Prompt Consistency for Zero-Shot Task Generalization

Chunting Zhou[*1]**, Junxian He**[*1]**, Xuezhe Ma**[2]**, Taylor Berg-Kirkpatrick**[3]**, Graham Neubig**[1]

[1]Language Technologies Institute, Carnegie Mellon University
[2] Information Sciences Institute, University of Southern California
[3]Department of Computer Science and Engineering, University of California San Diego
{chuntinz,junxianh,gneubig}@cs.cmu.edu
xuezhema@isi.edu, tberg@eng.ucsd.edu

Abstract

One of the most impressive results of recent NLP history is the ability of pre-trained language models to solve new tasks in a *zero-shot* setting. To achieve this, NLP tasks are framed as natural language prompts, generating a response indicating the predicted output. Nonetheless, the performance in such settings often lags far behind its supervised counterpart, suggesting a large space for potential improvement. In this paper, we explore methods to utilize unlabeled data to improve zero-shot performance. Specifically, we take advantage of the fact that multiple prompts can be used to specify a single task, and propose to regularize *prompt consistency*, encouraging consistent predictions over this diverse set of prompts. Our method makes it possible to fine-tune the model either with extra unlabeled training data, or directly on test input at inference time in an unsupervised manner. In experiments, our approach outperforms the state-of-the-art zero-shot learner, T0 (Sanh et al., 2022), on 9 out of 11 datasets across 4 NLP tasks by up to 10.6 absolute points in terms of accuracy. The gains are often attained with a small number of unlabeled examples. [1]

1 Introduction

While the past decade has demonstrated that pre-trained language models (PLMs) are powerful tools for improving generalization from training datasets to test datasets (Devlin et al., 2019; Liu et al., 2019; Raffel et al., 2020), more recent work has shown that they can even perform *zero-shot generalization to new tasks* without any annotated examples (Brown et al., 2020; Wei et al., 2022; Sanh et al., 2022). These systems leverage natural language prompts that specify the task for the model and represent different tasks in a unified format (Liu

et al., 2021b). Zero-shot task generalization suggests a path towards generic systems that perform a wide variety of NLP tasks with no annotated examples. However, while enticing conceptually, zero-shot performance often remains relatively low compared to systems trained using even a small amount of task-specific labeled data.

In this paper, we examine methods to make PLMs better zero-shot learners using unlabeled text. Our work is motivated by consistency training methods that regularize model predictions to be invariant to perturbation (e.g. noise or paraphrasing) of the input examples. Consistency training is widely used in semi-supervised learning literature as an effective technique to utilize unannotated examples (Bachman et al., 2014; Sajjadi et al., 2016; Beyer et al., 2019; Xie et al., 2020a). It is often understood as a type of smoothness regularization or data augmentation (Xie et al., 2020a) and attains strong performance in semi-supervised learning. Instead of example-level consistency, we propose to regularize *prompt consistency*, where a model is regularized to make the same prediction across a diverse set of synonymous task prompts. Prompt consistency regularization makes sense intuitively since PLMs should be robust across synonymous prompts, whereas it is known that model predictions are empirically very sensitive to the wording of the task prompts (Jiang et al., 2020).

Specifically, we design a pairwise distillation loss that encourages consistency between every pair of prompts (Figure 1). We refer to our method as *swarm distillation*, and it has the advantage of being fully unsupervised, only requiring unannotated inputs. Notably, unannotated examples are often relatively easy to collect. Drafting several prompts for a task is also far cheaper than annotating labels for each example – in fact, there are already well-designed prompts available for a wide range of NLP tasks (Bach et al., 2022).

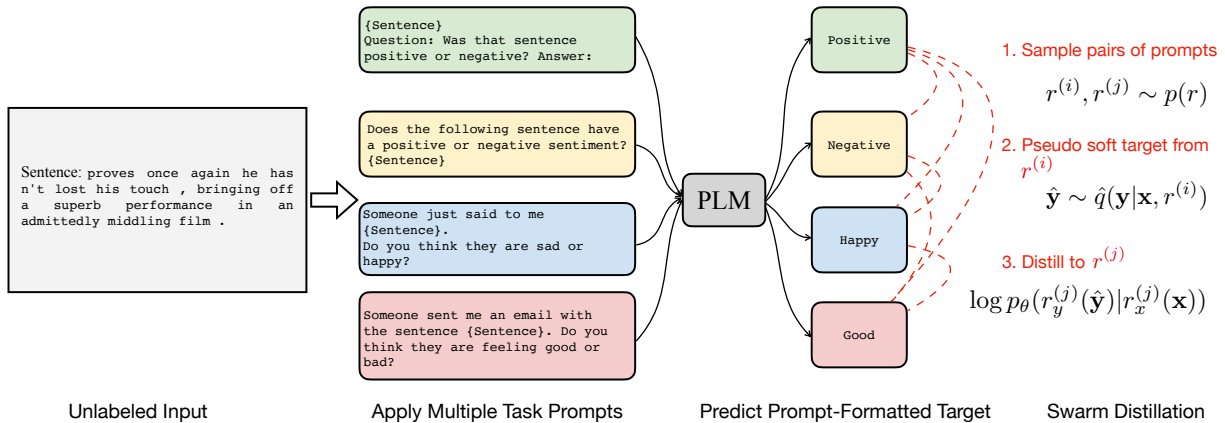Previous work on example-level consistency reg-

---

Figure 1: An example of the proposed approach in an sentiment classification task. We apply multiple synonymous prompts to the unlabeled example, then we regularize the consistency of the predictions from different prompts, through our swarm distillation loss as detailed in Eq. 2.

ularization typically minimizes a consistency loss along with a supervised loss in a semi-supervised setting (Miyato et al., 2018; Xie et al., 2020a). Recently, Elazar et al. (2021) performed experiments optimizing a prompt consistency loss in the context of a relation prediction task, also incorporating a supervised version of the masked language model pretraining objective. In contrast, we (1) optimize a novel prompt consistency loss alone, making our approach completely unsupervised and agnostic to the model's pretraining objective, and (2) experiment on and demonstrate the practicality of such an approach for a broad variety of NLP tasks. Notably, this unsupervised setting poses additional learning challenges: without explicit supervision, the model may suffer from catastrophic forgetting and even exhibit a form of collapse where the model always makes the same predictions for any input. To address this issue, we adopt two simple strategies: (1) we utilize parameter-efficient tuning techniques (Houlsby et al., 2019; He et al., 2022) to only update a small number of extra parameters, naturally mitigating catastrophic forgetting by fixing the original PLM parameters; (2) we propose an unsupervised criterion to select the model checkpoint before it falls into a collapsed local optimum.

In experiments, we build our method on top of a state-of-the-art zero-shot task learner, T0 (Sanh et al., 2022), and validate its performance on 11 datasets from 4 NLP tasks: natural language inference, coreference resolution, word sense disambiguation, and sentence completion. We perform experiments under two secenarios: (1) training the model with unlabeled training data; or (2) tuning the model with unlabeled test inputs directly. In

both settings, we show that our swarm distillation method improves the accuracy of the 3B-parameter T0 model on 9 out of 11 datasets by up to 10.6 absolute points. We further scale model size up to 11B parameters, and demonstrate that our approach outperforms the 11B-parameter T0 model on 4 out of 4 datasets. Remarkably, analysis implies that these gains are often possible with only tens of examples, suggesting a small computation overhead.

2 Prompt-based Zero-Shot Task Generalization

Given a task where the input is denoted as $\mathbf{x} \in \mathcal{X}$ and the goal is to predict $\mathbf{y} \in \mathcal{Y}$, we focus on the zero-shot task generalization setting: we aim to feed a PLM with $\mathbf{x}$ to predict $\mathbf{y}$, where the PLM is never trained on the specific task to be performed. Zero-shot task generalization goes beyond traditional dataset generalization, as the model must generalize to new functions $f : \mathcal{X} \rightarrow \mathcal{Y}$ as opposed to new input examples, $\mathbf{x}$. Recently, the development of prompting methods has advanced zero-shot task generalization by representing different tasks in a unified format (Liu et al., 2021b), and several prompt-based approaches have attained reasonable zero-shot performance (Brown et al., 2020; Sanh et al., 2022; Wei et al., 2022).

A prompt r consists of an input template r_x, an output template r_y, and metadata to re-format the original $\mathbf{x}$ and $\mathbf{y}$ into new prompt-formatted input and target, $r_x(\mathbf{x})$ and $r_y(\mathbf{y})$. For example, as shown in Figure 1, in a sentiment classification task where we must predict positive or negative sentiment of the text, the input includes the field Sentence and the target consists of the

field `Label`. An input template could be "Does the following sentence have a positive or negative sentiment? {Sentence}", and the target template is "Choices[{label}]". Here `Choices` is the metadata that is a list containing [`Positive`, `Negative`] to correspond to the numeric label ids. We note that such metadata is prompt-specific and can differ with different prompts for the same task – for instance, in Figure 1 the `Choices` list of the last prompt on the bottom is [`Good`, `Bad`]. In prompt-based approaches the PLM models the conditional probability $q(\mathbf{y}|\mathbf{x}, r)$ through $p_\theta(r_y(\mathbf{y})|r_x(\mathbf{x}))$ where θ denotes the model parameters. In classification tasks where $\mathcal{Y}$ is a finite label set, $q(\mathbf{y}|\mathbf{x}, r)$ is normalized over the possible labels at inference time to predict $\mathbf{y}$:

$$q(\mathbf{y}|\mathbf{x}, r) = \frac{p_\theta(r_y(\mathbf{y})|r_x(\mathbf{x}))}{\sum_{\mathbf{y}' \in \mathcal{Y}} p_\theta(r_y(\mathbf{y}')|r_x(\mathbf{x}))}. \quad (1)$$

In generation tasks where $\mathcal{Y}$ is an infinite sequence space, the target template is typically instantiated as the target itself, i.e. $p_\theta(r_y(\mathbf{y})|r_x(\mathbf{x})) = p_\theta(\mathbf{y}|r_x(\mathbf{x}))$, then the output can be directly decoded through sequence decoding approaches. Through designing such prompts for each task, all NLP tasks share the same data format, and models trained on one task may generalize to others.

3 Prompt Consistency Training

3.1 Problem Definition

In this paper, we aim to explore unannotated examples to improve prompt-based zero-shot task generalization. Formally, we are given an unlabeled dataset in the task of interest $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$, and we assume the dataset has K different prompts, $\{(r_x^{(1)}, r_y^{(1)}), \cdots, (r_x^{(K)}, r_y^{(K)})\}$. Our goal is to utilize these resources and adapt a PLM to predict $r_y(\mathbf{y})$ conditioned on $r_x(\mathbf{x})$. Unlabeled inputs are often available in practice, we consider two such scenarios in the paper.

First, we consider the case when unannotated examples from a non-test set are available. For many NLP tasks their inputs are plain text such as reviews, documents, or questions and can be easily collected (less so for other NLP tasks, like natural language inference the inputs are paired hypotheses and premises that can be non-trivial to obtain automatically). In this paper, we test this setting by utilizing the inputs of the training dataset. This is similar to Schick and Schütze (2021a,b) where

they directly use the inputs of the training split as unlabeled resources to help few-shot learning.

Second is the case when unannotated test inputs are available. This is almost always true for any task. We use the test split to mimic the setting. While the limited number of unlabeled examples could potentially limit the effectiveness of some unsupervised learning methods, we show in §4.4 that our method is effective even with tens to hundreds of unlabeled examples.

On the other hand, a diverse set of prompts is not exceedingly difficult to collect practically – drafting prompts for each task is easier than annotating labels for many examples. In fact, the community efforts have pushed out a Public Pool of Prompts (P3) that contains thousands of prompts for hundreds of NLP datasets already (Bach et al., 2022).

3.2 The Prompt Consistency Loss

Consistency regularization is a method that creates different views (e.g. paraphrases of text) of the input and regularizes the outputs to be close to each other, and has achieved significant success in semi-supervised learning (Clark et al., 2018; Xie et al., 2020a,b). While previous methods use an additional module to perturb each example and then optimize example-level consistency, we propose to optimize prompt-level consistency which (1) is conceptually simple, and (2) can mitigate the fact that the predictions of PLMs are typically inconsistent with different prompts for the same task (Jiang et al., 2020; Elazar et al., 2021). Intuitively, we propose to regularize the predictions of different prompts for a given input to be close to each other, using a pairwise distillation loss to draw the predictions from one prompt closer to those from the other. Concretely, we randomly sample a few pairs of prompts and distill the pseudo target $\hat{\mathbf{y}}$ from one prompt $r^{(i)}$ to the other prompt $r^{(j)}$, as illustrated in Figure 1. The loss function is defined as:

$$\begin{aligned} \mathcal{L} = -\mathbb{E}_{\mathbf{x} \sim p_d(\mathbf{x})} \mathbb{E}_{r^{(i)}, r^{(j)} \sim p(r)} \\ \mathbb{E}_{\hat{\mathbf{y}} \sim \hat{q}(\mathbf{y}|\mathbf{x}, r^{(i)})} \log p_\theta(r_y^{(j)}(\hat{\mathbf{y}})|r_x^{(j)}(\mathbf{x})), \end{aligned} \quad (2)$$

where $p_d(\mathbf{x})$ is the empirical data distribution, $p(r)$ is a uniform distribution over possible prompts, and $\hat{q}(\mathbf{y}|\mathbf{x}, r)$ is the conditional target distribution defined as in Eq. 1 but with a stopping gradient operator. We do not propagate gradients to $\hat{q}(\mathbf{y}|\mathbf{x}, r^{(i)})$ following Miyato et al. (2018) and Xie et al. (2020a).[2] Stopping the gradient of one side

---

[2]Note that $\hat{q}(\mathbf{y}|\mathbf{x}, r)$ still changes as we train the model.

in a pairwise consistency loss is also shown to help mitigate the collapse issue where all inputs lead to the same predictions (Chen and He, 2021). Different from traditional distillation that distills from a teacher model to a student model (Hinton et al., 2015), or previous consistency training that a single teacher distills to several students (Clark et al., 2018; Xie et al., 2020a), we perform distillation among a swarm of prompts where each prompt is a teacher and student at the same time, thus we term our method as *swarm distillation*. In our implementation, we approximate the expectation over the paired prompts $(r^{(i)}, r^{(j)})$ with k randomly sampled pairs for training efficiency.

Prompt consistency is related to example-level consistency when viewing different prompt-formatted inputs $r_x^{(i)}(\mathbf{x})$ as separated views of the same example, thus our swarm distillation approach shares spirit with previous work on example-level consistency training and can be understood similarly from the perspective of unsupervised data augmentation, smoothness regularization, or label propagation (Xie et al., 2020a). In this paper, we focus on classification tasks where $\mathcal{Y}$ is a finite label set, while Eq. 2 can be directly applied to sequence generation tasks as well with sequence distillation (Kim and Rush, 2016).

Our approach differs from previous consistency training methods which often combine an unsupervised consistency loss with a supervised loss in a semi-supervised setting (Miyato et al., 2018; Clark et al., 2018; Xie et al., 2020a). Elazar et al. (2021) try to improve prompt consistency for a relation filling task with a pairwise two-sided KL divergence loss, while they also optimize a supervised version of the original PLM objective that turns out to be important. In contrast, our approach minimizes the swarm distillation loss in Eq. 2 alone, and therefore is completely unsupervised and agnostic to the pre-training objective. However, this setting also poses challenges in learning, which we discuss next.

3.3 Training

Being trained without explicit supervision, the PLM may forget what it learns during pretraining since the unsupervised consistency loss is different from the pretraining objective. Also, we note that prompt consistency may be achieved with a trivial solution – if the predictions from each example and each prompt collapse to the same label then maximal consistency among prompts can be
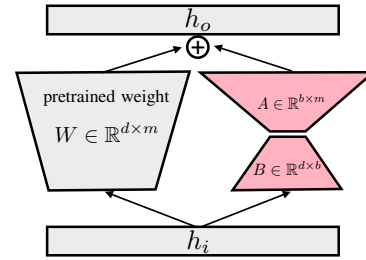

Figure 2: A diagram of LoRA in the FFN sublayer. Only the LoRA parameters, A and B, are updated during training.

reached. To mitigate such catastrophic forgetting and collapse issues, we propose two techniques:

Parameter-efficient tuning: It has recently been observed that updating a small number of added parameters in a PLM is able to achieve comparable performance to tuning all the parameters (Houlsby et al., 2019; Li and Liang, 2021; Hu et al., 2022; He et al., 2022). Parameter-efficient tuning methods naturally mitigate catastrophic forgetting and collapse through fixing the original PLM parameters. Specifically, we use LoRA (Hu et al., 2022), a low-rank adaptation method for PLMs. As shown in Figure 2, LoRA learns a low-rank approximation of the pretrained matrix updates: given a pretrained weight matrix $W \in \mathbb{R}^{d \times m}$, LoRA learns to update it as $W \leftarrow W + \alpha BA$, where $B \in \mathbb{R}^{d \times b}, A \in \mathbb{R}^{b \times m}$ are low-rank matrices and α is a hyperparameter, and only B and A are updated during training. $b \ll d$ is referred to as the *bottleneck dimension*. Following He et al. (2022), we apply LoRA to the feed-forward weight matrices of every layer in the pretrained transformer (Vaswani et al., 2017) model. In our preliminary experiments, we found that LoRA is less likely to suffer from collapse, while on some datasets the model still collapses in the end even though it learns well in the middle. This motivates us to develop a criterion to select the model checkpoint before it falls into a collapsed local optimum, which we describe next.

Unsupervised model selection criterion: Our zero-shot setting does not have labeled validation data for model selection, and the swarm distillation objective is not an ideal selection criterion since it is minimized at collapse. Therefore, we would like to have an unsupervised criterion that encourages consistency but simultaneously penalizes collapse. With that in mind, we focus on Fleiss' kappa (Fleiss, 1971), a commonly used metric to assess the reliability of agreement. In our setting,

Fleiss' kappa expresses the extent to which the amount of agreement among prompts exceeds what would be expected if all prompts made their predictions according to the marginalized distribution of labels. This design computes a notion of "relative consistency" and naturally penalizes collapse. Formally, let n_{ij} be the number of prompts that predict the j-th label for the i-th example. There are a total of NK predictions where N is the number of examples and K is the number of prompts. Given an example $\mathbf{x}_i$, the agreement probability p_i computes the normalized number of agreeing prompt pairs:

$$p_i = \sum_j n_{ij}(n_{ij} - 1)/(K(K-1)), \quad (3)$$

then the "absolute consistency" $\bar{P}$ is:

$$\bar{P} = \sum_{i=1}^{N} p_i/N. \quad (4)$$

$\bar{P}$ is maximized in the case of collapse. However, Fleiss' kappa considers the marginalized distribution of labels: how likely are two prompts consistent if they make predictions randomly according to the marginalized label distribution? This chance probability $\bar{P}_e$ is:

$$\bar{P}_e = \sum_j q_j^2, \qquad q_j = \sum_{i=1}^{N} n_{ij}/(NK), \quad (5)$$

where q_j represents the marginalized distribution of labels, i.e. $p(\mathbf{y} = j)$. $\bar{P}_e$ is large when collapse happens and one label dominates in the entire corpus. Finally, Fleiss' kappa is computed as:

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e}, \quad (6)$$

where $1 - \bar{P}_e$ gives the degree of consistency that is attainable above chance, $\bar{P} - \bar{P}_e$ gives the degree of consistency actually achieved above chance. κ ranges from -1 to 1. Eq. 6 naturally penalizes collapse, and in our experiments, we always observe a monotonic decrease of κ when collapse happens. Therefore, we select the model checkpoint after which κ monotonically decreases.[3] We emphasize that we perform validation on the data that the model is trained on and do not require an additional development dataset. We include ablation analysis for both LoRA and the model selection components in Appendix C that shows that they are important for the success of our method.

---

[3]In most of the settings, this criterion is equivalent to using maximal κ as the criterion, except for few cases where the beginning of training exhibits large fluctuations in κ.

4 Experiments

Our experiments below are designed to (1) measure whether swarm distillation is able to improve zero-shot task generalization; and (2) analyze how much resource (number of prompts and unlabeled examples) our method demands.

4.1 General Setup

Datasets: Following Sanh et al. (2022), we evaluate our method on 11 NLP datasets across 4 unseen tasks. They are (1) natural language inference: ANLI (Nie et al., 2020) (there are three versions of ANLI with different levels of difficulty, which we denote as ANLI R1/R2/R3), CB (De Marneffe et al., 2019), RTE (Wang et al., 2019a); (2) sentence completion: COPA (Roemmele et al., 2011), HellaSwag (Zellers et al., 2019), Story Cloze (Mostafazadeh et al., 2016); (3) coreference resolution: WSC, Winogrande (Levesque et al., 2012); and (4) word sense disambiguation: WIC (Pilehvar and Camacho-Collados, 2019). We access them using Hugging Face Datasets (Lhoest et al., 2021) and most of them are from the SuperGLUE benchmark (Wang et al., 2019a). All of these datasets are classification-based, predicting a discrete label from a finite set. Each of these datasets has a diverse set of prompts provided by the Public Pool of Prompts (Sanh et al., 2022) The number of prompts ranges from 4 to 15. Please refer to Appendix A for detailed statistics of these datasets.

Setup: We build our method on top of the PLM T0 (Sanh et al., 2022). T0 is an adapted version of the pretrained T5 model (Raffel et al., 2020) that is continually trained on multiple tasks with supervised, prompt-formatted examples. T0 outperforms GPT3 (Brown et al., 2020) and demonstrates state-of-the-art performance in zero-shot task generalization. All the tasks that we are studying are not included in T0's training data. We focus our major study on the T0 model version with 3 billion parameters (T0-3B), while we also include results using the largest T0 model with 11 billion parameters (T0-11B) on some datasets, due to the high computational cost of training T0-11B. The hyperparameters (e.g. the optimization hyperparameters) are tuned on the RTE dataset with its validation set and fixed for all other datasets. We use a bottleneck dimension of 1 for LoRA. Complete setup details can be found in Appendix B.

Task	Dataset	T0-3B		Self Dist. (train)		Swarm Dist. (train)		Swarm Dist. (test)	
		Ens.	Med.	Ens.	Med.	Ens.	Med.	Ens.	Med.
NLI	RTE	64.6	64.1	$64.9_{\pm0.2}$	$63.8_{\pm0.1}$	$75.2_{\pm0.8}$ ↑10.6	$73.9_{\pm0.8}$ ↑9.8	$75.2_{\pm0.2}$ ↑10.6	$73.5_{\pm0.1}$ ↑9.4
	CB	46.4	50.0	$47.0_{\pm1.0}$	$49.4_{\pm2.7}$	$47.6_{\pm1.0}$ ↑1.2	$48.2_{\pm0.0}$ ↓1.8	$46.4_{\pm0.0}$ ↑0.0	$48.8_{\pm1.0}$ ↓1.2
	ANLI R1	34.6	33.7	$36.1_{\pm0.1}$	$34.7_{\pm0.1}$	$38.4_{\pm0.5}$ ↑3.8	$35.7_{\pm0.4}$ ↑2.0	$38.5_{\pm0.3}$ ↑3.9	$35.7_{\pm0.5}$ ↑2.0
	ANLI R2	33.7	33.4	$35.3_{\pm0.1}$	$33.2_{\pm0.2}$	$37.9_{\pm0.8}$ ↑4.2	$36.6_{\pm0.5}$ ↑3.2	$37.7_{\pm0.2}$ ↑4.0	$35.4_{\pm0.4}$ ↑2.0
	ANLI R3	34.7	33.3	$33.1_{\pm0.0}$	$33.8_{\pm0.2}$	$34.0_{\pm0.3}$ ↓0.7	$34.6_{\pm0.1}$ ↑1.3	$34.1_{\pm0.2}$ ↓0.6	$33.5_{\pm0.0}$ ↑0.2
Compl.	COPA	78.0	79.0	$82.3_{\pm0.6}$	$78.2_{\pm0.3}$	$82.7_{\pm0.6}$ ↑4.7	$79.0_{\pm0.5}$ ↑0.0	$83.0_{\pm1.0}$ ↑5.0	$79.7_{\pm0.6}$ ↑0.7
	HellaSwag	27.8	27.5	$32.5_{\pm0.2}$	$32.7_{\pm0.3}$	$34.2_{\pm0.2}$ ↑6.4	$33.4_{\pm0.2}$ ↑5.9	$33.7_{\pm0.6}$ ↑5.9	$33.2_{\pm0.3}$ ↑5.7
	Story Cloze	86.5	85.1	$89.6_{\pm0.0}$	$88.7_{\pm0.0}$	–	–	$87.3_{\pm0.1}$ ↑0.8	$86.9_{\pm0.2}$ ↑1.8
Coref.	Wino.	50.9	50.5	$51.1_{\pm0.1}$	$50.7_{\pm0.1}$	$52.0_{\pm0.3}$ ↑1.1	$51.4_{\pm0.0}$ ↑0.9	$52.1_{\pm0.3}$ ↑1.2	$51.2_{\pm0.2}$ ↑0.7
	WSC	69.2	64.4	$69.2_{\pm0.0}$	$64.6_{\pm0.3}$	$58.3_{\pm1.1}$ ↓10.9	$59.3_{\pm2.0}$ ↓5.1	$57.7_{\pm0.0}$ ↓11.5	$58.8_{\pm0.6}$ ↓5.6
WSD	WIC	50.3	50.4	$50.3_{\pm0.0}$	$50.3_{\pm0.0}$	$55.4_{\pm1.1}$ ↑5.1	$54.4_{\pm0.7}$ ↑4.0	$55.5_{\pm0.8}$ ↑5.2	$54.8_{\pm0.5}$ ↑4.4

Table 1: Accuracy results on the validation set of 11 NLP datasets based on the T0-3B model. Swarm Distillation (train) and Swarm Distillation (test) use the unlabeled training split and validation split of datasets to train the model respectively, corresponding to training-time and test-time tuning. The Story Cloze dataset does not have a training split and its self distillation results are from tuning on the validation split. We report the mean and std across 3 random runs, and also denote the absolute accuracy change compared to the T0-3B baseline.

4.2 Evaluation

Metrics: We use accuracy as the metric for all datasets. We report two different types of accuracy given that we have multiple prompts. The *ensemble accuracy* (Ens.) averages the output distributions of multiple prompts and makes predictions according to it. Ensembling multiple prompts has been explored before and found superior to using a single prompt (Jiang et al., 2020; Qin and Eisner, 2021). The *median accuracy* (Med.) within the set of prompts serves as a proxy for the expected performance when users specify a single prompt and input a prompt-formatted example. As our approach assumes availability of a set of prompts for the downstream task, and it is relatively cheap to craft several prompts for a task, ensemble prediction is the better option given input x, and it does empirically yield higher accuracy overall than the median for both the baseline and our method. Therefore, we will report both numbers but mainly discuss ensemble accuracy. We report these metrics on the validation split of each dataset. We run the experiments with 3 random seeds and report the mean and standard deviation.

Evaluation scenarios: We provide our methods with different unlabeled sources which lead to two practical scenarios during evaluation: (1) *training-time tuning*: we use the unlabeled training split from the corresponding dataset to train the model. This is similar to traditional settings where training and test data are different; and (2) *test-time tuning* (Sun et al., 2020; Wang et al., 2021): we directly adapt the PLM on the test data. This setting is reasonable, as we will always have access to the test inputs at test time. Intuitively, the unlabeled test sample x often provides hints about the distribution it was drawn, suggesting that we may update the model before making the prediction. This scenario is attractive since it alleviates the common distribution mismatch issue when there is a distribution shift between the training and test data. Compared to training-time tuning, test-time tuning typically uses less unlabeled data in our experiments since it uses the validation split itself. In the major experiments, we focus on the offline test-time tuning where we assume access to the entire test data[4] and train our approach on all test examples, while in §4.4 we will discuss the potential for online adaptation where data arrives in a stream.

Baselines: As far as we know, there is no prior work studying unsupervised approaches for this prompt-based task generalization setting, thus T0 is the main baseline that we compare our approach against. However, we still implement an ablation baseline, self distillation, to separate the improvement from optimizing prompt consistency and pseudo-label distillation. Specifically, self distillation minimizes the same loss as in Eq. 2 but with $r^{(i)} = r^{(j)}$ – instead of pairwise distillation, the prompt always distills its own prediction to itself. This baseline can be viewed as a prompt

---

[4]To clarify, test data is not the test split of the dataset, but the data that we evaluate on, i.e. the validation split.

	T0-11B		Swarm Dist.	
Dataset	Ens.	Med.	Ens.	Med.
WSC	63.5	62.5	65.4 ↑1.9	62.0 ↓0.5
RTE	83.8	82.0	86.6 ↑2.8	85.0 ↑3.0
HellaSwag	34.4	33.6	45.0 ↑10.6	43.0 ↑9.4
WIC	57.2	56.8	62.1 ↑4.9	60.7 ↑3.9

Table 2: Accuracy based on T0-11B.

	T0-3B	No Shift	SD on MNLI	SD on QNLI
RTE	64.6	75.2	75.5	72.9
ANLI R1	34.6	38.4	38.2	37.4
ANLI R2	33.7	37.9	38.5	35.3
ANLI R3	34.7	34.0	36.9	34.4

Table 3: Ensemble accuracy on the distribution shift setting based on T0-3B. "No Shift" represents the original setting where we train the swarm distillation loss on the training split from the same dataset as the test examples. "SD on MNLI/QNLI" represents swarm distillation trained on the training split of MNLI/QNLI.

version of self-training, which has proven to effectively utilize unlabeled data (He et al., 2020; Xie et al., 2020b; Zhang et al., 2020). We report self distillation results in the training-time tuning setting only for simplicity.

4.3 Results

How well does swarm distillation work? We first compare swarm distillation against the T0-3B baseline. As shown in Table 1, the ensemble accuracy of swarm distillation exceeds the T0-3B baseline on 9 out of 11 datasets in both training- and test-time tuning settings. Particularly, our approach improves the zero-shot performance on RTE by around 10 absolute points in all cases. Our approach slightly hurts ensemble accuracy of ANLI R3 and median accuracy of CB, but is overall comparable on these two datasets. Compared to self distillation, swarm distillation outperforms it on 9 out of 11 datasets in terms of ensemble accuracy, by up to 10.3 absolute points. These results further confirm the effectiveness of encouraging prompt consistency. We note that swarm distillation severely fails on WSC with a 10-point accuracy decrease compared to both T0 and self distillation, this is because Fleiss' kappa selects a bad model checkpoint, while our approach actually improves the performance on WSC in the middle of training as we will discuss more in §4.4. Notably, our approach is helpful on several datasets where T0-3B only shows nearly chance accuracy, such as

ANLI R1/R2/R3 (3 labels), HellaSwag (4 labels), Winogrande (2 labels), and WIC (2 labels). In addition, we observe that swarm distillation in the test-time tuning setting performs comparably well to the training-time one despite using much less training data, as shown in Appendix A. It is worth noting that prompt-based zero-shot task generalization is challenging, for example, T0 with even 11 billion parameters reports a median accuracy of only ∼ 40 on ANLI R1/R2/R3, 33.7 on HellaSwag, and 57.2 on WIC (Sanh et al., 2022). Our results are surely still far from satisfactory, yet we hope to inspire future research to explore unlabeled data to build better zero-shot learners.

Scaling to 11B parameters: We now evaluate our method based on the largest version of T0 model, T0-11B. T0-11B is a very powerful zero-shot baseline that greatly outperforms GPT3. Due to the expensive computation to train T0-11B, we use one dataset per task, a total of 4 datasets as our benchmark, and only run with one random seed in the test-time tuning setting. Results are shown in Table 2. Swarm distillation outperforms T0-11B on all 4 datasets in terms of ensemble accuracy, and notably, improves the ensemble accuracy on HellaSwag from 34.4 to 45.0 without any annotation. Table 1 and Table 2 demonstrate the effectiveness of swarm distillation across different model sizes.

Robustness of swarm distillation: The main experiments so far train and test on examples from the same dataset, thus the results are based on the assumption that the training and test distributions are similar. To relax this assumption, here we evaluate swarm distillation with a more difficult setting by collecting unlabeled training examples from other datasets that are not listed in §4.1. Specifically, we focus on the NLI task, train swarm distillation on the unlabeled training split of MNLI (Williams et al., 2018) and QNLI (Wang et al., 2019b) respectively, and evaluate it on RTE and ANLI. Table 3 shows that our proposed method still performs well in such a distribution shift setup — the models trained on MNLI or QNLI help improve the T0-3B baseline in most cases. The results from MNLI training are generally comparable to the original "No Shift" numbers, while QNLI training causes a mild accuracy drop.

4.4 Analysis

Are predictions more consistent across prompts after swarm distillation? We wonder whether

	RTE	CB	ANLI R1	ANLI R2	ANLI R3	COPA	HS	Story.	Wino.	WSC	WIC	Avg.
T0-3B	0.644	0.440	0.221	0.189	0.170	0.586	0.164	0.765	0.396	0.255	0.398	0.384
Swarm Dist.	0.662	0.254	0.145	0.156	0.177	0.699	0.402	0.862	0.509	0.462	0.517	**0.440**

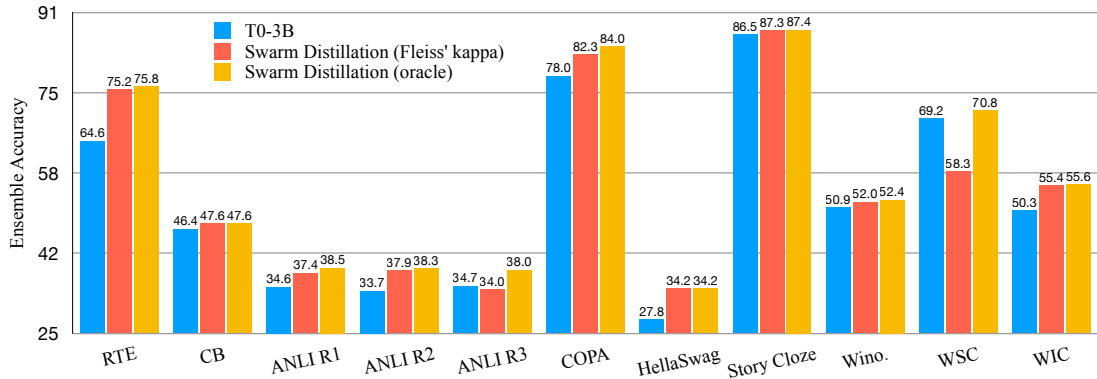Table 4: Fleiss' kappa on 11 datasets based on T0-3B. Swarm distillation is trained on the training split of the respective dataset.

Figure 3: Analysis results to compare the model checkpoints selected by the unsupervised criterion Fleiss' kappa with the oracle model checkpoints selected by validation accuracy.

the gains of swarm distillation are attained together with more consistent predictions across different prompts. To this end, we report Fleiss' kappa, a commonly used metric for group agreement as detailed in §3.3. Results are shown in Table 4. Fleiss' kappa on 8 out of 11 datasets increases after swarm distillation, which boosts the averaged Fleiss' kappa of T0-3B by 14.6% relatively. This implies that swarm distillation facilitates prompt consistency, and potentially improves the robustness of PLMs to different wording of prompts.

Does the unsupervised criterion select the best model checkpoint? In §3.3, we discussed using Fleiss' kappa to select the best model checkpoint for evaluation, here we report the oracle accuracy numbers obtained by selecting the model checkpoint with the best validation accuracy, and compare it to the one selected by Fleiss' kappa. We compare the ensemble accuracy using T0-3B in the training-time tuning setting, with results in Figure 3. On most of the datasets, Fleiss' kappa is able to achieve numbers close to the best ones. On all 11 datasets, our oracle number outperforms the T0-3B baseline. In Table 1 we show that swarm distillation hurts the performance on WSC a lot, while in Figure 3 swarm distillation (oracle) in fact outperforms T0-3B, implying that the issue lies on model selection. Therefore, swarm distillation could potentially work better if an annotated dev set is available or when it is combined with other tech-

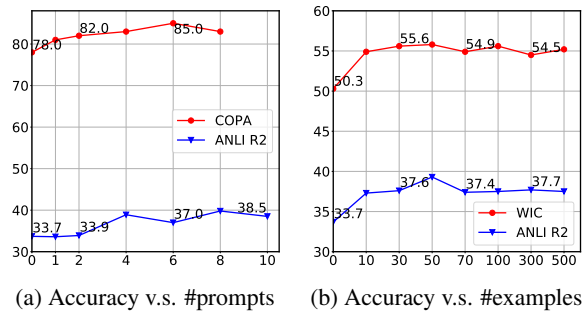(a) Accuracy v.s. #prompts (b) Accuracy v.s. #examples

Figure 4: Ensemble accuracy of swarm distillation on three example datasets based on T0-3B, demonstrating the effect of prompt size and unlabeled data size.

niques in few-shot learning settings, where good checkpoints may be selected out more easily.

How many prompts do we need? Our approach requires a diverse set of prompts to regularize prompt consistency. Here we perform ablation experiments to understand the effect of the number of prompts on the performance. We take COPA and ANLI R2 as example datasets which have 8 and 15 prompts, respectively. We then vary the number of available prompts by randomly sampling a subset of prompts before training. We report the ensemble accuracy of swarm distillation (train) in Figure 4a. On both COPA and ANLI R2, we observe gains as we increase the number of prompts from 0 (the baseline), yet the performance saturates very quickly and relatively stabilizes when we provide 4 prompts. This implies that swarm

distillation is not prompt-hungry and could work well with a small number of prompts. We note the with one prompt here Eq. 2 degenerates to a weaker version of self distillation compared to the one in Table 1 – self distillation in Table 1 utilizes all prompts during training while we assume access to only one prompt here.

How many unlabeled examples do we need? We measure the effect of unlabeled data size. Specifically, we randomly sample a subset of examples from the train split for training and report results on the entire validation dataset. Results on WIC and ANLI R2 are shown in Figure 4b. Notably, swarm distillation is able to outperform the baselines (#examples=0) by a large margin on both datasets with only 10 unlabeled examples, and the performance starts to saturate quickly afterward. These results suggest that swarm distillation is not data-hungry and works reasonably well with few *unlabeled* examples, allowing swarm distillation to remain as a relatively light approach while typical unsupervised training (e.g. pretraining) often requires a large amount of data and computation. Also, we argue that the phenomenon demonstrated in the results implies that swarm distillation may be applied to the online setting of test-time tuning, where the batches of test data arrive in a stream. Online test-time tuning is a practical setting in real life, and we leave it as future work to study.

5 Discussion

In this paper, we explore prompt consistency regularization to make PLMs better zero-shot learners. Our approach, swarm distillation, utilizes unlabeled examples to attain zero-shot gains. While we use swarm distillation in a post-adaptation manner, it could be potentially combined with the pretraining objectives in the pretraining stage (e.g. the multi-prompt training loss (Sanh et al., 2022; Wei et al., 2022)), or even with annotated data in few-shot learning settings. Combining swarm distillation with these other losses may easily bypass the model collapse issue since the other loss typically discourages the collapsed local optimum.

Acknowledgement

Limitations

There are two limitations of our work: (1) Because our method is operated in a fully unsupervised manner, there is no supervised development data for us to either select the best model or tune hyperparameters. Thus, we propose to use Fleiss' Kappa as our unsupervised development metric for model selection, which attains decent performance in most cases. However, we also observe on very few datasets that the proposed metric fails to select the best checkpoints and hurt the model's performance. As discussed in §4.4, our method can be combined with few-shot learning where a few labeled data are provided, and we believe this can largely alleviate the issues of model selection in the unsupervised setting. (2) The other limitation and at the same time an advantage of our method is that the proposed method can work well even with 10 unlabeled data points. This certainly makes our method a good candidate for the online setting where batches of test data come in a stream. However, as we discussed in §4.4, the performance of our model saturates quickly as we increase the number of unlabeled data, which means the performance of our method cannot scale well with tons of unlabeled data like self-supervised pretraining. As discussed in §5, we expect combining our method with few-shot learning setting / pre-training can lead to further improvements as the supervised signals may guide the model to a better local optimum.

Ethics Statement

Similar to T0, this work aims to produce an open-ended system that could perform all text-based tasks through designing different prompts. While the performance of GPT3, T0, and this work is far from the practical level on unseen tasks, we expect a greatly improved prompt-based system in the future could be built to help perform many daily tasks in real life. However, the resulted model in this paper also admits the same ethics concerns that T0 has. For example, the unrestricted use of prompts may easily trigger offensive generations or private information leakage, and how to fix unwanted LM behaviors is still an active research problem (Liu et al., 2021a; Perez et al., 2022).

References

Stephen H. Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht

Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan Fries, Maged S. Al-shaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Xiangru Tang, Mike Tian-Jian Jiang, and Alexander M. Rush. 2022. Promptsource: An integrated development environment and repository for natural language prompts.

Philip Bachman, Ouais Alsharif, and Doina Precup. 2014. Learning with pseudo-ensembles. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3365–3373.

Lucas Beyer, Xiaohua Zhai, Avital Oliver, and Alexander Kolesnikov. 2019. S4L: self-supervised semi-supervised learning. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 1476–1485. IEEE.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Xinlei Chen and Kaiming He. 2021. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925, Brussels, Belgium. Association for Computational Linguistics.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23, pages 107–124.

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

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2020. Revisiting self-training for neural sequence generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.

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

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

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

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021a. DExperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6691–6706, Online. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021b. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ArXiv preprint*, abs/2107.13586.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv preprint*, abs/1907.11692.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California. Association for Computational Linguistics.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online. Association for Computational Linguistics.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. 2022. Red teaming language models with language models. *ArXiv preprint*, abs/2202.03286.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. 2021. Zero-offload: Democratizing billion-scale model training. *arXiv preprint arXiv:2101.06840*.

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

Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. 2016. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1163–1171.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker,

Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021b. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. 2020. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 9229–9248. PMLR.

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

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3261–3275.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno A. Olshausen, and Trevor Darrell. 2021. Tent: Fully test-time adaptation by entropy minimization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Qizhe Xie, Zihang Dai, Eduard H. Hovy, Thang Luong, and Quoc Le. 2020a. Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Qizhe Xie, Minh-Thang Luong, Eduard H. Hovy, and Quoc V. Le. 2020b. Self-training with noisy student improves imagenet classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10684–10695. IEEE.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Yu Zhang, James Qin, Daniel S Park, Wei Han, Chung-Cheng Chiu, Ruoming Pang, Quoc V Le, and Yonghui Wu. 2020. Pushing the limits of semi-supervised learning for automatic speech recognition. *ArXiv preprint*, abs/2010.10504.

Task	Dataset	#train set	#validation set	#labels	#prompts
NLI	RTE	2,490	277	2	10
	CB	250	57	3	15
	ANLI R1	16,946	1000	3	15
	ANLI R2	45,460	1000	3	15
	ANLI R3	100,459	1200	3	15
Compl.	COPA	400	100	2	8
	HellaSwag	39,905	10,042	4	4
	Story Cloze	-	1,871	2	5
Coref.	Winogrande	40,398	1,267	2	5
	WSC	554	104	2	10
WSD	WIC	5,428	637	2	10

Table 5: Statistics of the datasets

A Datasets

We present the statistics of the 11 datasets in Table 5. For the training-time tuning scenario, we use up to 10,000 data points from the training set for training if the train set contains more than 10,000 data points.

B Experimental Setup

B.1 LoRA Setup

We use LoRA (Hu et al., 2022) as our parameter-efficient tuning model and set the bottleneck dimension of LoRA weight matrices to be 1 for both 3B and 11B models. We emphasize that the linear mapping matrix B (or A) in LoRA needs to be initialized as a zero matrix to ensure the output distribution after adding LoRA layers is the same as the original PLM before training, otherwise, the zero-shot ability of PLMs would be broken upon initialization and there is no supervision to learn it back. For both models, we set the dropout probability for the the LoRA intermediate representations to be 0.3. Let α denote the scaling factor of LoRA that is used to scale the output of the LoRA layer before adding to the hidden states of the pre-trained model. We set α to be 4 and 2 respectively for the 3B and 11B model. The peak learning rates of the 3B and 11B models are set to be 3e-5 and 5e-5 respectively with a warm-up stage of 100 steps and polynomial learning rate scheduler. We train for a maximum of 1,500 steps. Note that the hyperparameters for the 3B model is tuned on the RTE dataset and used for other datasets. We did not tune the hyperparameters of the 11B model.

B.2 Implementation Details

The reported T0 baseline numbers are obtained from our own running using the released T0 weights. We are able to reproduce the numbers

reported in Sanh et al. (2022), except for COPA where our T0 median number is higher than the originally reported one.

During training, at each update we first sample one input example $\mathbf{x}$ and apply all the prompts to reformat it as $r_x^1(\mathbf{x}), \cdots, r_x^K(\mathbf{x})$, then we perform inference for them and randomly shuffle the predictions. Next we iterate over them with a batch size of 5/10 (3B/11B)[5] and use the shuffled predictions to supervise them to compute the distillation loss, this implements the swarm distillation mechanism in Eq. 2 and amounts to approximating the expectation over paired prompts with K random pairs. We accumulate the gradients for 16 steps for one update so that each gradient descent is computed from 16 data examples. And we use 1 A40 GPU (45GB memory) to train the 3B model and 4 A40 GPUs with DeepSpeed Zero-2 (Ren et al., 2021) to train the 11B model. In general, training converges pretty fast and takes around 1 - 3 GPU hours for the 3B model and 2 - 6 hours for the 11B model depending on early stop points of different datasets. We use Adam (Kingma and Ba, 2015) as the optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 1e - 6$.

For the Transformer (Vaswani et al., 2017) models with model dimension d, the feed-forward intermediate dimension m and number of layers l, the additional parameters used in LoRA with bottleneck dimension b is calculated as $b*(m+d)*2*l*2$. As we set b to be 1 for both the 3B and 11B models, the additional number of LoRA parameters is 1,671,168 for the T0-3B model ($d = 1024, m = 16384, l = 24$) and 6,389,760 for the T0-11B model ($d = 1024, m = 65536, l = 24$).

C Ablation on LoRA and Model Selection

We report the ablation results on LoRA and unsupervised model selection in Table 6. Full fine-tuning hurts the T0-3B performance on all datasets – actually it collapses on almost all the datasets when we check its predictions, which could partially explain the low accuracies. Using LoRA alone is able to improve full fine-tuning generally and outperforms the T0-3B baseline sometimes. Moroeever, we find that unsupervised model selection is very effective to mitigate collapse and greatly improves full fine-tuning results. Finally, combining LoRA and unsupervised model selection gives the best

---

[5]Because the GPU memory sometimes cannot handle all the prompts within one batch.

	RTE	CB	ANLI R1	ANLI R2	ANLI R3	COPA	HS	Story.	Wino.	WSC	WIC
T0-3B	64.6	46.4	34.6	33.7	34.7	78.0	27.8	86.5	50.9	69.2	50.3
Full fine-tuning	59.6	8.9	33.3	33.5	33.3	54.0	25.7	51.5	50.4	36.5	50.0
+ LoRA	54.0	44.6	33.3	33.3	34.4	82.7	33.6	87.4	52.0	36.5	50.0
+ model selection	75.8	35.7	37.0	33.5	33.3	80.0	32.2	86.5	50.8	71.2	54.6
+ LoRA + model selection	75.2	47.6	38.4	37.9	34.0	82.7	34.2	87.3	52.0	58.3	55.4

Table 6: Ablation results on LoRA and unsupervised model selection. The training objective is the swarm distillation loss. Numbers are ensemble accuracy in the training-time tuning setting based on T0-3B. "Full fine-tuning" updates all the model parameters, while "+LoRA" means that we freeze the T0 parameters and only update the LoRA parameters.

results overall on these 11 datasets.

We clarify that the results in Table 6 are only for analysis purpose to better understand the effect of different components of our method, but were not used by us to make design decisions in our preliminary experiments– as stated in §3.3, we use LoRA because it collapses less often than full fine-tuning and develop a unsupervised model selection criterion since LoRA still collapses on some datasets. To judge collapse or not, we simply checked the model predictions, to see if the predictions for all the examples are almost the same.