Self-training Improves Pre-training for Few-shot Learning in Task-oriented Dialog Systems

Fei Mi[1], **Wanhao Zhou**[2], **Fengyu Cai**[2], **Lingjing Kong**[2], **Minlie Huang**[3] and **Boi Faltings**[2]
[1]Huawei Noah's Ark Lab
[2]LIA, EPFL
[3]CoAI, DCST, Tsinghua University
mifei2@huawei.com, aihuang@tsinghua.edu.cn
{wanhao.zhou,fengyu.cai,lingjing.kong,boi.faltings}@epfl.ch

Abstract

As the labeling cost for different modules in task-oriented dialog (ToD) systems is expensive, a major challenge is to train different modules with the least amount of labeled data. Recently, large-scale pre-trained language models, have shown promising results for few-shot learning in ToD. In this paper, we devise a self-training approach to utilize the abundant unlabeled dialog data to further improve state-of-the-art pre-trained models in few-shot learning scenarios for ToD systems. Specifically, we propose a self-training approach that iteratively labels the most confident unlabeled data to train a stronger *Student* model. Moreover, a new text augmentation technique (GradAug) is proposed to better train the *Student* by replacing non-crucial tokens using a masked language model. We conduct extensive experiments and present analyses on four downstream tasks in ToD, including intent classification, dialog state tracking, dialog act prediction, and response selection. Empirical results demonstrate that the proposed self-training approach *consistently* improves state-of-the-art pre-trained models (BERT, ToD-BERT) when only a small number of labeled data are available.

1 Introduction

Large-scale pre-trained language models, such as BERT (Devlin et al., 2019), UniLM (Dong et al., 2019), GPT (Radford et al., 2018), GPT-2 (Radford et al., 2019), and GPT-3 (Brown et al., 2020), have shown great few-shot or zero-shot learning abilities in various NLP tasks with the help of task-agnostic language knowledge learned via pre-training tasks. In task-oriented dialog (ToD) systems, the labeling cost is very high such that the size of well-labeled data is often small. Therefore, few-shot learning in ToD is important and valuable in many practical applications. Many attempts (Peng et al., 2020b,a; Wu et al., 2020) have been proposed to leverage large-scale pre-trained language models to improve few-shot learning in ToD. Specifically, a model pre-trained on general text corpora is further trained on public ToD datasets.

Although the size of labeled data is often small, a practical ToD system de facto has many unlabeled dialog data. Therefore, utilizing unlabeled data to improve a ToD system is practically important. In this paper, we take a semi-supervised self-training (ST) perspective to iteratively train a better *Student* model using unlabeled data (Scudder, 1965; Yarowsky, 1995). ST has been successfully applied to a variety of tasks, including image classification (Yalniz et al., 2019; Xie et al., 2020; Zoph et al., 2020), automatic speech classification (Synnaeve et al., 2019; Kahn et al., 2020; Park et al., 2020; Likhomanenko et al., 2020), sequence generation (He et al., 2020), and natural language understanding (Du et al., 2020).

We are going to study this research question: *can self-training provide complementary benefits on top of the strong pre-training models for few-shot learning in ToD?* Recently, Xie et al. (2020); Zoph et al. (2020) studied a similar question in the context of image classification, showing that ST effectively refines pre-training models. Du et al. (2020) also recently showed the benefit of ST over pre-training for general natural language understanding. Yet, their main proposal is to crawl a large amount of similar unlabeled data from the web.

In this paper, we propose a self-training approach based on iterative pseudo-labeling (Lee, 2013). It first trains a *Teacher* on the labeled samples. The *Teacher* then iteratively generates pseudo-labels for the most confident subset of unlabeled samples to train a better *Student*. To train a more robust *Student* during self-training, we propose a data augmentation technique called GradAug. GradAug first "masks" a fraction of tokens of a dialog input. Then, it reconstructs the corrupted text with a pre-trained masked language model of BERT. Different from Ng et al. (2020), the probability of masking

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 1887–1898
November 7–11, 2021. ©2021 Association for Computational Linguistics

a token is conditioned on the gradient of the corresponding token embedding w.r.t. the downstream task. In this way, GradAug prevents replacing tokens that are critical for a downstream task.

The main contribution of this paper is three-fold:

- This is the first attempt to study the effect of self-training on top of existing strong pre-trained models for ToD in few-shot learning scenarios.

- We propose a self-training method to gradually train a stronger *Student* by iteratively labeling the most confident unlabeled data and a new text augmentation technique (GradAug).

- We conduct extensive experiments on four downstream tasks in ToD, including intent classification, dialog state tracking, dialog act prediction, and response selection. Empirical results demonstrate that self-training *consistently* improves state-of-the-art pre-trained models (BERT, ToD-BERT Wu et al. (2020)).

2 Related Work

2.1 Pre-training for ToD Systems

Budzianowski and Vulic (2019) first applied GPT-2 to train a response generation model by taking the system belief state, database entries, and last dialog turn as input. Henderson et al. (2019) pre-trained a response selection model for ToD by first pre-training on general-domain conversational corpora (Reddit). Ham et al. (2020); Hosseini-Asl et al. (2020); Peng et al. (2020a) proposed to train GPT-2 on different sub-tasks (dialog state tracking, dialog act prediction, and response generation) as a sequence prediction problem.

Recent studies have shown that large-scale pre-trained language models are good few-shot learners (Brown et al., 2020). Several studies have also confirmed these findings for ToD. For the task of generating responses conditioned on a semantic representation, GPT-2 was leveraged by Peng et al. (2020b) to improve few-shot learning. Peng et al. (2020a) utilized GPT-2 for few-shot end-to-end response generation from dialog contexts. T5 (Raffel et al., 2020) is also recently applied to few-shot ToD by Lin et al. (2021) for dialog state tracking and by Kale and Rastogi (2020) for natural language generation. Wu et al. (2020) improve a BERT model for few-shot learning on four downstream tasks.

2.2 Self-training

The first focus of self-training is designing better policies to label unlabeled samples. Zhang and Zhou (2011) evaluated the confidence via a statistic-based data editing technique. Lee (2013) designed an annealing function that gradually increases the loss of labeled samples during training. Amiri (2019) utilized a Leitner queue (Dempster, 1989) to gradually put confident samples in the front. Niu et al. (2020) selected the most confident samples with prediction loss below some threshold. Kumar et al. (2010); Ma et al. (2017); Li et al. (2019); Mukherjee and Awadallah (2020) proposed to learn sampling weights for unlabeled data to control the selection process. Reinforcement learning (RL) methods (Chen et al., 2018; Wu et al., 2018; Ye et al., 2020) designed an additional Q-agent as the sample selector. Nevertheless, methods using learnable weights or RL provide marginal benefits compared to the elevated optimization cost. As designing new sample selection schemes is not our primary focus, we will go for a simple and effective pipeline described in Section 4.1. Specialized explorations on this topic are orthogonal to the focus of this paper and will be left as future work.

The second focus of self-training is to improve the robustness of the *Student* model trained from potentially noisy pseudo-labeled samples. Data augmentation techniques are widely used. In computer vision, recent works demonstrated the benefit of different stochastic augmentation tricks, including input transformations (Laine and Aila, 2017; Xie et al., 2020; Zoph et al., 2020), dropout (Laine and Aila, 2017; Xie et al., 2020; Zoph et al., 2020), adversarial samples (Miyato et al., 2019), and Mixup (Berthelot et al., 2019, 2020). Text augmentation is more challenging because of the complex syntactic and semantic structures. Miyato et al. (2017) utilized adversarial training to apply perturbations to word embeddings. Wei and Zou (2019) proposed EDA using basic synonym replacement, random insertion, swap, and deletion. Kumar et al. (2019) proposed to maximize a monotone sub-modular function to obtain diverse paraphrases. Xie et al. (2019) proposed UDA applying back-translation (Edunov et al., 2018) and word replacement using a Tf-Idf metric. He et al. (2020) studied the effect of dropout compared to back-translation during self-training for the neural sequence generation task. Chen et al. (2020) proposed MixText that utilizes Manifold Mixup (Verma et al., 2019) to interpolate hidden layers corresponding to semantic representations of BERT. Ng et al. (2020) proposed SSMBA utilizing the masked language model of BERT to replace words. In experiments, we compare the proposed GradAug technique with state-of-the-art text augmentation methods.

3 Background of Using Pre-trained Models for Downstream Tasks in ToD

In this section, we first briefly overview the pipeline of utilizing large-scale pre-trained models for four common downstream tasks (intent classification, dialog state tracking, dialog act prediction, and response selection) in ToD. We denote the input and label of different downstream tasks as x and y, and a prediction model is denoted as $\hat{y}_x = F(x)$. F can often be decomposed into two parts. The first part is a *feature extractor* $\boldsymbol{h} = A(x) \in \mathbb{R}^l$ which computes a hidden representation $\boldsymbol{h}$ of x, and the second part is an *output network* for prediction. Large-scale pre-trained language models serve as *feature extractor* A to compute a hidden representation for an input. For example, we use the [CLS] embedding of BERT as the hidden representation $\boldsymbol{h}$ when BERT is adopted as A. Different *output networks* are designed for different downstream tasks, and the details following ToD-BERT (Wu et al., 2020) are described below.

Intent classification. This is a multi-class classification problem to predict the single intent label y of an input utterance x. The model computes the probability over I possible intents as:

$$\boldsymbol{p}_{int} = Softmax(W_1 \cdot A(x)) \in \mathbb{R}^I, \qquad (1)$$

where $W_1 \in \mathbb{R}^{I \times l}$ is a trainable weight matrix, and the model is optimized by the standard cross-entropy loss compared to the ground truth.

Dialog state tracking. It is a multi-class classification problem based on a predefined ontology. Unlike intent classification, the dialog history (a sequence of utterances) is used as the input x. For each (domain, slot) pair, the model predicts a score over all potential slot values. For the i-th slot value v_i^j of the j-th pair, the cosine similarity score compared to the input x is computed as follows:

$$s_i^j = Cosine(G_j(A(x)), A(v_i^j)) \in \mathbb{R}^1, \qquad (2)$$

where G_j is the slot projection layer of the j-th pair, and the number of layers $|G|$ equals the number of (domain, slot) pairs. The model is trained with the cross-entropy loss summed over all the pairs.

Dialog act prediction. This is a multi-label classification problem to predict dialog act (DA) intents for the next system response. The model takes a dialog history as input x and predicts a Bernoulli outcome for each possible DA intent as:

$$\boldsymbol{a} = Sigmoid(W_2 \cdot A(x)) \in \mathbb{R}^N, \qquad (3)$$

where $W_2 \in \mathbb{R}^{N \times l}$ is a trainable weight matrix, and N is the number of possible DA intents. Values in $\boldsymbol{a}$ are between $[0, 1]$, and the model is optimized by a binary cross-entropy loss w.r.t. the ground truth. A threshold of 0.5 is applied during inference.

Response selection. This task predicts the most relevant system response from a candidate pool. A dual-encoder model (Henderson et al., 2019) is adopted to compute the similarity between the input dialog history x and the i-th candidate response c_i:

$$r_i = Cosine(A(x), A(c_i)) \in \mathbb{R}^1. \qquad (4)$$

During training, we randomly sample 20 negative responses for each ground truth response. A cross-entropy loss is applied aiming to rank the ground truth highest.

4 Self-training

In this section, we introduce our self-training (ST) algorithm. The overall ST algorithm is introduced in Section 4.1, and a new text augmentation method (GradAug) for ST to train a more robust *Student* is elaborated in Section 4.2.

4.1 Overall ST Algorithm

During training, two data pools are maintained and denoted as U (unlabeled data) and L (labeled data). Two versions of the model are maintained, *Teacher* (F^T) and *Student* (F^S). Before the iterations of ST start, the *Teacher* is first trained on the initial small number of labeled data L to “warm up”.

Pseudo-Labeling. In the beginning of an ST iteration, the *Teacher* first makes predictions on U. For every data input $x \in U$, the *Teacher* predicts the label of x as $\hat{y}_x = F^T(x)$. We set the predicted score of the prediction $\hat{y}_x$ as the *confidence score* s_x for this prediction. When there is only a single label in the prediction $\hat{y}_x$ (c.f. intent classification, response selection), s_x is the prediction score corresponding to the predicted label. When there are multiple labels in the prediction $\hat{y}_x$ (c.f. dialog state tracking, dialog act prediction), s_x takes the

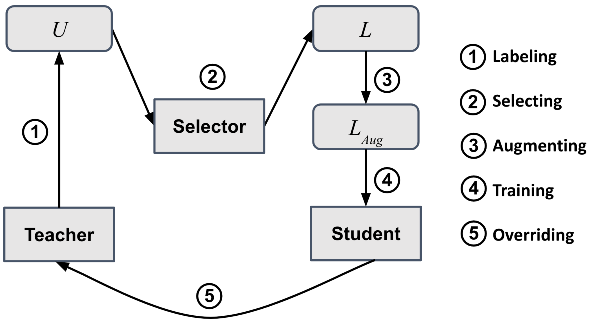

Figure 1: Pipeline of one ST iteration. The *Teacher* first generates predictions for data in U. Then, the *Selector* chooses the most confident samples based on the *Teacher*'s predictions and assign pseudo labels to them before appending to L. Afterwards, L is augmented by "GradAug" to train a *Student*. Lastly, the trained *Student* becomes the *Teacher* in the next iteration. Multiple iterations are computed till the *Student* converges.

mean of the prediction scores corresponding to the predicted labels. In each iteration, the *Selector* chooses top-k instances from U with the highest confidence scores, and assigns the corresponding predictions $\hat{y}_x$ as labels to them. These labeled instances will be moved from U to L.

Iterative *Student* training. The updated L is used to train a stronger *Student* model. We applied dropout (Srivastava et al., 2014) and a new text augmentation technique (GradAug) introduced later in Section 4.2 which augments L to L_{Aug}. At the end of each iteration, the *Teacher* model is overridden by the current *Student* to be used in the next iteration. We reinitialize the *Student* in every iteration to avoid over-fitting the initial and earlier data in L in multiple training iterations. As noted by Xie et al. (2020); Du et al. (2020), the *Student* should have an equal or larger capacity than the *Teacher* to gradually learn from L with increasing size. In this paper, we set the *Student* the same size as the *Teacher*, and we demonstrate in experiments that consistent improvements can be achieved without increasing model capacity.

Details of our ST algorithm are described in Algorithm 1, and the pipeline of one ST iteration (i.e., the "While" loop in Algorithm 1) is visualized in Figure 1.

4.2 Text Augmentation (GradAug)

Next, we propose a novel text augmentation technique called "GradAug" for data in L to train a more robust *Student*. Our method employs the masked language model (MLM, Devlin et al. (2019); Liu et al. (2019)), which is a common pretraining strategy for BERT-like architectures. In MLM, some tokens are replaced by the special token [MASK], and the model is asked to reconstruct the original tokens from the context.

Algorithm 1 Self-training (ST) for ToD

```
Input: Labeled data: $L$, Unlabeled data: $U$,
    Teacher: $F^T$, Student: $F^S$, Number of pseudo-
    labeled data in an iteration: $k$, Number of aug-
    mentations per input: $q$
Output: A trained Student $F^S$
Initialize $F^T$ and train $F^T$ on $L$
while $F^S$ not good enough $\&$ $U \neq Ø$ do
    Initialize $F^S$, $L' \leftarrow Priority\_list()$
    for $x \in U$ do
        Compute prediction label $\hat{y}_x = F^T(x)$
        Compute confidence score $s_x$
        $L'.insert(\{x, \hat{y}_x, s_x\})$
    end for
    $L' \leftarrow L'.top(k)$
    $L \leftarrow L \cup L', U \leftarrow U \backslash L'$
    $L_{Aug} \leftarrow GradAug(L, F^T, q)$
    Train $F^S$ on $L_{Aug}$ with dropout
    $F^T \leftarrow F^S$
end while
```

To utilize a pre-trained MLM (e.g. BERT) for text augmentation, the first step is to decide *which tokens to mask*. Random sampling is used by the original BERT framework and a recent text augmentation method (SSMBA, Ng et al. (2020)). However, if some crucial tokens are masked, the semantics might change after the reconstruction. For example, if the important token "status" in Figure 2 is masked, top predictions from the MLM of BERT includes "purpose", "cost", and "route", which will potentially change the original semantics.

Gradient-based token masking. Instead of randomly masking tokens, we compute a *masking probability* $\boldsymbol{p} = [p_1, ..., p_n]$ for an input x of n tokens. For input x with token embedding matrix[1] $X = [X_1, ..., X_n]^\intercal \in \mathbb{R}^{n \times d}$ and label y, the *importance* of tokens in x to the label y is computed by a *saliency map* (Simonyan et al., 2014) $\boldsymbol{m}$:

$$\begin{cases} \boldsymbol{m} = \left[M(X_1), \ldots, M(X_n)\right]^\intercal \in \mathbb{R}^n, \\ M(X_i) = \mathbb{1}^\intercal \left(\dfrac{\partial F_y^T(X)}{\partial X_i} \right) \in \mathbb{R}^1, \end{cases} \tag{5}$$

[1] We use the token embeddings of BERT-like architectures, rather than position or segmentation embeddings.

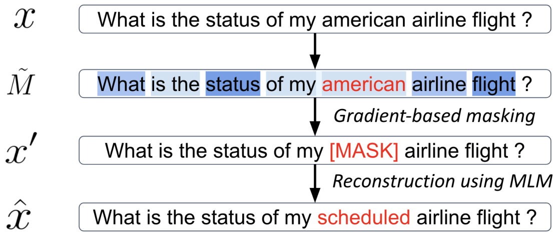
x
What is the status of my american airline flight ?
$\tilde{M}$
What is the status of my american airline flight ?
Gradient-based masking
x'
What is the status of my [MASK] airline flight ?
Reconstruction using MLM
$\hat{x}$
What is the status of my scheduled airline flight ?

Figure 2: An illustrative example of GradAug. First, the smooth saliency $\tilde{M}$ is computed for each token, and we highlight important tokens in blue for the intent label "flight_status". Less important tokens are more likely to be masked. Then, the masked token ("american") is reconstructed by the MLM of BERT and the replacement token "scheduled" does not change the semantics of the original sentence.

where $F_y^T(X)$ is the *Teacher* model's prediction score for the label y. $M(X_i)$ measures the *importance* of the i-th token by accumulating the gradients of all elements in its embedding $X_i \in \mathbb{R}^d$ by differentiating $F_y^T(X)$ w.r.t. X_i. The intuition is that tokens with large gradients are important to the label y. However, previous studies (Sundararajan et al., 2017; Smilkov et al., 2017) pointed out that raw gradients can be very noisy and may sharply fluctuate locally. To this end, we compute a *smooth saliency* measure (Smilkov et al., 2017) $\tilde{M}(X_i)$ for the i-th token as:

$$\begin{cases} \tilde{M}(X_i) = \dfrac{1}{m}\displaystyle\sum_{j=1}^{m} M(\tilde{X}_i^j) \in \mathbb{R}^1, \\ \tilde{X}_i^j = X_i + \boldsymbol{z}^j, \end{cases} \tag{6}$$

where m Gaussian noises $\boldsymbol{z}^j \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}) \in \mathbb{R}^d$ with mean $\mathbf{0}$ and diagonal co-variance matrix $\boldsymbol{\Sigma}$ are added to X_i to calculate m regular saliency measures, which average to the *smooth saliency* $\tilde{M}(X_i)$ for X_i. The probability p_i of masking the i-th token is inversely correlated to $\tilde{M}(X_i)$ as:

$$p_i \propto \frac{1}{\tilde{M}(X_i)^\beta}, \tag{7}$$

where β controls the flatness of the distribution $\boldsymbol{p}$, and $\boldsymbol{p}$ is normalized by its sum. As the probability p_i to mask a token x_i is inversely correlated to its importance $\tilde{M}(X_i)$ to a downstream task, more important tokens are less likely to be masked. We sample 15% [2] tokens of x based on $\boldsymbol{p}$ and replace them by [MASK] to corrupt x to x'. As F^T is updated in each ST iteration, $\boldsymbol{p}$ is dynamically calculated in each ST iteration.

Algorithm 2 GradAug

```
Input: Labeled data: $L$, Teacher: $F^T$, Number of
    augmentations per input: $q$
Output: Augmented labeled data $L_{Aug}$
Initialize $L_{Aug} \leftarrow L$
for $\{x, y\} \in L$ do
    Compute masking probability $\boldsymbol{p}$ using $F^T$
    for $j \in 1 \dots q$ do
        $x' \leftarrow$ Mask tokens of $x$ based on $\boldsymbol{p}$
        $\hat{x} \leftarrow$ Predict masked tokens by MLM
        $L_{Aug}.append(\{\hat{x}, y\})$
    end for
end for
```

Reconstruction using MLM. To reconstruct the masked tokens in x', we utilize a pre-trained MLM to predict the [MASK] tokens. For stochastic purposes suggested by Fan et al. (2018), we reconstruct each [MASK] by sampling 1 token from 10 most likely tokens according to their predicted probabilities. Afterwards, we get a paraphrased $\hat{x}$ of the original x as an augmentation. As our gradient-based masking scheme avoids replacing tokens crucial to the meaning of x, the label of $\hat{x}$ is preserved the same as x.

An illustrative example of GradAug is given in Figure 2, and the detailed procedure applying GradAug on L is described in Algorithm 2.

5 Experiments

5.1 Dataset Description

We evaluate four different datasets for four downstream tasks as in Wu et al. (2020).

OOS (Larson et al., 2019) is a benchmark dataset for intent classification in ToD. It consists of 150 in-domain intents and 1 out-of-scope intent. The full dataset contains 15,100/3,100/5,500 samples for train/validation/test, and all data are balanced across 151 different intents.

MWOZ (Eric et al., 2020) is evaluated in three downstream tasks, including dialog state tracking, dialogue act prediction, and response prediction. It contains 8,420/1,000/1,000 dialogues for train/validation/test. For dialog act prediction, we remove the domain information from original labels as in Wu et al. (2020), resulting 13 DA intents.

DSTC2 (Henderson et al., 2014) and **GSIM** (Shah et al., 2018) are two corpus used in dialog act prediction and response selection tasks. DSTC2 contains 1,612/506/1,117 dia-

[2] This is the default ratio used by BERT and SSMBA.

Data	Model	Acc. (all)	Acc. (in)	Acc. (out)	Recall (out)
1%	BERT	36.5% ± 2.7%	44.6% ± 3.3%	81.8% ± 0.8%	0.4% ± 0.2%
	BERT-ST	**70.1%** ± 2.2%	**82.2%** ± 3.7%	**84.3%** ± 1.0%	**15.5%** ± 1.3%
	ToD-BERT	39.0% ± 1.3%	47.1% ± 0.7%	82.0% ± 0.4%	2.3% ± 0.3%
	ToD-BERT-ST	**75.8%** ± 1.7%	**87.8%** ± 1.5%	**85.5%** ± 0.7%	**21.9%** ± 0.9%
10%	BERT	73.6% ± 1.9%	87.4% ± 2.1%	83.9% ± 0.5%	11.7% ± 1.1%
	BERT-ST	**80.6%** ± 1.7%	**94.3%** ± 1.5%	**84.9%** ± 0.6%	**17.1%** ± 0.9%
	ToD-BERT	75.5% ± 1.0%	89.4% ± 0.8%	84.1% ± 0.7%	13.3% ± 1.4%
	ToD-BERT-ST	**85.3%** ± 0.9%	**94.7%** ± 0.7%	**89.4%** ± 0.6%	**42.8%** ± 1.7%
Full*	BERT	84.9%	95.8%	88.1%	35.6%
	ToD-BERT	86.6%	96.2%	89.9%	43.6%

Table 1: Results of intent classification. Bold numbers indicate ST improves the corresponding pre-trained model. Results with * are taken from Wu et al. (2020).

logues for train/validation/test; GSIM contains 1,500/469/1,039 dialogues for train/validation/test. DA intent labels of DSTC2 and GSIM are mapped to universal dialogue acts (Paul et al., 2019), resulting in 19 and 13 DA intents respectively.

5.2 Experiment Settings

We randomly sample 1% or 10% of the training data to serve as the initial labeled data L, while the remainders are used as unlabeled data U. We report mean and standard deviation with three different random seeds for each experiment to reduce data sampling variance. We also report the upper bound of pre-trained models *without* ST using all labeled training data, referred to as "Full".

We test two pre-trained models: (i). uncased base BERT with 110M parameters; (ii). ToD-BERT [3] (Wu et al., 2020) that is further pre-trained on 9 public ToD datasets on top of BERT. When ST is applied to them, the corresponding MLM is used by GradAug to reconstruct masked tokens. Basic model parameters of the first 3 downstream tasks are set the same as Wu et al. (2020). In response selection, we reduced training batch size from 25 to 20 to fit our computation constraint.

BERT and Tod-BERT without ST are trained on the initial labeled data L until validation performance does not improve for 20 epochs [4]. For ST, when the *Student* is trained on L_{Aug} in one ST iteration (c.f. Algorithm 1 line 12), we apply early stop until validation performance does not improve for 10 epochs. Moreover, the best *Student* across multiple ST iterations is selected based on validation performance (c.f. Algorithm 1 line 2). It means that the best *Student* model does *not* necessarily use up all unlabeled data. Other hyper-parameters of ST selected base on validation performance are reported in Appendix A.3.

Data	Model	Joint Acc	Slot Acc
1%	BERT	8.0% ± 1.1%	84.3% ± 0.6%
	BERT-ST	**8.8%** ± 0.6%	**84.5%** ± 0.4%
	ToD-BERT	8.4% ± 0.5%	85.7% ± 0.4%
	ToD-BERT-ST	**9.9%** ± 0.3%	**86.5%** ± 0.2%
10%	BERT	21.2% ± 0.5%	92.0% ± 0.3%
	BERT-ST	**23.9%** ± 0.3%	**92.4%** ± 0.5%
	ToD-BERT	25.5% ± 0.6%	93.4% ± 0.2%
	ToD-BERT-ST	**28.3%** ± 0.4%	**93.7%** ± 0.1%
Full*	BERT	45.6%	96.6%
	ToD-BERT	48.0%	96.9%

Table 2: Results of dialog state tracking. Bold numbers indicate ST improves the corresponding pre-trained model. Results with * are taken from Wu et al. (2020).

5.3 Main Results of Four Downstream Tasks

Intent classification. Results of intent classification on OOS are presented in Table 1 with *accuracy* of **all** 151 intents; 150 **in**-domain intents; the **out**-of-scope intent, and the *recall* of the out-of-scope intent. ST significantly improves the pre-trained BERT and ToD-BERT. When only 1% labeled data are used, ST achieves 33.6% and 36.8% higher accuracy on all 151 intents for BERT and ToD-BERT respectively. For 10% labeled data, the above two margins are 7.0% and 9.8%. Furthermore, ST largely improves the recall of the out-of-scope intent, indicating that it is more robust to out-of-scope intents with noisy distributions.

Dialog state tracking. Results of dialog state tracking on MWOZ are presented in Table 2. Two common evaluation metrics (Budzianowski et al., 2018; Wu et al., 2019) are used: *slot accuracy* and *joint goal accuracy*. Slot accuracy is computed for each individual state (domain, slot, value) to check whether the value is correctly predicted. Joint goal accuracy checks whether the predicted states exactly matches the ground truth states. We could

[3] We used their joint version (ToD-BERT-jnt) pre-trained with the MLM and "response contrastive loss" objectives.

[4] Our different (often better) results compared to the ToD-BERT paper mainly come from this stricter early stop criteria.

Data	Model	MWOZ		DSTC2		GSIM	
		micro-F1	macro-F1	micro-F1	macro-F1	micro-F1	macro-F1
1%	BERT	83.5% ± 0.7%	61.2% ± 1.5%	79.1% ± 1.4%	26.8% ± 0.4%	70.3% ± 1.3%	27.9% ± 0.7%
	BERT-ST	82.7% ± 0.5%	**64.2%** ± 0.8%	**81.4%** ± 0.7%	**27.3%** ± 0.2%	**73.0%** ± 0.8%	**29.8%** ± 0.4%
	ToD-BERT	85.8% ± 0.2%	67.0% ± 0.6%	80.9% ± 0.7%	25.3% ± 0.4%	86.5% ± 3.0%	36.6% ± 2.1%
	ToD-BERT-ST	**86.9%** ± 0.4%	**71.8%** ± 0.3%	**82.7%** ± 0.8%	**28.5%** ± 0.4%	**92.6%** ± 1.1%	**40.8%** ± 0.9%
10%	BERT	89.8% ± 0.2%	77.8% ± 0.3%	88.9% ± 0.7%	35.7% ± 1.3%	97.1% ± 0.3%	44.1% ± 0.2%
	BERT-ST	89.5% ± 0.1%	**79.2%** ± 0.6%	**92.3%** ± 0.6%	**38.4%** ± 1.0%	**97.6%** ± 0.2%	**44.6%** ± 0.4%
	ToD-BERT	90.0% ± 0.2%	78.4% ± 1.0%	90.6% ± 2.1%	38.8% ± 1.9%	98.6% ± 0.2%	44.9% ± 0.2%
	ToD-BERT-ST	**90.2%** ± 0.2%	**79.6%** ± 0.4%	**92.9%** ± 0.8%	**40.5%** ± 0.9%	**99.3%** ± 0.3%	**45.6%** ± 0.4%
Full*	BERT	91.4%	79.7%	92.3%	40.1%	98.7%	45.2%
	ToD-BERT	91.7%	80.6%	93.8%	41.3%	99.5%	45.8%

Table 3: Results of dialog act prediction. Bold numbers indicate ST improves the corresponding pre-trained model. Results with * are taken from Wu et al. (2020).

Data	Model	MWOZ		DSTC2		GSIM	
		Recall@1	Recall@3	Recall@1	Recall@3	Recall@1	Recall@3
1%	BERT	7.3% ± 1.4%	19.5% ± 3.2%	3.5% ± 0.6%	9.8% ± 1.5%	4.0% ± 0.6%	11.4% ± 1.0%
	BERT-ST	**23.8%** ± 1.4%	**46.1%** ± 0.7%	**36.7%** ± 0.4%	**51.1%** ± 1.3%	**11.1%** ± 0.8%	**24.2%** ± 0.6%
	ToD-BERT	37.5% ± 1.9%	63.0% ± 1.1%	35.7% ± 0.9%	53.8% ± 0.7%	11.4% ± 1.1%	24.1% ± 0.9%
	ToD-BERT-ST	**43.5%** ± 0.7%	**66.3%** ± 0.6%	**48.0%** ± 0.5%	**64.6%** ± 0.3%	**27.8%** ± 1.0%	**42.9%** ± 0.8%
10%	BERT	26.1% ± 3.0%	56.5% ± 3.5%	27.7% ± 2.1%	42.9% ± 3.4%	13.4% ± 0.6%	28.3% ± 1.7%
	BERT-ST	**43.1%** ± 1.1%	**66.1%** ± 1.3%	**53.7%**± 2.0%	**67.1%** ± 2.9%	**22.3%** ± 0.4%	**40.4%** ± 0.9%
	ToD-BERT	47.2% ± 1.1%	69.4% ± 1.1%	51.3% ± 0.7%	66.0% ± 0.49%	28.5% ± 0.7%	47.8% ± 1.0%
	ToD-BERT-ST	**60.2%** ± 1.3%	**81.9%** ± 1.6%	**58.8%** ± 0.8%	**72.2%** ± 1.1%	**41.8%** ± 0.9%	**64.9%** ± 1.4%
Full	BERT	47.5%	75.5%	46.6%	62.1%	13.4%	32.9%
	ToD-BERT	66.9%	89.1%	59.5%	73.1%	43.0%	65.3%

Table 4: Results of response selection. Bold numbers indicate ST improves the corresponding pre-trained model.

see that ST consistently improves both BERT and ToD-BERT. E.g., ST has 1.5% and 2.8% joint goal accuracy improvement over ToD-BERT when 1% and 10% labeled data are used respectively. Similar margins can be observed for ST on top of BERT.

Dialog act prediction. Experiments are conducted on three datasets and results are reported in Table 3. We report *micro-F1* and *macro-F1* scores for this multi-label classification task. Again, the benefit of ST can be observed by the improvement for both BERT and ToD-BERT. When 10% labeled data are used, BERT and ToD-BERT perform similarly to their upper bound (Full), and the improvement margin of ST is limited. When 1% labeled data are used, more notable margins of ST can be seen on the two simpler datasets (DSTC2, GSIM) and the macro-F1 score of MWOZ.

Response selection. Results of response selection on three datasets are reported in Table 4. We randomly sample 100 responses as negative responses and report Recall@1&3 (Henderson et al., 2019) indicating whether the true response is ranked in the top-1 or top-3 predicted responses. When 1% labeled data are used, ST achieves 6%, 12.3%, and 16.4% higher Recall@1 accuracy over ToD-BERT on three datasets respectively. For 10% labeled data, the three margins above are 13.0%, 7.5%, and 14.4% respectively. Larger improvements can be observed for ST on top of BERT.

Altogether, our experiments on four different downstream tasks reveal that:

- Self-training provides complementary benefits on top of pre-training. ST consistently improves both BERT and ToD-BERT on all four downstream tasks with only 1% and 10% labeled data.
- Self-training is on par with customized pre-training for ToD. BERT performs worse than ToD-BERT, yet BERT-ST achieves comparable or even better performance than ToD-BERT which is heavily pre-trained on ToD corpora.
- Self-training bridges the gap between few-shot learning and full supervision. BERT and ToD-BERT with 10% labeled data perform much worse than models using all labeled data ("Full") for intent classification and response selection. ST largely improves performances in these two cases with results comparable to "Full".
- The benefit of self-training is evident on two simpler single-label prediction tasks (intent classification, response selection), indicated by 6-37% gain with 1% labeled data; 7-15% gain with 10% labeled data. The margin is smaller on two other more challenging multi-label prediction tasks.

	IC Acc. (all)	RS Recall@3
ToD-BERT-ST	85.3%	64.9%
w/o Smooth Saliency	81.9%	64.4%
w/o Augmentation	80.4%	54.8%
w/o Pseudo-Labeling	76.9%	49.7%
ToD-BERT	75.5%	47.8%

Table 5: Ablation study of ST for intent classification (IC) on OOS and response selection (RS) on GSIM.

5.4 In-depth Analyses of Self-training

In this section, we provide in-depth analyses of the proposed self-training approach. As case studies, we limited our discussion on intent classification (**IC**) on OOS and response selection (**RS**) on GSIM using ToD-BERT-ST with 10% labeled data. Reported results are accuracies on all intents and Recall@3 respectively.

Ablation study. In Table 5, we compare three simplified versions of ToD-BERT-ST to understand the effects of different components. We can observe that: (i) Masking tokens using the smooth saliency computed in Eq. (6) for GradAug is beneficial because replacing it by the vanilla saliency in Eq. (5) ("w/o Smooth Saliency") degrades the performance by 3.4% and 0.5% on IC and RS. (ii) Training a more robust *Student* using data augmented by GradAug is advantageous because dropping this augmentation step ("w/o Augmentation") impairs performance by 4.9% and 10.1%. (iii) The Pseudo-Labeling operation to iteratively label unlabeled data is important for ST, indicated by the 8.4% and 15.2% performance drop of "w/o Pseudo-Labeling" that only applies GradAug to the initial labeled data without utilizing unlabeled data.

Comparison to other *Selectors* in ST. In Table 6, we compare our scheme of selecting samples with top-k confident predictions from U in each iteration with (i) Random-k: randomly select k samples; (ii) Least-k: select samples with least-k confident predictions (iii) Select-all (Xie et al., 2020; Du et al., 2020): label all samples of U in an iteration and relabel them in the next iteration. We could see that "Random-k" and "Least-k" perform worse than ours, yet they both outperform "Select-all" by large margins. It means that the initial *Teacher* trained on limited labeled data is not good enough to assign reliable labels to a large number of unlabeled data.

	IC Acc. (all)	RS Recall@3
Top-k (Ours)	85.3%	64.9%
Random-k	84.0%	64.1%
Least-k	82.7%	61.4%
Select-all	76.0%	50.8%

Table 6: Comparison to other *Selectors* in ST for intent classification (IC) on OOS and response selection (RS) on GSIM.

	IC Acc. (all)	RS Recall@3
GradAug (Ours)	85.3%	64.9%
SSMBA (Ng et al., 2020)	84.6%	64.2%
MixText (Chen et al., 2020)	83.6%	62.7%
UDA (Xie et al., 2019)	82.5%	62.2%
EDA (Wei and Zou, 2019)	77.2%	57.6%
w/o Augmentation	80.4%	54.8%

Table 7: Comparison to other text augmentation methods to train the *Student* for intent classification (IC) on OOS and response selection (RS) on GSIM.

Comparison to other text augmentation methods. In Table 7, we compare GradAug with four representative text augmentation methods to augment L. We follow the default setting of these techniques and apply them to our ST pipeline to generate three paraphrases for each input as in GradAug. We could see that GradAug consistently outperforms the current state-of-the-art (SSMBA, MixText, UDA), and it outperforms EDA by large margins. As EDA might easily change the input semantics, it even performs worse than using no data augmentation for intent classification. This result reinforces the importance of preserving semantics during augmentation for ToD.

6 Conclusion

We study using self-training to improve the strong pre-trained models for few-shot learning tasks in ToD. An iterative self-training method with a new text augmentation technique (GradAug) is proposed to gradually train a stronger *Student* model using unlabeled data. Extensive empirical results on four downstream tasks in ToD demonstrate the consistent improvements of self-training on top of pre-trained models. Our findings on using self-training to improve learning from limited labeled data may inspire future studies towards building more sample-efficient and scalable ToD systems.

References

Hadi Amiri. 2019. Neural self-training through spaced repetition. In *NAACL-HLT (1)*, pages 21–31. Association for Computational Linguistics.

David Berthelot, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. 2020. Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *ICLR*.

David Berthelot, Nicholas Carlini, Ian J. Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. 2019. Mixmatch: A holistic approach to semi-supervised learning. In *NeurIPS*, pages 5050–5060.

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. *CoRR*, abs/2005.14165.

Pawel Budzianowski and Ivan Vulic. 2019. Hello, it's GPT-2 - how can I help you? towards the use of pretrained language models for task-oriented dialogue systems. In *NGT@EMNLP-IJCNLP*, pages 15–22. Association for Computational Linguistics.

Pawel Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz - A large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *EMNLP*, pages 5016–5026. Association for Computational Linguistics.

Chenhua Chen, Yue Zhang, and Yuze Gao. 2018. Learning how to self-learn: Enhancing self-training using neural reinforcement learning. In *IALP*, pages 25–30. IEEE.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *ACL*, pages 2147–2157. Association for Computational Linguistics.

Frank N Dempster. 1989. Spacing effects and their implications for theory and practice. *Educational Psychology Review*, 1(4):309–330.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *NeurIPS*, pages 13042–13054.

Jingfei Du, Edouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Ves Stoyanov, and Alexis Conneau. 2020. Self-training improves pre-training for natural language understanding. *CoRR*, abs/2010.02194.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *EMNLP*, pages 489–500. Association for Computational Linguistics.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Kumar Goyal, Peter Ku, and Dilek Hakkani-Tür. 2020. Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *LREC*, pages 422–428. European Language Resources Association.

Angela Fan, Mike Lewis, and Yann N. Dauphin. 2018. Hierarchical neural story generation. In *ACL (1)*, pages 889–898. Association for Computational Linguistics.

DongHoon Ham, Jeong-Gwan Lee, Youngsoo Jang, and Kee-Eung Kim. 2020. End-to-end neural pipeline for goal-oriented dialogue systems using GPT-2. In *ACL*, pages 583–592. Association for Computational Linguistics.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2020. Revisiting self-training for neural sequence generation. In *ICLR*.

Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014. The second dialog state tracking challenge. In *SIGDIAL Conference*, pages 263–272. Association for Computational Linguistics.

Matthew Henderson, Ivan Vulic, Daniela Gerz, Iñigo Casanueva, Pawel Budzianowski, Sam Coope, Georgios Spithourakis, Tsung-Hsien Wen, Nikola Mrksic, and Pei-Hao Su. 2019. Training neural response selection for task-oriented dialogue systems. In *ACL (1)*, pages 5392–5404. Association for Computational Linguistics.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *CoRR*, abs/2005.00796.

Jacob Kahn, Ann Lee, and Awni Hannun. 2020. Self-training for end-to-end speech recognition. In *ICASSP*, pages 7084–7088. IEEE.

Mihir Kale and Abhinav Rastogi. 2020. Template guided text generation for task oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6505–6520, Online. Association for Computational Linguistics.

Ashutosh Kumar, Satwik Bhattamishra, Manik Bhandari, and Partha P. Talukdar. 2019. Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation. In *NAACL-HLT (1)*, pages 3609–3619. Association for Computational Linguistics.

M. Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *NIPS*, pages 1189–1197. Curran Associates, Inc.

Samuli Laine and Timo Aila. 2017. Temporal ensembling for semi-supervised learning. In *ICLR (Poster)*.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An evaluation dataset for intent classification and out-of-scope prediction. In *EMNLP/IJCNLP (1)*, pages 1311–1316. Association for Computational Linguistics.

Dong-Hyun Lee. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3.

Xinzhe Li, Qianru Sun, Yaoyao Liu, Qin Zhou, Shibao Zheng, Tat-Seng Chua, and Bernt Schiele. 2019. Learning to self-train for semi-supervised few-shot classification. In *NeurIPS*, pages 10276–10286.

Tatiana Likhomanenko, Qiantong Xu, Jacob Kahn, Gabriel Synnaeve, and Ronan Collobert. 2020. slimipl: Language-model-free iterative pseudo-labeling. *CoRR*, abs/2010.11524.

Zhaojiang Lin, Bing Liu, Seungwhan Moon, Paul A Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Andrea Madotto, Eunjoon Cho, and Rajen Subba. 2021. Leveraging slot descriptions for zero-shot cross-domain dialogue statetracking. In *NAACL-HLT (1)*, pages 5640–5648. Association for Computational Linguistics.

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. *CoRR*, abs/1907.11692.

Fan Ma, Deyu Meng, Qi Xie, Zina Li, and Xuanyi Dong. 2017. Self-paced co-training. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 2275–2284. PMLR.

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

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2019. Virtual adversarial training: A regularization method for supervised and semi-supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(8):1979–1993.

Subhabrata Mukherjee and Ahmed Hassan Awadallah. 2020. Uncertainty-aware self-training for text classification with few labels. *CoRR*, abs/2006.15315.

Nathan Ng, Kyunghyun Cho, and Marzyeh Ghassemi. 2020. SSMBA: Self-supervised manifold based data augmentation for improving out-of-domain robustness. In *EMNLP*, pages 1268–1283. Association for Computational Linguistics.

Yilin Niu, Fangkai Jiao, Mantong Zhou, Ting Yao, Jingfang Xu, and Minlie Huang. 2020. A self-training method for machine reading comprehension with soft evidence extraction. In *ACL*, pages 3916–3927. Association for Computational Linguistics.

Daniel S. Park, Yu Zhang, Ye Jia, Wei Han, Chung-Cheng Chiu, Bo Li, Yonghui Wu, and Quoc V. Le. 2020. Improved noisy student training for automatic speech recognition. In *INTERSPEECH*, pages 2817–2821. ISCA.

Shachi Paul, Rahul Goel, and Dilek Hakkani-Tür. 2019. Towards universal dialogue act tagging for task-oriented dialogues. In *INTERSPEECH*, pages 1453–1457. ISCA.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2020a. SOLOIST: few-shot task-oriented dialog with A single pre-trained auto-regressive model. *CoRR*, abs/2005.05298.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020b. Few-shot natural language generation for task-oriented dialog. In *EMNLP (Findings)*, pages 172–182. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

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

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*, 21:1–67.

Henry Scudder. 1965. Probability of error of some adaptive pattern-recognition machines. *IEEE Trans. Inf. Theory*, 11(3):363–371.

Pararth Shah, Dilek Hakkani-Tür, Bing Liu, and Gökhan Tür. 2018. Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and on-line reinforcement learning. In *NAACL-HLT (3)*, pages 41–51. Association for Computational Linguistics.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR (Workshop Poster)*.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. 2017. Smoothgrad: removing noise by adding noise. *CoRR*, abs/1706.03825.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR.

Gabriel Synnaeve, Qiantong Xu, Jacob Kahn, Edouard Grave, Tatiana Likhomanenko, Vineel Pratap, Anuroop Sriram, Vitaliy Liptchinsky, and Ronan Collobert. 2019. End-to-end ASR: from supervised to semi-supervised learning with modern architectures. *CoRR*, abs/1911.08460.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. 2019. Manifold mixup: Better representations by interpolating hidden states. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 6438–6447. PMLR.

Jason W. Wei and Kai Zou. 2019. EDA: easy data augmentation techniques for boosting performance on text classification tasks. In *EMNLP/IJCNLP (1)*, pages 6381–6387. Association for Computational Linguistics.

Chien-Sheng Wu, Steven Hoi, Richard Socher, and Caiming Xiong. 2020. TOD-BERT: Pre-trained natural language understanding for task-oriented dialogues. In *EMNLP*, pages 917–929. Association for Computational Linguistics.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. In *ACL (1)*, pages 808–819. Association for Computational Linguistics.

Jiawei Wu, Lei Li, and William Yang Wang. 2018. Reinforced co-training. In *NAACL-HLT*, pages 1252–1262. Association for Computational Linguistics.

Qizhe Xie, Zihang Dai, Eduard H. Hovy, Minh-Thang Luong, and Quoc V. Le. 2019. Unsupervised data augmentation. *CoRR*, abs/1904.12848.

Qizhe Xie, Minh-Thang Luong, Eduard H. Hovy, and Quoc V. Le. 2020. Self-training with noisy student improves imagenet classification. In *CVPR*, pages 10684–10695. IEEE.

I. Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. 2019. Billion-scale semi-supervised learning for image classification. *CoRR*, abs/1905.00546.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*, pages 189–196. Morgan Kaufmann Publishers / ACL.

Zhiquan Ye, Yuxia Geng, Jiaoyan Chen, Jingmin Chen, Xiaoxiao Xu, Suhang Zheng, Feng Wang, Jun Zhang, and Huajun Chen. 2020. Zero-shot text classification via reinforced self-training. In *ACL*, pages 3014–3024. Association for Computational Linguistics.

Min-Ling Zhang and Zhi-Hua Zhou. 2011. Cotrade: Confident co-training with data editing. *IEEE Trans. Syst. Man Cybern. Part B*, 41(6):1612–1626.

Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin Dogus Cubuk, and Quoc Le. 2020. Rethinking pre-training and self-training. In *NeurIPS*, volume 33.

Task	Dataset	Ratio of Labeled Data	Initial Samples in L	Hyper-parameter k
Intent classification	OOS	1% 10%	151 1510	500 1510
Dialog state tracking	MWOZ	1% 10%	567 5672	200 2000
Dialog act prediction	MWOZ	1% 10%	482 4824	500 2000
	DSTC2	1% 10%	116 1160	200 500
	GSIM	1% 10%	66 664	100 500
Response selection	MWOZ	1% 10%	482 4824	500 4500
	DSTC2	1% 10%	100 1006	100 700
	GSIM	1% 10%	66 664	80 500

Table 8: Dataset specifications for experiments in different downstream tasks and the hyper-parameter k (**the right-most column**) indicating the number of pseudo-labeled data in each self-training iteration in different experiments for four downstream tasks.

Appendix

A Reproducibility Checklist

A.1 Code

Our code will be available at `https://github.com/MiFei/ST-ToD` soon.

A.2 Dataset Specifications for Different Tasks

The exact dataset scales regarding the initial 1% or 10% labeled data in different few-shot learning scenarios for different downstream tasks are reported in Table 8 in the column headed with "**Initial Samples in** L".

A.3 Hyper-parameters of ST

	Source	Definition	Value
q	Alg. 2	# of augmentations per input	3
β	Eq. 7	flatness of distribution $\boldsymbol{p}$	1.0
m	Eq. 6	# of Gaussian noises	20
$\boldsymbol{\Sigma}$	Eq. 6	diagonal co-variance matrix	1e-4$\cdot\boldsymbol{I}$

Table 9: Hyper-parameters of ST that are shared across different downstream tasks and datasets.

The number (k) of pseudo-labeled data in each ST iteration in each experiment setting is reported in the rightmost column of Table 8. Other hyper-parameters of ST are reported in Table 9, and they are shared across different downstream tasks and datasets.

An exhaustive search on ST hyper-parameters is not conducted because it is very expensive to finely tune large pre-trained models on all four downstream tasks for different datasets. Therefore, we fix q, m and manually tune other hyper-parameters within reasonable ranges around current values indicated in Table 8 and Table 9. We could expect that even better results of ST can be achieved with a thorough hyper-parameter search by researchers or practitioners without computation constraints. To provide more insight, we found that setting k too small compromises computation time, while setting it too large compromises performance.

All experiments are conducted using a single GPU (GTX TITAN X), and eight GPUs are used in total.