsymbol{z}}$, it will be compared to a threshold value δ; each synthesized sample $(\tilde{\boldsymbol{x}}, \tilde{y})$ will be included in training only if $\bar{z}_{\tilde{y}} > \delta$.

We update the ensembled predictions $\bar{\boldsymbol{z}}$ on all samples in $\mathcal{D}_{\text{gen}}$ every 200 steps, and set the threshold value for sample filtering $\delta = 0.8$.

Computation Environment. The experiments are conducted on NVIDIA A100 GPUs.

C. Derivation of Meta Weight Gradient Update

We first write out the gradient update of $\hat{\boldsymbol{\theta}}_p^{(t)}\left(\boldsymbol{\omega}^{(t)}\right)$ and $\boldsymbol{\omega}^{(t+1)}$ according to Algorithm 1 as follows:

$$\hat{\boldsymbol{\theta}}_p^{(t)}\left(\boldsymbol{\omega}^{(t)}\right) = \boldsymbol{\theta}_p^{(t)} - \alpha \left.\frac{\partial \mathcal{L}_{\text{w-gen}}\left(\boldsymbol{\theta}_p; \boldsymbol{\omega}^{(t)}\right)}{\partial \boldsymbol{\theta}_p}\right|_{\boldsymbol{\theta}_p=\boldsymbol{\theta}_p^{(t)}} = \boldsymbol{\theta}_p^{(t)} - \alpha \sum_{j=1}^{n} w_j\left(\boldsymbol{\omega}^{(t)}\right) \left.\frac{\partial \mathcal{L}_{\text{gen}}^{j}(\boldsymbol{\theta}_p)}{\partial \boldsymbol{\theta}_p}\right|_{\boldsymbol{\theta}_p=\boldsymbol{\theta}_p^{(t)}} \tag{6}$$

$$\boldsymbol{\omega}^{(t+1)} = \boldsymbol{\omega}^{(t)} - \beta \left.\frac{\partial \mathcal{L}_{\text{disc}}\left(\hat{\boldsymbol{\theta}}_p^{(t)}(\boldsymbol{\omega})\right)}{\partial \boldsymbol{\omega}}\right|_{\boldsymbol{\omega}=\boldsymbol{\omega}^{(t)}}. \tag{7}$$

where α and β are step sizes.

The gradient in Equation (7) is calculated as:

$$\begin{aligned}
&\left.\frac{\partial \mathcal{L}_{\text{disc}}\left(\hat{\boldsymbol{\theta}}_p^{(t)}(\boldsymbol{\omega})\right)}{\partial \boldsymbol{\omega}}\right|_{\boldsymbol{\omega}=\boldsymbol{\omega}^{(t)}} \\
&= \left.\frac{\partial \mathcal{L}_{\text{disc}}\left(\hat{\boldsymbol{\theta}}_p\right)}{\partial \hat{\boldsymbol{\theta}}_p}\right|_{\hat{\boldsymbol{\theta}}_p=\hat{\boldsymbol{\theta}}_p^{(t)}} \left.\frac{\partial \hat{\boldsymbol{\theta}}_p(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}}\right|_{\boldsymbol{\omega}=\boldsymbol{\omega}^{(t)}} \\
&= \left.\frac{\partial \mathcal{L}_{\text{disc}}\left(\hat{\boldsymbol{\theta}}_p\right)}{\partial \hat{\boldsymbol{\theta}}_p}\right|_{\hat{\boldsymbol{\theta}}_p=\hat{\boldsymbol{\theta}}_p^{(t)}} \left(-\alpha \sum_{j=1}^{n} \left.\frac{\partial \mathcal{L}_{\text{gen}}^{j}(\boldsymbol{\theta}_p)}{\partial \boldsymbol{\theta}_p}\right|_{\boldsymbol{\theta}_p=\boldsymbol{\theta}_p^{(t)}}^{\top} \left.\frac{\partial w_j(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}}\right|_{\boldsymbol{\omega}=\boldsymbol{\omega}^{(t)}}\right) && \text{Plugging in Eq. (6)} \\
&= -\alpha \sum_{j=1}^{n} \left(\underbrace{\left.\frac{\partial \mathcal{L}_{\text{disc}}\left(\hat{\boldsymbol{\theta}}_p\right)}{\partial \hat{\boldsymbol{\theta}}_p}\right|_{\hat{\boldsymbol{\theta}}_p=\hat{\boldsymbol{\theta}}_p^{(t)}} \left.\frac{\partial \mathcal{L}_{\text{gen}}^{j}(\boldsymbol{\theta}_p)}{\partial \boldsymbol{\theta}_p}\right|_{\boldsymbol{\theta}_p=\boldsymbol{\theta}_p^{(t)}}^{\top}}_{\triangleq d_j} \left.\frac{\partial w_j(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}}\right|_{\boldsymbol{\omega}=\boldsymbol{\omega}^{(t)}}\right)
\end{aligned}$$

Therefore,

$$-\left.\frac{\partial \mathcal{L}_{\text{disc}}\left(\hat{\boldsymbol{\theta}}_p^{(t)}(\boldsymbol{\omega})\right)}{\partial \boldsymbol{\omega}}\right|_{\boldsymbol{\omega}=\boldsymbol{\omega}^{(t)}} \propto \sum_{j=1}^{n} d_j \left.\frac{\partial w_j(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}}\right|_{\boldsymbol{\omega}=\boldsymbol{\omega}^{(t)}}, \quad d_j = \left.\frac{\partial \mathcal{L}_{\text{disc}}\left(\hat{\boldsymbol{\theta}}_p\right)}{\partial \hat{\boldsymbol{\theta}}_p}\right|_{\hat{\boldsymbol{\theta}}_p=\hat{\boldsymbol{\theta}}_p^{(t)}} \left.\frac{\partial \mathcal{L}_{\text{gen}}^{j}(\boldsymbol{\theta}_p)}{\partial \boldsymbol{\theta}_p}\right|_{\boldsymbol{\theta}_p=\boldsymbol{\theta}_p^{(t)}}^{\top}.$$

D. GLUE Tasks

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

MNLI: Multi-genre Natural Language Inference (Williams et al., 2018) requires predicting whether a given premise sentence entails, contradicts or neutral with respect to a given hypothesis sentence.

QQP: Quora Question Pairs (Shankar et al., 2017) requires judging whether a pair of questions asked are semantically equivalent.

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

SST-2: Stanford Sentiment Treebank (Socher et al., 2013) requires determining if a movie review has positive or negative sentiment.

CoLA: Corpus of Linguistic Acceptability (Warstadt et al., 2019) requires determining whether a given sentence is linguistically acceptable or not.

RTE: Recognizing Textual Entailment (Bentivogli et al., 2009; Dagan et al., 2005; Giampiccolo et al., 2007; Haim et al., 2006) requires predicting whether a given premise sentence entails a given hypothesis sentence or not.

MRPC: Microsoft Research Paraphrase Corpus (Dolan & Brockett, 2005) requires predicting whether two sentences are semantically equivalent or not.

Table 7: Prompts used for GPT3Mix augmentation. For sequence-pair tasks, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ denote the first and second input sequence, respectively. For single-sequence tasks, $\boldsymbol{x}$ denotes the input sequence. y denotes the label name. Only one example is shown in the template for clarity; in practice, we concatenate $k = 4$ samples according to the optimal setting in GPT3Mix (Yoo et al., 2021).

Task	Template	Label name
SST-2	Each item in the following list contains a movie review and the respective sentiment. The sentiment is one of 'positive' or 'negative'. Movie review: $\boldsymbol{x}$ (Sentiment: y) ...	positive: positive negative: negative
CoLA	Each item in the following list contains a text and the respective grammar. The grammar is one of 'correct' or 'incorrect'. Text: $\boldsymbol{x}$ (Grammar: y) ...	grammatical: correct not grammatical: incorrect
MNLI	Each item in the following list contains a premise, a hypothesis and their logical relation. The logical relation is one of 'entailment', 'neutral' or 'contradiction'. Premise: $\boldsymbol{x}_1$ Hypothesis: $\boldsymbol{x}_2$ (Logical relation: y) ...	entailment: entailment neutral: neutral contradiction: contradiction
QNLI	Each item in the following list contains a question, an answer and their logical relation. The logical relation is one of 'entailment' or 'neutral'. Question: $\boldsymbol{x}_1$ Answer: $\boldsymbol{x}_2$ (Logical relation: y) ...	entailment: entailment not entailment: neutral
RTE	Each item in the following list contains a premise, a hypothesis and their logical relation. The logical relation is one of 'entailment' or 'neutral'. Premise: $\boldsymbol{x}_1$ Hypothesis: $\boldsymbol{x}_2$ (Logical relation: y) ...	entailment: entailment not entailment: neutral
MRPC	Each item in the following list contains two sentences and their semantic relation. The semantic relation is one of 'equivalent' or 'different'. Sentence 1: $\boldsymbol{x}_1$ Sentence 2: $\boldsymbol{x}_2$ (Semantic relation: y) ...	equivalent: equivalent not equivalent: different
QQP	Each item in the following list contains two questions and their semantic relation. The semantic relation is one of 'equivalent' or 'different'. Question 1: $\boldsymbol{x}_1$ Question 2: $\boldsymbol{x}_2$ (Semantic relation: y) ...	equivalent: equivalent not equivalent: different

E. Data Augmentation Baseline Details

Details About MixText (Chen et al., 2020). We use the TMix version of MixText to perform data interpolation on the few-shot labeled dataset (since there is no access to unlabeled task-specific data under the strict few-shot learning setting (Gao et al., 2021)). We adapt the label mix-up operation to fit prompt-based fine-tuning by interpolating the label words instead of categorical labels; we observe that this results in better few-shot performance than the original TMix, probably analogous to why prompt-based fine-tuning outperforms standard fine-tuning for few-shot learning. We train the classifier with supervised loss combined with consistency loss over the interpolated samples as in the original paper. We follow the default hyperparameters in MixText.

Details About Back Translation. We use two trained Marian (Junczys-Dowmunt et al., 2018) models to perform data augmentation via back translation. We translate our labeled examples from English to French, and then back to English. As in UDA (Xie et al., 2020), we employ random sampling with a tunable temperature to generate a diverse set of derivative examples. We generate 32 examples from each few-shot training example and let the synthesized samples share the same label with the original few-shot training sample. After combining with the original examples, we fine-tune the classifier and observe performance.

Details About GPT3Mix (Yoo et al., 2021). We use the 175B GPT3 model for generating the augmentations. For creating each augmentation, we randomly sample $k = 4$ (the optimal setting according to GPT3Mix) examples from the few-shot training set as demonstrations. The prompts follow the suggested format proposed in the original paper (Yoo et al., 2021) and are shown in Table 7. We create $5,000$ augmented samples per label to make the resulting training set size equal to that of FewGen. After obtaining the augmented examples and their pseudo labels (the probability predictions over all labels by GPT3), we use them along with the real few-shot samples for fine-tuning the classifier, following the setting in GPT3Mix (Yoo et al., 2021).

Details About Standard Generator Fine-Tuning. We fine-tune the same 1.6B CTRL (Keskar et al., 2019) model as used in FewGen with the standard maximum likelihood objective. Different from previous studies (Anaby-Tavor et al., 2020; Kumar et al., 2020) that prepend categorical labels to the training samples, we enhance the generator fine-tuning with label-descriptive prompts (shown in Table 5) used in FewGen. We create $5,000$ augmented samples per label to make the resulting training set size equal to that of FewGen.

F. Concrete Generation Results

We present some concrete generation results (from $\mathcal{D}_{\text{gen}}$) for all tasks in Tables 8, 9, 10, 11, 12, 13, and 14. To compare $\mathcal{D}_{\text{gen}}$ with $\mathcal{D}_{\text{train}}$, we also show the few-shot training samples ($\mathcal{D}_{\text{train}}$) of SST-2 in Table 15,

Comparing Tables 8 with 15, it can be seen that the synthetic samples are accurate and quite different from the given training samples to serve as effective augmentations. For sequence-pair tasks, because we randomly sample the first sequence from the pretraining corpus and let the generator create the second sequence given certain labels, the resulting generated samples will be certainly different from the given training samples.

Table 8: Examples of concrete generations for SST-2.

Example	Label	Generated Text
#1	positive	i was a bit skeptical when i first saw the trailer for this movie, but after watching it, it was quite enjoyable.
	negative	i was expecting a fun and entertaining movie with some nice special effects, but instead it was just plain boring.
#2	positive	that is, a film that has the potential to move an audience to tears, or at least to a state of mind where you feel something.
	negative	a film that has the potential to be one of the most moving and powerful experiences ever made is instead marred by an unfocused script, bad acting and a lack of emotional depth.
#3	positive	this is a good movie – it's not just for people who like to see things blow up or have explosions or shoot guns.
	negative	i am a big fan of the first two movies but this one is just plain bad.

Table 9: Examples of concrete generations for CoLA.

Example	Label	Generated Text
#1	grammatical	Between the two of us, I think you have a good chance to make it to the finals.
	not grammatical	A few weeks later we saw them on television. We thought there were a hundred more. It wasn't that they came out one at time.
#2	grammatical	Doing the dishes is a task that I have to do because I am not allowed to go out and play with my friends.
	not grammatical	Only those whose lives depend on it, but only when you are at its mercy do it harm.
#3	grammatical	To be able to speak the language of the other person is a prerequisite for communication.
	not grammatical	The only good way to avoid getting into trouble for being too friendly would to let me go on a trip.

Table 10: Examples of concrete generations for QQP. The first question (*italicized*) is randomly sampled from the pretraining corpus; the second question (underlined) is generated by $G_{\boldsymbol{\theta}_p}$.

Example	Label	Generated Text
#1	equivalent	Question 1: *How long does it take for a project to be completed?* Question 2: How long does it take to complete a project?
	not equivalent	Question 1: *How long does it take for a project to be completed?* Question 2: What is the total cost of completing this project?
#2	equivalent	Question 1: *What mascots or characters would you like to see included as Super Mario Maker DLC in the future?* Question 2: What would you like to see in Super Mario Maker DLC that you did not see in the game?
	not equivalent	Question 1: *What mascots or characters would you like to see included as Super Mario Maker DLC in the future?* Question 2: How do I get a copy of this game?

Table 11: Examples of concrete generations for MNLI. The first sentence (*italicized*) is randomly sampled from the pretraining corpus; the second sentence (underlined) is generated by $G_{\boldsymbol{\theta}_p}$.

Example	Label	Generated Text
#1	entailment	Sentence 1: *Air is provided for the combustion by an electric blower.* Sentence 2: The blower provides air to a combustion chamber.
	neutral	Sentence 1: *Air is provided for the combustion by an electric blower.* Sentence 2: Electric blowers are available in most gas stations.
	contradiction	Sentence 1: *Air is provided for the combustion by an electric blower.* Sentence 2: The blower does not work.
#2	entailment	Sentence 1: *Since its base is almost at sea level, it is only the 15th highest light in the United States, the first 14 being built on higher ground.* Sentence 2: It is the 15th highest light in the United States.
	neutral	Sentence 1: *Since its base is almost at sea level, it is only the 15th highest light in the United States, the first 14 being built on higher ground.* Sentence 2: The lighthouse was originally constructed to be a beacon for ships passing by and as such has been used since before World War II.
	contradiction	Sentence 1: *Since its base is almost at sea level, it is only the 15th highest light in the United States, the first 14 being built on higher ground.* Sentence 2: It is located on a mountain top.

Table 12: Examples of concrete generations for QNLI. The question (*italicized*) is randomly sampled from the pretraining corpus; the answer (underlined) is generated by $G_{\boldsymbol{\theta}_p}$.

Example	Label	Generated Text
#1	entailment	Question: *What makes you want to step up to the next level?* Answer: I want to be the best player I can be.
	not entailment	Question: *What makes you want to step up to the next level?* Answer: The new program will be called "Project 10" and it is expected that a total of $450 million in federal funding would go toward it.
#2	entailment	Question: *How do all those shops know what you would like to buy?* Answer: The stores are able to track your preferences and provide you with a list of products that are best for you.
	not entailment	Question: *How do all those shops know what you would like to buy?* Answer: The stores are not required by law or regulation in the United States and Canada but they have been known for years as a source of illegal sales on eBay.

Table 13: Examples of concrete generations for RTE. The first sentence (*italicized*) is randomly sampled from the pretraining corpus; the second sentence (underlined) is generated by $G_{\boldsymbol{\theta}_p}$.

Example	Label	Generated Text
#1	entailment	Sentence 1: *In 1970, the Science Council of Canada recommended that the government of Canada immediately invest in industrial development of the aviation industry, including construction of aircraft, navigation aids, and regulation of air traffic.* Sentence 2: The government of Canada has invested in the aviation industry.
	not entailment	Sentence 1: *In 1970, the Science Council of Canada recommended that the government of Canada immediately invest in industrial development of the aviation industry, including construction of aircraft, navigation aids, and regulation of air traffic.* Sentence 2: The Aviation Industry was established by a Royal Decree on June 1, 1970.
#2	entailment	Sentence 1: *All of the Centre's staff are fluently bilingual in both English and Chinese and are familiar with the traditions of the Chinese culture.* Sentence 2: The Centre is a bilingual institution.
	not entailment	Sentence 1: *All of the Centre's staff are fluently bilingual in both English and Chinese and are familiar with the traditions of the Chinese culture.* Sentence 2: The Centre is a cultural centre for learning about China.

Table 14: Examples of concrete generations for MRPC. The first sentence (*italicized*) is randomly sampled from the pretraining corpus; the second sentence (underlined) is generated by $G_{\boldsymbol{\theta}_p}$.

Example	Label	Generated Text
#1	entailment	Sentence 1: *It embodies the aesthetic ideals of the Queen Anne style through its design and detailing and shows the type of house that attracted prosperous Napans at the end of the Victorian era.* Sentence 2: It is a fine example of the Queen Anne style of architecture through its design and detailing and shows the type of house that attracted prosperous Napans at the end of the Victorian era.
	not entailment	Sentence 1: *It embodies the aesthetic ideals of the Queen Anne style through its design and detailing and shows the type of house that attracted prosperous Napans at the end of the Victorian era.* Sentence 2: The building is a fine example in this style, with an elegant facade reminiscent to those found on many grand mansions built by wealthy merchants during America's Gilded Age.
#2	entailment	Sentence 1: *Crosbie ran unsuccessfully for the leadership of the Liberal Party of Newfoundland and Labrador in 1969, losing to Smallwood, and was also a candidate in the Progressive Conservative Party of Canada's 1983 leadership election, placing third.* Sentence 2: Crosbie was a candidate in the Progressive Conservative Party of Canada's 1983 leadership election, placing third.
	not entailment	Sentence 1: *Crosbie ran unsuccessfully for the leadership of the Liberal Party of Newfoundland and Labrador in 1969, losing to Smallwood, and was also a candidate in the Progressive Conservative Party of Canada's 1983 leadership election, placing third.* Sentence 2: He lost his bid as leader after he failed twice at running against John Diefenbaker.

Table 15: 16-shot training samples of SST-2.

Label	Example	Review Text
positive	#1	(ramsay) visually transforms the dreary expanse of dead-end distaste the characters inhabit into a poem of art , music and metaphor .
	#2	the film jolts the laughs from the audience – as if by cattle prod .
	#3	the film presents visceral and dangerously honest revelations about the men and machines behind the curtains of our planet .
	#4	a film that will enthrall the whole family .
	#5	serious movie-goers embarking upon this journey will find that the road to perdition leads to a satisfying destination .
	#6	sweet and memorable film .
	#7	shyamalan takes a potentially trite and overused concept (aliens come to earth) and infuses it into a rustic , realistic , and altogether creepy tale of hidden invasion .
	#8	a crisp psychological drama (and) a fascinating little thriller that would have been perfect for an old " twilight zone " episode .
	#9	my big fat greek wedding is not only the best date movie of the year , it 's also a – dare i say it twice – delightfully charming – and totally american , i might add – slice of comedic bliss .
	#10	a comedy-drama of nearly epic proportions rooted in a sincere performance by the title character undergoing midlife crisis .
	#11	diggs and lathan are among the chief reasons brown sugar is such a sweet and sexy film .
	#12	you 're not merely watching history , you 're engulfed by it .
	#13	the concept is a hoot .
	#14	the filmmakers ' eye for detail and the high standards of performance convey a strong sense of the girls ' environment .
	#15	a haunting tale of murder and mayhem .
	#16	neil burger here succeeded in ... making the mystery of four decades back the springboard for a more immediate mystery in the present .
negative	#1	nothing happens , and it happens to flat characters .
	#2	as lively an account as seinfeld is deadpan .
	#3	so we got ten little indians meets friday the 13th by way of clean and sober , filmed on the set of carpenter 's the thing and loaded with actors you 're most likely to find on the next inevitable incarnation of the love boat .
	#4	the plot is nothing but boilerplate cliches from start to finish , and the script assumes that not only would subtlety be lost on the target audience , but that it 's also too stupid to realize that they 've already seen this exact same movie a hundred times
	#5	ultimately , sarah 's dedication to finding her husband seems more psychotic than romantic , and nothing in the movie makes a convincing case that one woman 's broken heart outweighs all the loss we witness .
	#6	the big finish is a bit like getting all excited about a chocolate eclair and then biting into it and finding the filling missing .
	#7	this picture is mostly a lump of run-of-the-mill profanity sprinkled with a few remarks so geared toward engendering audience sympathy that you might think he was running for office – or trying to win over a probation officer .
	#8	just because a walk to remember is shrewd enough to activate girlish tear ducts does n't mean it 's good enough for our girls .
	#9	often lingers just as long on the irrelevant as on the engaging , which gradually turns what time is it there ?
	#10	this movie , a certain scene in particular , brought me uncomfortably close to losing my lunch .
	#11	but it would be better to wait for the video .
	#12	a rude black comedy about the catalytic effect a holy fool has upon those around him in the cutthroat world of children 's television .
	#13	just a collection of this and that – whatever fills time – with no unified whole .
	#14	although god is great addresses interesting matters of identity and heritage , it 's hard to shake the feeling that it was intended to be a different kind of film .
	#15	the chocolate factory without charlie .
	#16	in that setting , their struggle is simply too ludicrous and borderline insulting .