Meta-learning a Language Modeling Objective

Jeffrey Shen
Department of Computer Science
Stanford University
jshen2@stanford.edu

Abstract

A large component of recent succcess in natural language processing (NLP) has simply been pretraining larger Transformer models with more data for longer, independent of any architectural change. We ask whether a better language modeling objective could more effectively use the unlabeled text during pretraining. In particular, we attempt to meta-learn the language modeling objective to better capture the relationship between the unlabeled data and downstream tasks.

In order to meta-learn a language modeling objective, we model the pretraining objective using a teacher network. The teacher learns to output a full input-output text example, which a student pretrains on. The goal of the teacher is to minimize the student's loss on the query set of labeled examples after it finetunes on the support set of labeled examples. This can be seen as extension to Meta Pseudo Labels [1], where the teacher only outputs a single classification label, and just minimizes the student's loss on labeled examples directly (e.g. no support set). We adapt their equations and methods to this setting and evaluate two approaches, using hard labels and using soft labels, against a standard masked language modeling (MLM) approach.

We perform two sets of experiments, one on a toy Two Moons dataset converted to text and one on an NLP dataset comprised of Wikipedia and SuperGLUE. For the Two Moons dataset, we find that across finetune datasets of varying sizes (10, 20, 40, 80, 160), MLM performs equivalently or better than both meta-learning approaches. The MLM objective yields a strong bias towards the unlabeled dataset, whereas the meta objectives possibly allow the teacher to overfit to teaching the small finetune dataset. Furthermore, we find that the hard labels teacher teaches mostly MLM-like tasks, whereas the soft labels teacher teaches a mixture between MLM-like and downstream tasks, possibly due to receiving more gradients as feedback from the student.

We also compare the soft labels and MLM approaches on the NLP dataset. Again, we find that MLM approach performed better on all SuperGLUE tasks compared to the soft labels approach (52.4 vs 46.6). Notably, the soft labels approach performed worse than outputing the most frquent class. We find that the teacher's outputs degenerate during pretraining, most likely because it is too easily influenced by the downstream tasks.

While unfortunately none of the meta-learning approaches performed better than MLM, we conclude with some possible extensions that can help alleviate the issue with the teacher overfitting to the downstream tasks, including constraining the teacher's output to always be a denoising objective. We hope that future work develops a better meta-objective for pretraining language models.

1 Introduction

Many recent approaches for natural language processing (NLP) have found tremendous success leveraging vast amounts of unlabeled text for pretraining a network, which can then be fine-tuned on downstream tasks. The performance of these approaches scales well with the model size, dataset size, and amount of compute [2], even across wide ranges of resource sizes. However, as a consequence, a

large component of the recent success has simply been pretraining larger models with more data for longer, independent of any architectural change [3, 4, 5].

On the other hand, changing the pretraining objective can substantially affect the performance [4], and also yield vastly more compute-efficient methods [6], but not necessarily across all types of downstream tasks. It is also important for the pretrain data and objective to be in-domain for the downstream tasks, even if the domain is smaller or more constrained [4]. We posit that a better language modeling objective could better capture the relationship between the unlabeled data and downstream tasks.

In this project, we attempt to meta-learn the language modeling objective. Similar to Meta Pseudo Labels [1], we model the pretraining objective with a teacher network in order to optimize it. In this work, we can extend their meta objective from computer vision to an NLP context by formulating the pretrain-finetune approach as a three step process (which involves one additional gradient step). As opposed to Meta Pseudo Labels, where the pretraining task and the finetuning task are the same classification task, Meta Language Modeling involves pretraining tasks with long input and output text, which makes meta-learning more challenging.

There are generally two ways to formulate the meta-optimization: with hard labels and with soft labels. Contrary to Meta Pseudo Labels, we find that in the new NLP setting, the behavior of each method is markedly different. Notably, soft labeling allows the teacher to derive more signal from the meta loss whereas hard labeling only allows one degree of freedom per update. Unfortunately, however, when we compare both of these methods against a standard masked language modeling (MLM) objective, we found that they were both outperformed by the MLM objective. This may be due to the downstream tasks affecting the pretrain objective, and causing the teacher to overfit to the downstream objectives, rather than finding a more general language modeling objective.

Overall, more study is needed to solve the issues with extending the meta-objective to an NLP setting, where the pretrain and finetune tasks are markedly different and where the pretraining objective can involve high-dimensional outputs. In doing so, we hope that we can find language modeling objectives that more effectively use the unlabeled data and that have better scalability with pretrain data and compute.

2 Related Work

Many approaches study different fixed pretraining objectives. For example, Radford et al. [7] train GPT with a prefix language modeling objective, Devlin et al. [8] train BERT with a masked language modeling and a next sentence prediction objective, Yang et al. [9] train XLNet with a permutation-based autoregressive objective, and Joshi et al. [10] train SpanBERT with a span prediction objective. Liu et al. [3] show that with additional compute and data, BERT with just masked language modeling (RoBERTa) could perform as well as any other method at the time.

Raffel et al. [4], compare different objectives on an encoder-decoder Transformer (T5), including prefix language modeling, BERT-style masked language modeling, and filling in missing spans, and find that the span replacement task performs substantially better than other methods for most downstream tasks. Clark et al. [6] show that one can use a small generator model to generate inputs for a disciminator model (ELECTRA), resulting in a method that is several times more compute-efficient than RoBERTa and XLNet.

Several works have applied meta-learning to NLP in the few-shot setting. Bansal et al. [11] (LEOPARD) adapt MAML (Finn et al. [12]) to a pretrained BERT model to develop a better initialization for learning downstream tasks in few shots. Bansal et al. [13] create a new pretraining objective, where unlabeled data is used for a classification task with a single masked word as the class, so that they can apply MAML during pretraining. However, both of these works only evaluate their method in the few-shot setting, so it is not clear whether these methods also apply to improving the scalability of pretraining.

All the works mentioned thus far can be seen as using a type of denoising objective, with the input being a perturbation of a sample from the unlabeled dataset, and the desired output being information to recover from the original. This usually results in a pretrain-finetune discrepancy, as the downstream tasks usually take full text samples as the input. Furthermore, none of the methods condition the pretraining procedure on the downstream tasks. This work heavily relies on the approach by Pham

et al. [1] for meta-learning pseudo-labels in computer vision. Pham et al. [1] use a teacher network to generate labels for a student on unlabeled image data, and train it with the student's performance on labeled data. They find that this approach even performs better than other methods using supervised learning on the same pretraining data labeled.

3 Meta Language Modeling

For a typical self-supervised pretraining approach, we might pretrain a model by applying an objective (e.g. masked language modeling) to generate a labeled example from unlabeled data. Afterwards, we might finetune the model using labeled examples from some downstream task. In order to meta-learn, we model the pretraining objective using a teacher network (with the original model being trained as the student) and thus, we can optimize the language modeling objective by optimizing the teacher.

3.1 Problem Statement

Suppose we are given an unlabeled pretraining dataset D_u and a labeled finetuning dataset D_l. Let T and S be a teacher and student network with parameters θ_T and θ_S respectively. That is, for any batch of token sequences x, the teacher network $T(x;\theta_T)$ denotes a probability distribution over possible input-output pairs, e.g. we can sample $\hat{x}, \hat{y} \sim T(x;\theta_T)$. For the student network, $S(x;\theta_T)$ denotes a probability distribution over outputs only. Let $\mathrm{CE}(q,p)$ denote the cross-entropy loss between distributions q and p, where if q is a token, it is understood to be the one-hot distribution, and if q is a sequence or batch of sequences, it is the average cross-entropy.

Instead of unrolling all of pretraining and finetuning, we approximate each with a single gradient step. If the student has parameters $\theta_S^{(0)}$, we first perform a gradient step corresponding to pretraining:

$$\theta_S^{(1)} = \theta_S^{(0)} - \eta_S^{(0)} \nabla_{\theta_S^{(0)}} \mathrm{CE}(\hat{y}_u, S(\hat{x}_u; \theta_S^{(0)})) \tag{1}$$

where we sample pseudo-data $\hat{x}_u, \hat{y}_u \sim T(x_u;\theta_T)$. Then, we perform a second gradient step corresponding to finetuning:

$$\theta_S^{(2)} = \theta_S^{(1)} - \eta_S^{(1)} \nabla_{\theta_S^{(1)}} \mathrm{CE}(y_l^{(0)}, S(x_l^{(0)}; \theta_S^{(1)})) \tag{2}$$

where we sample labeled data $(x_l^{(0)}, y_l^{(0)}) \sim D_l$. The desire is to minimize the final loss, which we denote L, on a different batch of labeled data, i.e.

$$\min_{\theta_T} L = \min_{\theta_T} \mathrm{CE}(y_l^{(1)}, S(x_l^{(1)}; \theta_S^{(2)})) \tag{3}$$

where $(x_l^{(1)}, y_l^{(1)}) \sim D_l$. However, as the problem is currently given, we cannot backpropagate through the sampling step, and thus must add an expected value in order to differentiate.

For the following sections, unless otherwise specified, we take all expected values with respect to the probability distribution of the teacher's output $T(x_u;\theta_T)$, i.e. $\mathbb{E} = \mathbb{E}_{(\hat{x}_u,\hat{y}_u)\sim T(x_u;\theta_T)}$.

3.2 Hard Labels

From the REINFORCE algorithm [14], using the Jacobian formulation of the derivative, we have that

$$\frac{\partial}{\partial\theta} \mathop{\mathbb{E}}_{p(x;\theta)}[f(x)] = \mathop{\mathbb{E}}_{p(x;\theta)}[f(x) \frac{\partial}{\partial\theta} \log p(x;\theta)] \tag{4}$$

for any vector function f not dependent on θ.

Meta Pseudo Labels. Pham et al. [1] add an expectation in (1), i.e. $\bar{\theta}_S^{(1)} = \mathbb{E}[\theta_S^{(1)}]$, and replace the corresponding terms in (2). Using chain rule and the REINFORCE algorithm this leads to the following derivative of (3):

$$\frac{\partial L}{\partial \theta_T} = \frac{\partial \,\mathrm{CE}(y_l^{(1)}, S(x_l^{(1)}; \theta_S^{(2)}))}{\partial \bar{\theta}_S^{(1)}} \mathbb{E}\left[\eta_S^{(0)} \frac{\partial \,\mathrm{CE}(\hat{y}_u, S(\hat{x}_u; \theta_S^{(0)}))}{\partial \theta_S^{(0)}} \cdot N \frac{\partial \,\mathrm{CE}((\hat{x}_u, \hat{y}_u), T(x_u;\theta_T))}{\partial \theta_T}\right] \tag{5}$$

where $N = \dim(\hat{x}_u) + \dim(\hat{y}_u)$ so that we can use the (average) cross entropy loss instead of the negative log of the probability distribution from $T(x_u; \theta_T)$. Note that the $\theta_S^{(0)}$ term in (1) gets omitted from the expectation since the derivative with respect to θ_T (before applying (4)) is zero. As is usual practice, we drop the N term so that the update size is unaffected by the sequence length, which is equivalent to weighting each sequence by the inverse of its length.

Pham et al. [1] omit (2), i.e. $\theta_S^{(2)} = \bar{\theta}_S^{(1)}$, since their pretrain task and finetune task are similar. By taking a Monte Carlo approximation of (5), they obtain the gradient

$$\nabla_{\theta_T} L = h \cdot \nabla_{\theta_T} \operatorname{CE}((\hat{x}_u, \hat{y}_u), T(x_u; \theta_T)). \tag{6}$$

where the scalar, $h = \eta_S^{(0)} \nabla_{\theta_S^{(1)}} \operatorname{CE}(y_l^{(1)}, S(x_l^{(1)}; \theta_S^{(1)})) \nabla_{\theta_S^{(0)}} \operatorname{CE}(\hat{y}_u, S(\hat{x}_u; \theta_S^{(0)}))$, adjusts the teacher's update by the dot product of the student's two gradients. In addition, they use the *cosine similarity* $S_C(A, B) = \frac{A \cdot B}{\|A\| \|B\|}$ instead of the dot product for numerical stability.

First Order Approximation. In our case, we keep (2), and the derivative with respect to $\bar{\theta}_S^{(1)}$ is simply the MAML [12] gradient. We take the first order approximation (first-order MAML), which is the derivative of the last step, i.e.

$$\frac{\partial \operatorname{CE}(y_l^{(1)}, S(x_l^{(1)}; \theta_S^{(2)}))}{\partial \bar{\theta}_S^{(1)}} \approx \frac{\partial \operatorname{CE}(y_l^{(1)}, S(x_l^{(1)}; \theta_S^{(2)}))}{\partial \theta_S^{(2)}} \tag{7}$$

which instead gives the scalar $h = \eta_S^{(0)} \nabla_{\theta_S^{(2)}} \operatorname{CE}(y_l^{(1)}, S(x_l^{(1)}; \theta_S^{(2)})) \nabla_{\theta_S^{(0)}} \operatorname{CE}(\hat{y}_u, S(\hat{x}_u; \theta_S^{(0)}))$.

For adaptive optimizers (e.g. Adam [15]), we can instead compute h as

$$h = S_C(\nabla_{\theta_S^{(2)}} \operatorname{CE}(y_l^{(1)}, S(x_l^{(1)}; \theta_S^{(2)})), (\theta_S^{(0)} - \theta_S^{(1)})) \tag{8}$$

where S_C is the cosine similarity.

Note on Expectation. Another approach, which we did not explore but could be an interesting formulation, is instead to take the expectation after (3), i.e. $\bar{L} = \mathbb{E}[L]$. Then applying the REINFORCE algorithm and approximating immediately gives $h = -L$, i.e. $-L$ is the teacher's reward after selecting the input-output pair.

3.3 Soft Labels

In the case that the student network can directly take soft input-output pairs, the entire process becomes fully differentiable. This can be seen as replacing $\hat{x}_u, \hat{y}_u$ with their expectations $\mathbb{E}[\hat{x}_u]$, $\mathbb{E}[\hat{y}_u]$. In most cases, such a modification is straightforward. For example, if the first layer of the student network is a learned embedding layer, as is the case with a Transformer architecture, we can simply replace it with a linear one and input the probabilities from the teacher. Furthermore, the cross entropy naturally accepts any probabilities as the label. With this approach, we need to keep the entire computation graph in memory, but we also receive a gradient for every token in $\hat{x}_u$ and $\hat{y}_u$ instead of just a single scalar h.

3.4 Auxiliary Losses

In addition to the meta loss given in (3), we add two additional losses for the teacher. We train the teacher with both to generate a masked language modeling (MLM) task as well as perform MLM [8]. That is, we take the same unlabeled example x_u, except we give it an MLM task token to denote it as an MLM task. Then, let x_m, y_m denote a (random) MLM task example from x_u. We train the teacher to output x_m, x_u given x_u, and to output x_m, y_m given x_m. Note that we cannot train the teacher to output x_m, y_m given x_u since then whether a token in y_m is used in the loss depends on whether the corresponding token in x_m is a mask token, and the tokens then cannot be sampled independently. For the total teacher loss, we give the meta loss a weight $\lambda_{\texttt{meta}}$.

4 Experiments

4.1 Experimental Setup

Training Procedure. We divide the training into two phases: pretraining and finetuning. During pretraining, we jointly train the teacher and student on their respective losses for a fixed number of unlabeled examples. At every step, we sample a batch of examples from the unlabeled dataset, and two batches (a support and query set) from the labeled dataset. For pretraining with *soft labels*, we augment the labeled dataset with the MLM task on unlabeled data, which we sample with a $1/2$ probability. Otherwise, the labeled task is sampled uniformly regardless of the number of examples for each. We always sample the support and query example from the same task, and in order to ensure the student receives enough support examples of each label, we repeatedly sample from the same task a configurable number of times, usually equal to the batch size.

Note that during the pretraining phase, the student is only trained on pseudo-data generated by the teacher. We select the latest model for finetuning, regardless of the validation performance. During the finetuning phase, the student then trains on labeled examples, and we select the best model via performance on a validation set.

Gradient Accumulation. We implement gradient accumulation for both the student and teacher. Note that for the teacher, this is subtly different than increasing the batch size, since for gradient accumulation, we perform multiple sets of student gradient steps to calculate the teacher loss on each batch. However, no matter how much we increase the batch size, the student still only takes one set of gradient steps on that batch. For the teacher, this can be important for training stability, as it can decrease the stochasticity.

Models. In each experiment, we use the same model architecture and sizes for the teacher and student, with separate weights. We use a Transformer encoder architecture [16, 8], except with layer normalization placed at the start of each residual block instead of after. For the teacher, we attach two language modeling heads, one for generating the input and one for the output. We use a byte level byte pair encoding tokenizer to convert any text into tokens [17]. We prepend the text with a special classification token with a unique token per task, and use a separator token for separating or terminating any token sequences.

Optimization. We use the AdamW optimizer [15, 18] during pretraining and finetuning for the outer step of both the teacher and student with the same hyperparameters. For the student's inner step (2), we use SGD with an inner learning rate of 0.1. We also use an inverse square root learning rate schedule $\texttt{lr}/\sqrt{n}$ with some number of linear warmup optimizer steps. This may be a different formulation than other papers, so as an example, for 10k warmup steps, the learning rate increases during the warmup period until it hits the peak learning rate of $\texttt{lr}/100$, and will later halve at 40k steps.

4.2 Two Moons Experiment

Following Pham et al. [1], we study the approaches on a simple Two Moons dataset [19] to illustrate the behavior on a small scale. We compare both hard and soft labels against an MLM objective [8, 3].

Dataset. We scale and floor all coordinates to integers in the range $[0, 64)$ and convert them to space-separated text. For pretraining, we use 32k unlabeled text sequences of sorted x, y coordinates, each with 16 datapoints from the original Two Moons dataset. For finetuning, we use either 10, 20, 40, 80, or 160 labeled examples with a validation set of the same size. We use a sufficiently large vocabulary size (512) for the tokenizer so that every coordinate is compressed to a single token. Note that the conversion to tokens makes the task more difficult, since we lose any inductive bias from the linearity of the coordinates, and sorting the datapoints by coordinate imposes a non-Euclidean bias.

Training details. We use an extremely small Transformer model with only 2 layers, a hidden size of 8, a feed forward size of 32, and 2 attention heads. We pretrain for 640k samples, with batch size 4, repeat sampling 4 times per task, gradient accumulation 4, warmup of 1k steps, and a learning rate of 0.08. For soft labels, we use $\lambda_{\texttt{meta}} = 100$ and for hard labels, we use $\lambda_{\texttt{meta}} = 1$. For finetuning, we

use a learning rate of 0.04 with only 10 warmup steps, and train for 100 epochs. For computational reasons, we truncate the validation set to 128 examples or less. All training is done on a CPU. We give the full set of training details in Appendix A.

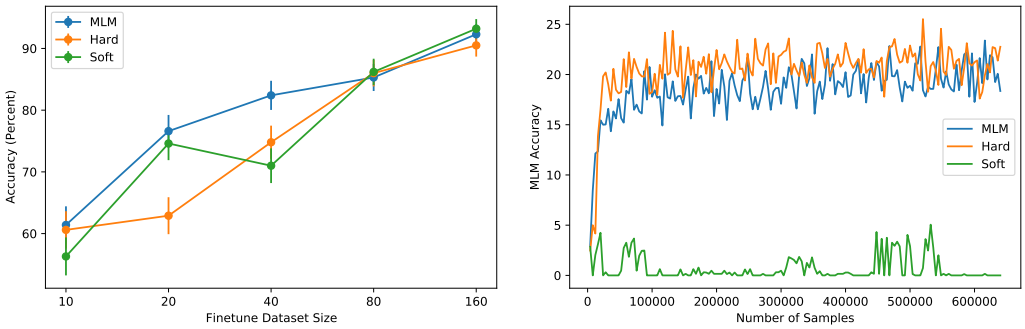

Figure 1: **Left**. The test performance of each of the objectives with a finetune dataset of different sizes. **Right.** The performance of each of the objectives with the 160 example finetune dataset on an MLM task during pretraining.

Results. We evaluate on a test set of 1k randomly sampled datapoints, and also plot visualizations on the entire possible 64 by 64 input set with 4096 examples. MLM performed the best overall across different finetune dataset sizes (Figure 1). The soft labels objective had slightly higher but not statistically significant performance for larger finetune dataset sizes. We initially had different results, which showed MLM performing much worse on a larger finetune dataset, which we discuss in Appendix B. We give a visualization of models' outputs for one of the runs on the test set in Figure 2.

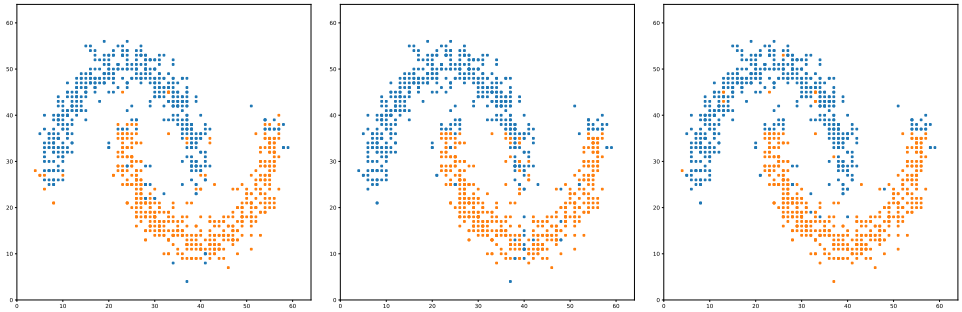

Figure 2: The outputs from the models on the test set, with a finetune dataset of 160 examples. **Left**. MLM. **Middle**. Hard Labels. **Right**. Soft Labels.

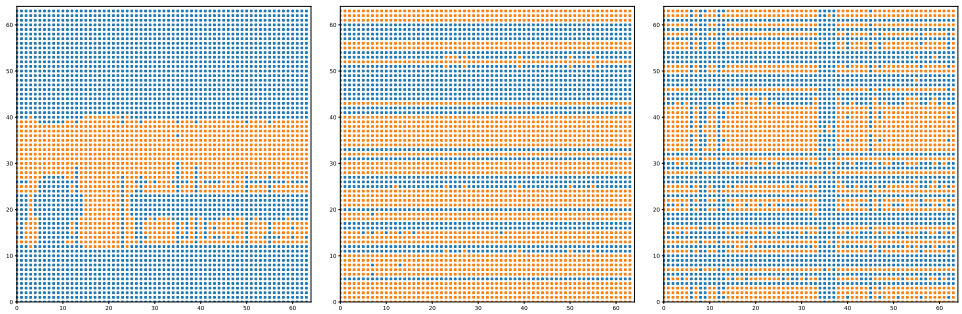

Figure 3: The outputs from the models on every possible coordinate for different objectives, with a finetune dataset of only 10 examples. **Left**. MLM. **Middle**. Hard Labels. **Right**. Soft Labels.

Analysis. By looking at the outputs on the smallest finetune dataset (Figure 3), we can visualize inductive bias of each model trained under the different objectives. We can see that each objective prefers to group points based on the y-coordinate, with soft labels also grouping based on the x-coordinate. The MLM objective results in a coarser classification, whereas the hard and soft labels objectives generate finer classification boundaries. Since the meta objectives allow the downstream task to influence the pretraining objective, this may result in a bias that's overfit to the small finetune dataset.

We also observe that both hard labels and MLM students become good at MLM during pretraining (Figure 1). This is since the hard labels teacher never strays from generating MLM tasks (Table 1, whereas the soft labels teacher generates a mixture between an MLM task and the finetuning classification task. We posit that since the hard labels teacher only receives a single scalar of feedback, whereas the soft labels teacher receives a gradient for each possible token in the sequence and is more easily able to adjust the generated pseudo-data.

x_u	[CLS0]6 24 15 52 23 36 25 51 27 24 27 28 28 26 28 34 28 55 30 12 31 46 39 33 47 14 47 16 56 28 56 39[SEP]	[CLS0]20 50 21 50 24 36 26 33 26 35 27 22 29 16 29 19 29 52 36 41 38 33 40 28 45 9 47 18 55 27 56 32[SEP]
$\hat{x}_u$	[CLS1]6 24 15 52[MASK] 36 25 51[MASK] 24 27 28 28 26 28 34 28 55 30 12 31& 39 33 47 14 47 16 56 28 56 39[SEP]	[SEP][MASK] 50 21 50[MASK][MASK] 26[MASK] 26 46 27 34 52 16 29 19 29 52[MASK] 41 37[MASK] 40 28[MASK] 7 47 18 55 27 56 32[SEP]
$\hat{y}_u$	[CLS1]6 24 15 52 23 36 25 51 27 24 27 28 28 26 28 34 28 55 30 12 31 46 39 33 47 14 47 16 56 28 56 39[SEP]	528 50 21 50 24 42 26 56[CLS2] 46 486 52[CLS2] 29 19 29 52 36 41 37 56 40 28 47 7 50 18 55[PAD] 56 32[SEP]

Table 1: A sample of the teacher's generated pseudo-data on unlabeled data in the Two Moons dataset for hard and soft labels. **Left**. Hard Labels. **Right**. Soft Labels

4.3 Wikipedia and SuperGLUE Experiment

For our NLP experiment, we use Wikipedia as the unlabeled dataset and SuperGLUE [20] as the set of downstream tasks. For this experiment we only compare soft labels against an MLM objective.

Dataset. Wikipedia [1] (17GB) contains 6M articles of English text. The tokenizer is trained with a vocabulary size of 30k tokens using the full dataset. We split out 1k articles as a validation set. For computational reasons, we uniformly sample a random character from the whole set of articles, then tokenize the article into disjoint windows each with the same number of tokens depending on the model's sequence length, and select the window containing the selected character. This way, we can sample a sequence without having to tokenize all of Wikipedia beforehand.

SuperGLUE [20] contains a set of 8 tasks [21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31] each with their own dataset and evaluation method. We convert each of these tasks into a common format with the input and output specified by token sequences so that we can use the same language modeling head throughout training. For example, we might assign a token for a given label, and the desired output is the token repeated for the length of the input sequence.

Training details. For computational reasons, we use a small Transformer model with 6 layers, a hidden size of 256, a feed forward size of 1024, 4 attention heads, and a maximum sequence length of 128. We pretrain for 6M samples, with batch size 16, repeat sampling 16 times per task, gradient accumulation 4, warmup of 10k steps, and a learning rate of 0.08. For soft labels, we use $\lambda_{\texttt{meta}} = 10$. We finetune each task separately, use a learning rate drawn from $\{0.04, 0.01, 0.005\}$ with only 100 warmup steps, and train for 100 epochs. For computational reasons, we truncate each validation set to 128 examples or less. All training is done on a single Nvidia V100 GPU. We give the full set of training details in Appendix A.

[1] The English Wikipedia dump from `https://dumps.wikimedia.org/enwiki/` which we use a cleaned version (20200501.en) at `https://huggingface.co/datasets/wikipedia`.

Model	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE	WiC	WSC	Overall
	Acc	F1/Acc	Acc	F1a/EM	F1/EM	Acc	Acc	Acc	Avg
MLM	66.1	67.3/71.4	57.0	52.0/10.4	12.7/10.6	56.0	56.1	72.1	52.4
Soft	60.0	48.3/69.6	57.0	42.4/4.9	2.2/1.2	54.2	53.6	63.5	46.6

Table 2: SuperGLUE validation set results for models trained using different pretraining objectives.

Results. For each task, we select the model with the best truncated validation set performance among all the learning rates. We then give the score on the full validation set in Table 2. The model trained with the MLM objective performed better on all tasks compared to the soft labels objective, which performs slightly worse than a most frequent class baseline given in the original SuperGLUE paper [20]. The MLM pretraining took about 11h whereas the soft labels pretraining took approximately 41h or 3.7x longer.

x_u	`[CLS0]-5.00 ERA. On offense, the Athletics also struggled; the loss of their two best players (Mark McGwire and Rickey Henderson) to injury and a trade, respectively, contributed to their scoring only 715 runs (10th of 14 AL teams). The Athletics' 68-94 finish was their worst since 1982. Moreover, the 1993 Athletics (as of 2018) remain the only team in Oakland history to finish last in the AL West after finishing first one-year earlier. Offseason October 27, 1992: Jamie Quirk was released by the Athletics. November 17, 1992[SEP]`
$\hat{x}_u$	` that that that that that that that that that that that that [...]`
$\hat{y}_u$	`[CLS9][CLS9][CLS9][CLS9][CLS9][CLS9][CLS9][CLS9][CLS9][CLS9][...]`

Table 3: A sample of the teacher's generated pseudo-data on unlabeled data on Wikipedia for soft labels.

Analysis. From a look at the generated pseudo-data (Table 3), we can see that the teacher gets stuck generating examples aiming to replicate the downstream tasks. Furthermore, when generating the new example, the initial text gets destroyed, and the new input often doesn't make sense. We posit that the soft labels may make the teacher too easily influenced by the downstream data and overfit.

5 Conclusion

We extended the teacher-student framework of Meta Pseudo Labels [1] to an NLP context, by modeling the pretrain-finetune process as two gradient steps. However, we find that meta-learning in this context, with high-dimensional inputs and outputs, may give the teacher too many degrees of freedom, and cause it to overfit the unlabeled data to the simpler downstream task. However, more gradients from the student to the teacher, e.g. with soft labels, may also be useful for the teacher to learn a better objective. One way to solve these issues might be to constrain the teacher's possible outputs, e.g. by only letting the teacher specify the input of the pseudo-data, which turns the task into a denoising objective (see Appendix B). Another might be by adding more inner steps to the meta-objective, so that the teacher won't overfit to teaching the downstream task. We hope future work looks deeper at solving this tension to develop a better meta-objective.

References

[1] Hieu Pham, Zihang Dai, Qizhe Xie, Minh-Thang Luong, and Quoc V. Le. Meta pseudo labels. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021. URL `https://arxiv.org/abs/2003.10580`.

[2] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020. URL `https://arxiv.org/abs/2001.08361`.

[3] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT

pretraining approach. *CoRR*, abs/1907.11692, 2019. URL http://arxiv.org/abs/1907.11692.

[4] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

[5] Sharan Narang, Hyung Won Chung, Yi Tay, William Fedus, Thibault Fevry, Michael Matena, Karishma Malkan, Noah Fiedel, Noam Shazeer, Zhenzhong Lan, Yanqi Zhou, Wei Li, Nan Ding, Jake Marcus, Adam Roberts, and Colin Raffel. Do transformer modifications transfer across implementations and applications?, 2021.

[6] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020. URL https://openreview.net/pdf?id=r1xMH1BtvB.

[7] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. 2018.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

[9] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf.

[10] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. SpanBERT: Improving pre-training by representing and predicting spans. *arXiv preprint arXiv:1907.10529*, 2019.

[11] Trapit Bansal, Rishikesh Jha, and Andrew McCallum. Learning to few-shot learn across diverse natural language classification tasks. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING)*, 2020.

[12] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/finn17a.html.

[13] Trapit Bansal, Rishikesh Jha, Tsendsuren Munkhdalai, and Andrew McCallum. Self-supervised meta-learning for few-shot natural language classification tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 522–534, 2020.

[14] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015. URL http://arxiv.org/abs/1412.6980.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett,

editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[17] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://www.aclweb.org/anthology/P16-1162.

[18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[19] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors. *Semi-Supervised Learning*. The MIT Press, 2006. ISBN 9780262033589. URL http://dblp.uni-trier.de/db/books/collections/CSZ2006.html.

[20] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint 1905.00537*, 2019.

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

[22] Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. The CommitmentBank: Investigating projection in naturally occurring discourse. 2019. To appear in proceedings of Sinn und Bedeutung 23. Data can be found at https://github.com/mcdm/CommitmentBank/.

[23] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of NAACL-HLT 2019*, 2019.

[24] Mohammad Taher Pilehvar and Jose Camacho-Collados. WiC: The word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of NAACL-HLT*, 2019.

[25] Hector J Levesque, Ernest Davis, and Leora Morgenstern. The Winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, volume 46, page 47, 2011.

[26] Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer, 2006.

[27] Roy Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. The second PASCAL recognising textual entailment challenge. 2006.

[28] Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9. Association for Computational Linguistics, 2007.

[29] Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. The fifth PASCAL recognizing textual entailment challenge. 2009.

[30] Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. 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 252–262, 2018.

[31] Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. ReCoRD: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint 1810.12885*, 2018.

A Training Details

A.1 Two Moons

We give the model hyperparameters in Table 4 with both the student and teacher sharing the same sizes. We give the pretrain hyperparameters for each objective, with both the student and teacher sharing the same hyperparameters in Table 5. The only difference between the objectives is the meta loss weight. We only use the inner learning rate in MLM for reporting h values during training. We give the finetune hyperparameters for each objective over the five runs in Table 6, which are the same for each objective.

Hyperparameter	Value
Layers	2
Hidden Size	8
FNN inner hidden size	32
Attention Heads	2
Activation	GELU
Attention Dropout	0.1
Dropout	0.1
Activation Dropout	0.0
Max Positions	34

Table 4: Model size for Two Moons.

Hyperparameter	MLM	Hard	Soft
Gradient Clipping	5.0	5.0	5.0
Adam ϵ	1e-6	1e-6	1e-6
Adam β_1	0.9	0.9	0.9
Adam β_2	0.98	0.98	0.98
Weight Decay	0.01	0.01	0.01
Batch Size	4	4	4
Samples per task	4	4	4
Gradient Accumulation	4	4	4
Epoch Size	32000	32000	32000
Epochs	20	20	20
Warmup steps	1000	1000	1000
Learning Rate (Peak)	0.08 (2.5e-3)	0.08 (2.5e-3)	0.08 (2.5e-3)
Inner LR	0.1	0.1	0.1
Mask Probability	0.15	0.15	0.15
Random Mask Probability	0.1	0.1	0.1
Unmask Probability	0.1	0.1	0.1
Weight $\lambda_{\texttt{meta}}$	-	1.0	100.0
MLM as Labeled Task	-	False	True

Table 5: Pretraining hyperparameters for Two Moons.

A.2 NLP

We give the model hyperparameters in Table 7 with both the student and teacher sharing the same sizes. We give the pretrain hyperparameters for each objective, with both the student and teacher sharing the same hyperparameters in Table 8. We only use the inner learning rate in MLM for reporting h values during training. We give the finetune hyperparameters for each objective, which is the same for each task, in Table 9.

B Informal Experiments

In this section, we give a brief, informal summary of interesting experiments that we performed but did not make it to the main section of the paper.

Hyperparameter	MLM	Hard	Soft
Gradient Clipping	5.0	5.0	5.0
Adam ϵ	1e-6	1e-6	1e-6
Adam β_1	0.9	0.9	0.9
Adam β_2	0.98	0.98	0.98
Weight Decay	0.01	0.01	0.01
Batch Size	1, 1, 1, 8, 16	1, 2, 4, 8, 16	1, 2, 4, 8, 16
Gradient Accumulation	1	1	1
Train Dataset Size	10, 20, 40, 80, 160	10, 20, 40, 80, 160	10, 20, 40, 80, 160
Epochs	100	100	100
Warmup steps	10	10	10
Learning Rate (Peak)	0.04 (1.2e-2)	0.04 (1.2e-2)	0.04 (1.2e-2)

Table 6: Finetuning hyperparameters for Two Moons over the five runs. Only the dataset size and batch size vary from run to run.

Hyperparameter	Value
Layers	6
Hidden Size	256
FNN inner hidden size	1024
Attention Heads	4
Activation	GELU
Attention Dropout	0.1
Dropout	0.1
Activation Dropout	0.0
Max Positions	128

Table 7: Model size for the NLP experiment.

Hyperparameter	MLM	Soft
Gradient Clipping	5.0	5.0
Adam ϵ	1e-6	1e-6
Adam β_1	0.9	0.9
Adam β_2	0.98	0.98
Weight Decay	0.01	0.01
Batch Size	64	16
Samples per task	16	16
Gradient Accumulation	1	4
Epoch Size	20000	20000
Epochs	300	300
Warmup steps	10000	10000
Learning Rate (Peak)	0.08 (8e-4)	0.08 (8e-4)
Inner LR	0.1	0.1
Mask Probability	0.15	0.15
Random Mask Probability	0.1	0.1
Unmask Probability	0.1	0.1
Weight $\lambda_{\texttt{meta}}$	-	10.0

Table 8: Pretraining hyperparameters for the NLP experiment. Note that for MLM, the batch size of 64 and gradient accumulation of 1 is equivalent to a batch size of 16 and gradient accumulation of 4.

Hyperparameter	MLM	Soft
Gradient Clipping	5.0	5.0
Adam ϵ	1e-6	1e-6
Adam β_1	0.9	0.9
Adam β_2	0.98	0.98
Weight Decay	0.0	0.0
Batch Size	16	16
Gradient Accumulation	1	1
Epochs	20	20
Warmup steps	100	100
Learning Rate	{0.04, 0.01, 0.005}	{0.04, 0.01, 0.005}
(Peak Learning Rate)	{4e-3, 1e-3, 5e-4}	{4e-3, 1e-3, 5e-4}

Table 9: Finetuning hyperparameters for the NLP experiment over all the tasks. The learning rate is varied for each task.

1. For Two Moons, we had initially chosen the model with the best MLM loss during pretraining for the MLM objective. Compared to the MLM model selected from the latest checkpoint, this model performed better with fewer finetune datapoints, but worse with additional examples. See Table 10.
2. For Two Moons, we also did a couple runs where we fix the teacher's output $\hat{y}_u$ to be equal to the unlabeled text. This effectively makes it a denoising objective. On the settings we tested, this performed better than all other methods (Table 10).
3. For Two Moons with soft labels, we initially had the inner optimizer be the same as the outer optimizer (AdamW). Since the required learning rates may be different, changing this to SGD allowed the teacher to generate more sensible examples.
4. For Two Moons with soft labels, we initially did not include the MLM task as an additional labeled task. Thus, the teacher tended to generate examples that only taught the downstream task. Including the MLM task made the teacher generate more mixed outputs.
5. We initially did an experiment in the style of Meta Pseudo Labels, e.g. without (2), on the NLP setup. Without the additional inner step, the teacher learns mostly to copy the unlabeled text, i.e. $\hat{x}_u = x_u$, and provide a rather poor label distribution for $\hat{y}_u$.
6. We tracked the h value for MLM throughout the pretraining for the NLP set up. We do not have a point of comparison, since we did not run a hard labels experiment. However, it was always positive during pretraining, and the cosine similarity version averaged around 8e-4 and the unscaled version was around 1e-3.

Model	10 examples	20 examples	40 examples	80 examples	160 examples
MLM (latest)	61.40	76.60	82.40	85.30	92.30
MLM (best loss)	73.40	81.20	82.80	83.50	88.90
Hard	60.60	62.90	74.80	86.00	90.50
Soft	56.30	74.60	71.00	86.20	93.20
Soft (fixed y)	75.10	-	-	-	94.00

Table 10: Results for Two Moons.