PEST: Combining Parameter-Efficient Fine-Tuning with Self-Training and Co-Training

Hunter Lang MIT CSAIL
Monica Agrawal MIT CSAIL
Yoon Kim MIT CSAIL
David Sontag MIT CSAIL

Abstract

We demonstrate how to improve the zero-shot and few-shot performance of large language models (LLMs) by using the T-Few parameter-efficient fine-tuning method (Liu et al., 2022) with self-training or co-training. Our methods apply to settings where labeled data is very limited, but unlabeled data is plentiful. Specifically, we combine T-Few with (i) the co-training techniques of Lang et al. (2022a), and (ii) SETRED, a self-training algorithm that uses a very simple data selection criterion (Li and Zhou, 2005). By using the efficient T-Few method, we are able to scale co-training to larger models (from T0-3B to T0-11B) and cut down on wallclock training time, improving the zero-shot co-training results of Lang et al. (2022a). By performing multiple iterations of self- or co-training, we significantly improve over the few-shot performance of T-Few reported by Liu et al. (2022) without using any additional labeled data. Our methods are relatively fast (2.5 hours to self-train T0-11B on a single A100 80GB) and allow T0-11B to match the few-shot performance of models with an order of magnitude more parameters.

1 Introduction

Large language models such as GPT-3 (Brown et al., 2020), FLAN (Wei et al., 2021), and T0 (Sanh et al., 2022) perform well at few-shot and even zero-shot learning for many tasks, but there is still a large gap between their few-shot performance and the state-of-the-art fully-supervised performance. As a result, several prior works have focused on parameter- and data-efficient methods for improving the accuracy of these models with a small amount of labeled data.

To further boost performance, we combine a parameter-efficient fine-tuning method called T-Few (Liu et al., 2022) with two popular semi-supervised learning methods: self-training (e.g., Scudder, 1965; Yarowsky, 1995) and co-training (Blum and Mitchell, 1998). These methods both start with an initial model f_0, which may have been trained on a small set of (possibly pseudo-)labeled data. In iteration t, model f_t is used to pseudolabel a large unlabeled dataset, yielding $\{(x_i, f_t(x_i))\}_{i=1}^N$. A *data selection method* is then used to select a good subset of the pseudolabeled data, which we use to fine-tune the next model f_{t+1}. In self-training, we fine-tune the *same* model architecture in every round. In co-training, we alternate the training between two different model types.

In this work, we use T-Few to efficiently fine-tune T0 in each iteration of self-training and co-training. We use the *cut statistic* (Muhlenbach et al., 2004) as the data selection criterion for both approaches, similar to the SETRED (Li and Zhou, 2005) and CoTRADE (Zhang and Zhou, 2011) algorithms, respectively. The resulting parameter-efficient self-training (PEST) and parameter-efficient co-training (PECT) methods achieve strong few-shot and zero-shot performance on a variety of NLP datasets, often matching or exceeding the few-shot performance of GPT-3 175B and FLAN-137B, models with an order-of-magnitude more parameters. Compared to prior work co-training T0 (Lang et al., 2022a), PECT achieves the same performance with more than 10x reduction in wall-clock time.

2 Related Work

Several prior works combine self-training with large language models; these methods primarily differ in their data selection methods. For example, Wang et al. (2021), Ye et al. (2020), and Mukherjee and Awadallah (2020) use meta-learning, reinforcement learning, and uncertainty estimates from Bayesian deep learning methods (respectively) to learn how to select and weight pseudolabels for each round of self-training. However, these approaches all rely on having access to some labeled data in order to learn the selection module.

The closest works to ours are Wang et al. (2022) and Lang et al. (2022a). Wang et al. (2022) combine self-training with cloze-style prompt tuning (Gao et al., 2021; Schick and Schütze, 2021) for data and parameter efficiency. However, in each iteration, their method uses a meta-learning procedure to choose good pseudolabeled samples that is more complex than our selection criterion and relies on a small amount of labeled data. Our selection procedure is nonparametric, does not rely on labeled data, and can be implemented in less than thirty lines of code. Additionally, we consider parameter-efficient tuning methods for large encoder-decoder language models (T5, T0) rather than cloze-style prompt tuning, which was developed for encoder-only models (e.g., BERT, RoBERTa).

Lang et al. (2022a) combine *soft* prompt tuning (Lester et al., 2021) with a variant of co-training (Blum and Mitchell, 1998; Zhang and Zhou, 2011) that uses the same selection method as our work. Unlike `T-Few`, soft prompt tuning requires many gradient steps to obtain the best possible performance; following Lester et al. (2021), Lang et al. (2022a) tunes for 30000 steps per iteration. Therefore, each co-training iteration for T0-3B takes 7–12 hours on an A100 80GB GPU. Additionally, the batch size used by Lester et al. (2021) and Lang et al. (2022a) (32 examples) means that scaling the method up to a larger model such as T0-11B requires too much gradient accumulation to be practical on a small compute budget. In this work, we replace soft prompt tuning with `T-Few`. The efficiency gain allows us to scale co-training up to T0-11B, and for T0-3B, our method obtains the same or better performance as Lang et al. (2022a) with greater than 10x reduction in wallclock time.

3 Method

3.1 PEST: Parameter-Efficient Self-Training

To perform self-training with the T0 model (Sanh et al., 2022), we combine the SETRED algorithm (Li and Zhou, 2005) with `T-Few` (Liu et al., 2022) for fine-tuning. SETRED differs from regular self-training by using a data selection procedure known as the cut statistic (Muhlenbach et al., 2004) in each round. Intuitively, we should be more confident in groups of similar examples that are all assigned the same pseudolabel. The cut statistic makes this intuition quantitative by constructing a nearest neighbor graph over the pseudolabeled examples and ranking examples according to how constant the pseudolabel is on their neighborhood. This nonparametric selection method relies on a good representation of the input examples. Recent work (Lang et al., 2022a,b) showed that using the cut statistic on top of large language model representations results in high-quality pseudolabeled training datasets. We include a detailed description of the cut statistic method in Appendix A.

Formally, for each round $t \in \{0, \ldots, T-1\}$ of self-training T0, we select (up to) $K \cdot b^t$ examples per (pseudo)class. Our experiments exclusively use $K = 16$ and $b = 2$. For zero-shot self-training, the initial K examples for each class are selected using the cut statistic and the pre-trained T0 model as f_0. For few-shot self-training, we first select a small set (typically 32) of random gold examples (total, not per class), then run one round of `T-Few` to obtain f_0. The training data for subsequent rounds $t \geq 1$ is the union of the gold examples and the $K \cdot b^t$ top examples for each (pseudo)class according to the cut statistic. Following Lang et al. (2022a), we use the hidden state of the first decoded token in the T0 decoder's last layer as the representation for the cut statistic. Note that only the gold examples (if any) are guaranteed to be used in every training round.

3.2 PECT: Parameter-Efficient Co-Training

To perform co-training with T0 and BERT as the two model types, we follow Lang et al. (2022a), but replace the soft-prompt fine-tuning with `T-Few`. The only other difference is that when selecting data for T0 fine-tuning, we select a small number per (pseudo)-class, as we do in `PEST`. This is

because T-Few works best when using a small number of highly accurate examples. Lang et al. (2022a) instead selects a constant fraction β of the unlabeled examples when fine-tuning T0.

Formally, for each co-training iteration $t \in \{0, \ldots, T-1\}$ we select $N(\beta_0 + t\beta)$ examples to fine-tune BERT, where N is the total number of unlabeled examples. We use the final-layer hidden state of the T0 decoder for the cut statistic when selecting the data for fine-tuning BERT. In iteration t, we select $K \cdot b^t$ examples to fine-tune T0, as in PEST. We use the final-layer CLS token representation from the BERT model for the cut statistic when selecting the dataset for fine-tuning T0.

4 Experiments

4.1 Setup

Self-training. For self-training, we start with $K = 16$ examples per class and increase by a $b = 2$ factor each iteration, so there is up to $16 \cdot 2^t$ pseudolabeled training examples per class in iteration t. We run for $T = 5$ iterations, so the final round consists of training on (up to) 256 pseudolabeled examples per class. As described in the previous section, the initial K examples for each class are selected from the pseudolabeled data using the cut statistic in the zero-shot case. For the few-shot case, we randomly choose 32 gold examples (total) for the initial fine-tuning round. For each round, we run T-Few according to Liu et al. (2022): we train for 1000 steps with Adafactor using learning rate 3e-3 and batch size 8, and do not use an additional validation set for model selection.

Co-training. For co-training, we follow Lang et al. (2022a) almost exactly. We run $T = 5$ co-training iterations. To select a training set for T0 in each round, we select $Kb^t = 16 \cdot 2^t$ examples per class using the cut statistic over the BERT representations from the previous round. Aside from using T-Few instead of soft prompt tuning, this is the only difference from the setup in Lang et al. (2022a). To select a training set for BERT in each round, we select a $\beta_0 + \beta t$ fraction of the unlabeled data with $\beta_0 = 0.5$, $\beta = 0.1$. We use the cut statistic ranking on the final decoder hidden state of the T0 model from the previous round. We fine-tune the last three layers (last transformer layer, pooler, and linear layer) of DeBERTa-large (He et al., 2021) for 40 epochs using AdamW, weight decay 0.01, learning rate 1e-5, and batch size 16. As in Lang et al. (2022a) we perform model selection using *pseudo*-labeled validation data also selected using the cut statistic, and we use the MNLI-pretrained model on RTE and CB.

Datasets. We evaluate PEST on several natural language benchmarks for which there is a nontrivial gap between few-shot learning and full supervision (Wang et al., 2018; Brown et al., 2020). We used

Table 1: Results for PEST (T-Few + self-training) on several datasets. These results use the full T0-11B model and the T0 numbers are taken directly from Sanh et al. (2022) with the exception of BoolQ, which is not reported in the T0 paper. Standard deviations across five random seeds are reported. Each seed takes 2.5 hours on a single A100 80GB GPU. †Our reproduction of a baseline.

Method	RTE	CB	WiC	WSC
T0 zero-shot (median of single-prompt performance)	81.2	78.6	57.2	64.4
T0 zero-shot (random prompt per example)	$80.7_{0.8}$	$69.6_{4.7}$	$56.7_{0.9}$	$61.5_{1.4}$
PEST zero-shot	$83.5_{1.3}$	$72.9_{5.9}$	$55.7_{0.6}$	$67.7_{5.9}$
T-Few 32-shot (Liu et al., 2022)	$84.5_{2.8}$	$83.9_{5.4}$	$60.8_{6.4}$	$73.1_{6.3}$
PEST 32-shot	$86.0_{0.6}$	$87.1_{3.5}$	$61.8_{4.0}$	$74.8_{4.0}$
FLAN 137B few-shot (best dev) (Wei et al., 2021)	84.5	82.1	57.8	70.2
GPT-3 175B few-shot (best dev) (Brown et al., 2020)	72.9	82.1	55.3	75.0

	BoolQ	COPA	HellaSwag	Winogrande
T0 zero-shot (median of single-prompt performance)	70.7	90.8	33.7	60.7
T0 zero-shot (random prompt per example)	$72.7_{0.4}$	$90.8_{0.8}$	$33.3_{0.4}$	$60.8_{0.8}$
PEST zero-shot	$82.6_{1.4}$	$94.2_{1.7}$	$40.5_{6.4}$	$76.7_{0.3}$
T-Few 32-shot (Liu et al., 2022)	$81.9^{\dagger}_{2.3}$	$92.0_{2.0}$	$64.5_{6.6}$	$72.7_{1.0}$
PEST 32-shot	$84.2_{0.8}$	$94.2_{1.2}$	$64.6_{4.5}$	$77.8_{0.2}$
FLAN 137B few-shot (best dev) (Wei et al., 2021)	84.6	87.0	59.2	72.8
GPT-3 175B few-shot (best dev) (Brown et al., 2020)	77.5	92.0	79.3	77.7

Table 2: Comparison between co-training methods for T0-3B: PECT / T-Few versus soft prompt tuning (SPT). Using T-Few in the co-training procedure obtains better performance at far less wallclock time due to the fewer number of steps required per iteration. Co-training using SPT takes 7–12 hours *per iteration*, whereas co-training T0-3B using T-Few takes less than 2 hours *total*.

Model	**T0 tuning method**	**RTE**	**CB**
T0-3B zero-shot (median single prompt)	N/A	58.8	58.9
T0-3B + co-training	SPT	$84.8_{0.8}$	$64.6_{2.4}$
DeBERTa-large-mnli + co-training	SPT	$86.4_{0.7}$	$72.9_{1.3}$
T0-3B + co-training	T-Few	$86.1_{0.6}$	$78.9_{9.5}$
DeBERTa-large-mnli + co-training	T-Few	$87.1_{0.3}$	$79.3_{9.4}$

Table 3: Results for PECT (T-Few + co-training). These results use the full T0-11B model. We show standard deviations over five seeds for PECT. Each seed takes 5 hours on a single A100 80GB GPU. †The median performance for CB is 87.5 for both models; one seed had lower performance.

Model	**RTE**	**CB**
T0 zero-shot	81.2	78.6
PECT zero-shot (T0)	$87.1_{0.4}$	$^{\dagger}79.9_{14.2}$
PECT zero-shot (DeBERTa-large)	$88.5_{0.4}$	$^{\dagger}81.3_{10.0}$

the following datasets: RTE (Dagan et al., 2005), CB (De Marneffe et al., 2019), WiC (Pilehvar and Camacho-Collados, 2018), WSC (Levesque et al., 2012), BoolQ (Clark et al., 2019), COPA (Roemmele et al., 2011), HellaSwag (Zellers et al., 2019), and Winogrande (Sakaguchi et al., 2021). Following Lang et al. (2022a), we evaluate PECT on RTE and CB.

4.2 Results

Self-training. Table 1 shows the PEST results for zero-shot and few-shot learning. In the zero-shot case, the self-training iterations greatly improve over the initial T0-11B model (*T0 zero shot* versus *PEST zero-shot*), except for WiC, where the initial training signal is not strong enough to boost. Some zero-shot improvements are very large, such as the 21% (relative) improvement for HellaSwag and the 26% (relative) improvement for Winogrande. In some cases (e.g., RTE, COPA, Winogrande, BoolQ) PEST zero-shot is competitive with the *few-shot* performance of much larger models (GPT-3, FLAN). Self-training also improves over few-shot T-Few (*T-Few 32-shot* versus *PEST 32-shot*), though the gain is smaller than in the zero-shot case.

Co-training. Table 2 shows the performance of PECT versus co-training with soft prompt tuning (SPT) from Lang et al. (2022a). Switching SPT to T-Few improves the performance of co-training and decreases the wallclock time of a full training run from nearly 50 hours to 2 hours. Table 3 shows the performance of PECT for zero-shot cotraining T0-11B. The smaller number of steps required per iteration makes the algorithm practical for this model size, and the co-trained T0 model outperforms few-shot GPT-3, few-shot FLAN, and few-shot PEST on RTE.

5 Conclusion

We have shown that combining two semi-supervised learning approaches (co-training and self-training) with the T-Few fine-tuning method can improve the zero- and few-shot performance of large language models. The efficiency of T-Few compared to soft prompt tuning enables the scaling of these techniques to larger models (T0-11B) and dramatically cuts down on the wallclock time required for training. The simple, nonparametric data selection method and the relatively light computational burden (3–5 hours per run for T0-11B on a single A100 80GB) make these methods easy to implement and practical even for moderate computing budgets. Our empirical results show that PEST can improve the *zero*-shot performance of T0 by up to 26% (relative) and generally match the *few*-shot performance of GPT-3 and FLAN, which have more than 10x the parameters.

References

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. (2019). BoolQ: Exploring the surprising difficulty of natural yes/no questions. 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 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Dagan, I., Glickman, O., and Magnini, B. (2005). The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

De Marneffe, M.-C., Simons, M., and Tonhauser, J. (2019). The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23, pages 107–124.

Gao, T., Fisch, A., and Chen, D. (2021). Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830.

He, P., Liu, X., Gao, J., and Chen, W. (2021). Deberta: Decoding-enhanced bert with disentangled attention. In *2021 International Conference on Learning Representations*.

Lang, H., Agrawal, M., Kim, Y., and Sontag, D. (2022a). Co-training improves prompt-based learning for large language models. *arXiv preprint arXiv:2202.00828*.

Lang, H., Vijayaraghavan, A., and Sontag, D. (2022b). Training subset selection for weak supervision. *arXiv preprint arXiv:2206.02914*.

Lester, B., Al-Rfou, R., and Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Levesque, H., Davis, E., and Morgenstern, L. (2012). The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*.

Li, M. and Zhou, Z.-H. (2005). Setred: Self-training with editing. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 611–621. Springer.

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. (2022). Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *arXiv preprint arXiv:2205.05638*.

Muhlenbach, F., Lallich, S., and Zighed, D. A. (2004). Identifying and handling mislabelled instances. *Journal of Intelligent Information Systems*, 22(1):89–109.

Mukherjee, S. S. and Awadallah, A. H. (2020). Uncertainty-aware self-training for few-shot text classification. In *NeurIPS 2020 (Spotlight)*. ACM.

Pilehvar, M. T. and Camacho-Collados, J. (2018). Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. *arXiv preprint arXiv:1808.09121*.

Roemmele, M., Bejan, C. A., and Gordon, A. S. (2011). Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI spring symposium: logical formalizations of commonsense reasoning*, pages 90–95.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. (2021). Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Sanh, V., Webson, A., Raffel, C., Bach, S., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Le Scao, T., Raja, A., et al. (2022). Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations*.

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

Scudder, H. (1965). Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Wang, Y., Mukherjee, S., Chu, H., Tu, Y., Wu, M., Gao, J., and Awadallah, A. H. (2021). Meta self-training for few-shot neural sequence labeling. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1737–1747.

Wang, Y., Mukherjee, S., Liu, X., Gao, J., Awadallah, A., and Gao, J. (2022). LiST: Lite prompted self-training makes parameter-efficient few-shot learners. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2262–2281, Seattle, United States. Association for Computational Linguistics.

Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. (2021). Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196.

Ye, Z., Geng, Y., Chen, J., Chen, J., Xu, X., Zheng, S., Wang, F., Zhang, J., and Chen, H. (2020). Zero-shot text classification via reinforced self-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3014–3024.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. (2019). Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Zhang, M.-L. and Zhou, Z.-H. (2011). Cotrade: Confident co-training with data editing. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(6):1612–1626.

Zhao, Z., Wallace, E., Feng, S., Klein, D., and Singh, S. (2021). Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR.

A Cut Statistic

Let ϕ be a representation for input examples. For example, for text data, ϕ could be the hidden state of the `[CLS]` token in the last layer of a pretrained large language model, or the hidden state of the first token in the last decoder layer of an encoder-decoder model such as T0. We use both choices for ϕ in this work, as described in Section 3. The description below is essentially due to Lang et al. (2022b).

Let $\mathcal{T} = \{(x_i, \hat{Y}(x_i))\}$ be the pseudolabeled training set, where $\hat{Y}(x_i) = \text{argmax}\, f(x_i)$ is the hard pseudolabel assigned to example x_i by model f. To compute the cut statistic using ϕ, we first form a graph $G = (V, E)$ with one vertex for each x_i and edges connecting vertices that are M-nearest neighbors in ϕ (we use $M = 20$). That is, for each example x_i, let

$$\text{NN}_\phi(x_i) = \{x_j : (x_i, x_j) \text{ are } M\text{-nearest-neighbors in } \phi\}.$$

Then we set $V = \{i : x_i\}$, $E = \{(i, j) : x_i \in \text{NN}_\phi(x_j) \text{ or } x_j \in \text{NN}_\phi(x_i)\}$. For each node i, let $N(i) = \{j : (i, j) \in E\}$ denote its neighbors in G. We assign a weight w_{ij} to each edge so that nodes closer together in ϕ have a higher edge weight: $w_{ij} = (1 + ||\phi(x_i) - \phi(x_j)||_2)^{-1}$. We say an edge (i, j) is *cut* if $\hat{Y}(x_i) \neq \hat{Y}(x_j)$, and capture this with the indicator variable $I_{ij} := \mathbb{I}[\hat{Y}(x_i) \neq \hat{Y}(x_j)]$. If ϕ is a good representation, nodes with few incident cut edges should have high-quality pseudolabels—these examples have the same label as most of their neighbors. On the other hand, nodes with a large number of cut edges likely correspond to mislabeled examples. The cut statistic heuristically quantifies this idea to produce a ranking.

Suppose (as a null hypothesis) that the labels $\hat{Y}$ were sampled i.i.d. from the marginal distribution $\mathbb{P}[\hat{Y} = y]$. Large deviations from the null should represent the most noise-free vertices. For each vertex i, consider the test statistic: $J_i = \sum_{j \in N(i)} w_{ij} I_{ij}$. The mean of J_i under the null hypothesis is: $\mu_i = (1 - \mathbb{P}[\hat{Y}(x_i)]) \sum_{j \in N(i)} w_{ij}$, and the variance is: $\sigma_i^2 = \mathbb{P}[\hat{Y}(x_i)](1 - \mathbb{P}[\hat{Y}(x_i)]) \sum_{j \in N(j)} w_{ij}^2$. Then for each i we can compute the Z-score $Z_i = \frac{J_i - \mu_i}{\sigma_i}$ and rank examples by Z_i. Lower is better, since nodes with the smallest Z_i have the least noisy $\hat{Y}$ assignments in ϕ. We can choose the points with the smallest values of Z_i to use for confident training data in the next iteration. We can easily stratify this selection based on the pseudolabel to select a fixed number of confident examples per (pseudo)class.

In the supplementary material, we provide code for a simple (< 30 lines) function to compute the Z_i values given the representations $\{\phi(x_i) : x_i\}$. Since the cut statistic does not require soft pseudolabels, it can be used when the soft pseudolabels tend to be badly miscalibrated, which is the case with large language models like GPT-3 (Zhao et al., 2021).