Training Subset Selection for Weak Supervision

Hunter Lang
MIT CSAIL
hjl@mit.edu

Aravindan Vijayaraghavan
Northwestern University
aravindv@northwestern.edu

David Sontag
MIT CSAIL
dsontag@mit.edu

Abstract

Existing weak supervision approaches use all the data covered by weak signals to train a classifier. We show both theoretically and empirically that this is not always optimal. Intuitively, there is a tradeoff between the amount of weakly-labeled data and the precision of the weak labels. We explore this tradeoff by combining pretrained data representations with the *cut statistic* [23] to select (hopefully) high-quality subsets of the weakly-labeled training data. Subset selection applies to any label model and classifier and is very simple to plug in to existing weak supervision pipelines, requiring just a few lines of code. We show our subset selection method improves the performance of weak supervision for a wide range of label models, classifiers, and datasets. Using *less* weakly-labeled data improves the accuracy of weak supervision pipelines by up to 19% (absolute) on benchmark tasks.

1 Introduction

Due to the difficulty of hand-labeling large amounts of training data, an increasing share of models are trained with *weak supervision* [30, 28]. Weak supervision uses expert-defined "labeling functions" to programatically label a large amount of training data with minimal human effort. This *pseudo*-labeled training data is used to train a classifier (e.g., a deep neural network) as if it were hand-labeled data.

Labeling functions are often simple, coarse rules, so the pseudolabels derived from them are not always correct. There is an intuitive tradeoff between the *coverage* of the pseudolabels (how much pseudolabeled data do we use for training?) and the *precision* on the covered set (how accurate are the pseudolabels that we do use?). Using all the pseudolabeled training data ensures the best possible generalization to the population pseudolabeling function $\hat{Y}(X)$. On the other hand, if we can select a high-quality subset of the pseudolabeled data, then our training labels $\hat{Y}(X)$ are closer to the true label Y, but the smaller training set may hurt generalization. However, existing weak supervision approaches such as Snorkel [28], MeTaL [29], FlyingSquid [10], and Adversarial Label Learning [3] use *all* of the pseudolabeled data to train the classifier, and do not explore this tradeoff.

We present numerical experiments demonstrating that the status quo of using all the pseudolabeled data is nearly always suboptimal. Combining good pretrained representations with the *cut statistic* [23] for subset selection, we obtain subsets of the weakly-labeled training data where the weak labels are very accurate. By choosing examples with the same pseudolabel as many of their nearest neighbors in the representation, the cut statistic uses the representation's *geometry* to identify these accurate subsets without using any ground-truth labels. Using the smaller but higher-quality training sets selected by the cut statistic improves the accuracy of weak supervision pipelines by up to 19% accuracy (absolute). Subset selection applies to any "label model" (Snorkel, FlyingSquid, majority vote, etc.) and any classifier, since it is a modular, intermediate step between creation of the pseudolabeled training set and training. We conclude with a theoretical analysis of a special case of weak supervision where the precision/coverage tradeoff can be made precise.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

2 Background

The three components of a weak supervision pipeline are the *labeling functions*, the *label model*, and the *end model*. The labeling functions are maps $\Lambda_k : \mathcal{X} \to \mathcal{Y} \cup \{\varnothing\}$, where $\varnothing$ represents abstention. For example, for sentiment analysis, simple *token-based* labeling functions are effective, such as:

$$\Lambda_1(x) = \begin{cases} 1 & \text{“good”} \in x \\ \varnothing & \text{otherwise} \end{cases} \qquad \Lambda_2(x) = \begin{cases} -1 & \text{“bad”} \in x \\ \varnothing & \text{otherwise} \end{cases}$$

If the word “good” is in the input text x, labeling function Λ_1 outputs 1; likewise when “bad” $\in x$, Λ_2 outputs -1. Of course, an input text could contain both “good” and “bad”, so Λ_1 and Λ_2 may conflict. Resolving these conflicts is the role of the *label model*.

Formally, the label model is a map $\hat{Y} : (\mathcal{Y} \cup \{\varnothing\})^K \to \mathcal{Y} \cup \{\varnothing\}$. That is, if we let $\mathbf{\Lambda}(x)$ refer to the vector $(\Lambda_1(x), \ldots, \Lambda_K(x))$, then $\hat{Y}(\mathbf{\Lambda}(x))$ is a single pseudolabel (or “weak label”) derived from the vector of K labeling function outputs. This resolves conflicts between the labeling functions. Note that we can also consider $\hat{Y}$ as a deterministic function of X. The simplest label model is *majority vote*, which outputs the most common label from the set of non-abstaining labeling functions:

$$\hat{Y}_{MV}(x) = \text{mode}(\{\Lambda_k(x) : \Lambda_k(x) \neq \varnothing\})$$

If all the labeling functions abstain (i.e., $\Lambda_k(x) = \varnothing$ for all k), then $\hat{Y}_{MV}(x) = \varnothing$. More sophisticated label models such as Snorkel [30] and FlyingSquid [10] parameterize $\hat{Y}$ to learn better aggregation rules, e.g. by accounting for the accuracy of different Λ_k's or accounting for correlations between pairs (Λ_j, Λ_k). These parameters are learned using unlabeled data only; the methods for doing so have a rich history dating back at least to Dawid and Skene [8]. Many label models (including Snorkel and its derivatives) output a “soft” pseudolabel, i.e., a distribution $\hat{P}[Y|\Lambda_1(X), \ldots, \Lambda_K(X)]$, and set the hard pseudolabel as $\hat{Y}(X) = \text{argmax}_y \hat{P}[Y = y|\Lambda_1(X), \ldots, \Lambda_K(X)]$.

Given an unlabeled sample $\{x_i\}_{i=1}^n$, the label model produces a pseudolabeled training set $\mathcal{T} = \{(x_i, \hat{Y}(x_i)) : \hat{Y}(x_i) \neq \varnothing\}$. The final step in the weak supervision pipeline is to use $\mathcal{T}$ like regular training data to train an *end model* (such as a deep neural network), minimizing the zero-one loss:

$$\hat{f} := \underset{f \in \mathcal{F}}{\text{argmin}} \frac{1}{|\mathcal{T}|} \sum_{i=1}^{|\mathcal{T}|} \mathbb{I}[f(x_i) \neq \hat{Y}(x_i)] \tag{1}$$

or a convex surrogate like cross-entropy. For many applications, we fine-tune a *pretrained* representation instead of training from scratch. For example, on text data, we can fine-tune a pretrained BERT model. We refer to the pretrained representation used by the end model as the *end model representation*, where applicable.

Notably, all existing methods use the full pseudolabeled training set $\mathcal{T}$ to train the end model. $\mathcal{T}$ consists of *all* points where $\hat{Y} \neq \varnothing$. In this work, we experiment with methods for choosing higher-quality subsets $\mathcal{T}' \subset \mathcal{T}$ and use $\mathcal{T}'$ in (1) instead of $\mathcal{T}$.

Related Work. The idea of selecting a subset of high-quality training data for use in fully-supervised or semi-supervised learning algorithms has a long history. It is also referred to as *data pruning* [1], and a significant amount of work has focused on removing mislabeled examples to improve the training process [e.g., 26, 18, 7, 25]. These works do not consider the case where the pseudolabels come from deterministic labeling functions, and most try to estimate parameters of a specific noise process that is assumed to generate the pseudolabels. Many of these approaches require iterative learning or changes to the loss function, whereas typical weak supervision pipelines do one learning step and little or no loss correction. Maheshwari et al. [21] study *active* subset selection for weak supervision, obtaining a small number of human labels to boost performance.

In *self-training* [e.g., 34], an initial labeled training set is iteratively supplemented with the pseudolabeled examples where a trained model is most confident (according to the model's probability scores). The model is retrained on the new training set in each step. Yarowsky [40] used this approach starting from a *weakly*-labeled training set; Yu et al. [41], Karamanolakis et al. [15] also combine self-training with an initial weakly-labeled training set, and both have deep-model-based procedures for selecting confident data in each round. We view these weakly-supervised self-training methods as orthogonal to our approach, since their *main* focus is on making better use of the data that is not covered by weak rules, not on selecting good pseudolabeled subsets. Indeed, we show in Appendix

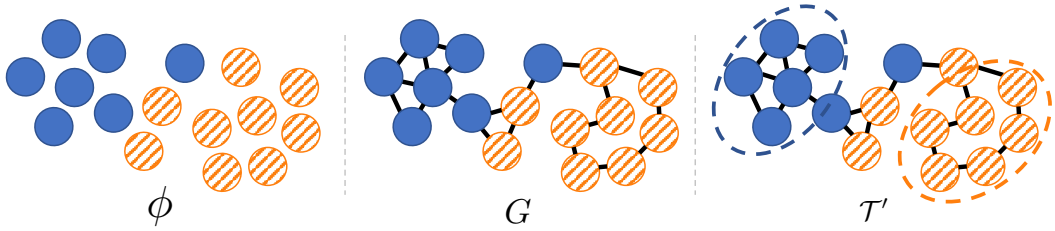


Figure 1: Cut statistic procedure. A representation ϕ is used to compute the nearest-neighbor graph G. Nodes that have the same pseudolabel as most of their neighbors are chosen for the subset $\mathcal{T}'$.

B.7 that combining our method with these approaches improves their performance. Other selection schemes, not based on model confidence, have also been investigated for self-training [e.g., 44, 24].

Muhlenbach et al. [23] introduced the *cut statistic* as a heuristic for identifying mislabeled examples in a training dataset. Li and Zhou [19] applied the cut statistic to self-training, using it to select high-quality pseudolabeled training data for each round. Zhang and Zhou [43] applied the cut statistic to co-training and used learning-with-noise results from Angluin and Laird [2] to optimize the amount of selected data in each round. Lang et al. [16] also used co-training and the cut statistic to co-train large language models such as GPT-3 [5] and T0 [31] with smaller models such as BERT [9] and RoBERTa [20]. These previous works showed that the cut statistic performs well in *iterative* algorithms such as self-training and co-training; we show that it works well in *one-step* weak supervision settings, and that it performs especially well when combined with modern pre-trained representations. Our empirical study shows that this combination is very effective at selecting good pseudolabeled training data *across a wide variety of label models, end models, and datasets*.

As detailed in Section 3, the performance of the cut statistic relies on a good representation of the input examples x_i to find good subsets. Zhu et al. [45] also used representations to identify subsets of mislabeled labels and found that methods based on representations outperform methods based on model predictions alone. They use a different ranking method and do not evaluate in weakly-supervised settings. Chen et al. [6] also use pretrained representations to improve the performance of weak supervision. They created a new representation-aware label model that uses nearest-neighbors in the representation to label *more* data and also learns finer-grained label model parameters. In contrast, our approach applies to any label model, can be implemented in a few lines of code, and does not require representations from very large models like GPT-3 or CLIP. Combining the two approaches is an interesting direction for future work.

3 Subset Selection Methods for Weak Supervision

In this work, we study techniques for selecting high-quality subsets of the pseudolabeled training set $\mathcal{T}$. We consider two simple approaches to subset selection in this work: *entropy scoring* and the *cut statistic*. In both cases, we construct a subset $\mathcal{T}'$ by first ranking all the examples in $\mathcal{T}$, then selecting the top β fraction according to the ranking. In our applications, $0 < \beta \leqslant 1$ is a hyperparameter tuned using a validation set. Hence, instead of $|\mathcal{T}|$ covered examples for training the end model in (1), we use $\beta|\mathcal{T}|$ examples. Instead of a single, global ranking, subset selection can easily be stratified to use multiple rankings. For example, if the true label balance $\mathbb{P}[Y]$ is known, we can use separate rankings for each set $\mathcal{T}_y = \{x_i : \hat{Y}(x_i) = y\}$ and select the top $\beta\mathbb{P}[Y = y]|\mathcal{T}|$ points from each $\mathcal{T}_y$. This matches the pseudolabel distribution on $\mathcal{T}'$ to the true marginal $\mathbb{P}[Y]$. For simplicity, we use a global ranking in this work, and our subset selection does not use $\mathbb{P}[Y]$ or any other information about the true labels. Below we give details for the entropy and cut statistic rankings.

Entropy score. Entropy scoring only applies to label models that output a "soft" pseudo-label $\hat{P}[Y|\mathbf{\Lambda}(X)]$. For this selection method, we rank examples by the Shannon entropy of the soft label, $H(\hat{P}[Y|\mathbf{\Lambda}(x_i)])$, and set $\mathcal{T}'$ to the $\beta|\mathcal{T}|$ examples with the lowest entropy. Intuitively, the label model is the "most confident" on the examples with the lowest entropy. If the label model is well-calibrated, the weak labels should be more accurate on these examples.

Cut statistic [23]. Unlike the entropy score, which only relies on the soft label distribution $\hat{P}[Y|\mathbf{\Lambda}]$, the cut statistic relies on a good *representation* of the input examples x_i. Let ϕ be a representation for

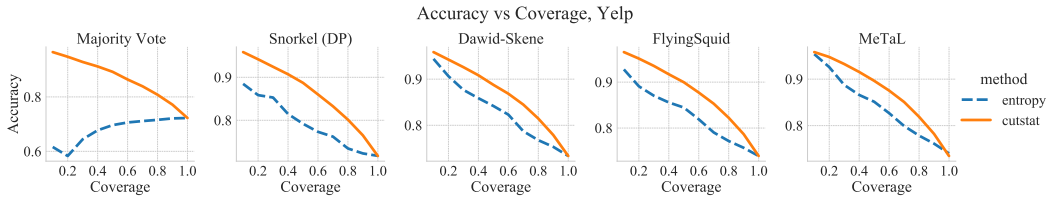


Figure 2: Accuracy of the pseudolabeled training set versus the selection fraction β for five different label models. A pretrained BERT model is used as ϕ for the cut statistic. The accuracy of the weak training labels is better for $\beta < 1$, indicating that sub-selection can select higher-quality training sets.

examples in $\mathcal{X}$. 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.

Recall that $\mathcal{T} = \{(x_i, \hat{Y}(x_i)) : \hat{Y}(x_i) \neq \varnothing\}$. To compute the cut statistic using ϕ, we first form a graph $G = (V, E)$ with one vertex for each covered x_i and edges connecting vertices who are K-nearest neighbors in ϕ. That is, for each example x_i with $\hat{Y}(x_i) \neq \varnothing$, let

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

Then we set $V = \{i : \hat{Y}(x_i) \neq \varnothing\}$, $E = \{(i,j) : x_i \in \mathrm{NN}_\phi(x_j) \text{ or } x_j \in \mathrm{NN}_\phi(x_i)\}$. 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 $I_{ij} := \mathbb{I}[\hat{Y}(x_i) \neq \hat{Y}(x_j)]$. As suggested in Figure 1, 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. As with entropy scoring, we set $\mathcal{T}'$ to be the the $\beta|T|$ points with the smallest values of Z_i. We provide code for a simple ($<$ 30 lines) function to compute the Z_i values given the representations $\{\phi(x_i) : x_i\}$ in Appendix C. Calling this function makes it very straightforward to incorporate the cut statistic in existing weak supervision pipelines. Since the cut statistic does not require soft pseudolabels, it can also be used for label models that only produce hard labels, and for label models such as Majority Vote, where the soft label tends to be badly miscalibrated.

3.1 Cut Statistic Selects Better Subsets

To explore the two scoring methods, we visualize how $\mathcal{T}'$ changes with β for entropy scoring and the cut statistic. We used label models such as majority vote and Snorkel [30] to obtain soft labels $\hat{P}[Y|\mathbf{\Lambda}(x_i)]$, and set $\hat{Y}(x_i)$ to be the argmax of the soft label. We test using the **Yelp** dataset from the WRENCH weak supervision benchmark [42]. The task is sentiment analysis, and the eight labeling functions $\{\Lambda_1, \ldots, \Lambda_8\}$ consist of seven keyword-based rules and one third-party sentiment polarity model. For ϕ in the cut statistic, we used the `[CLS]` token representation of a pretrained BERT model. Section 4 contains more details on the datasets and the cut statistic setup.

For each $\beta \in \{0.1, 0.2, \ldots, 1.0\}$, Figure 2 plots the accuracy of the pseudolabels on the training subset $\mathcal{T}'(\beta)$. This shows how training subset quality varies with the selection fraction β. We can compute this accuracy because most of the WRENCH benchmark datasets also come with ground-truth labels Y (even on the training set) for evaluation. Appendix B contains the same plot for several other WRENCH datasets and figures showing the histograms of the entropy scores and the Z_i values.

Figure 2 shows that combining the cut statistic with a BERT representation selects better subsets than the entropy score for all five label models tested, especially for majority vote, where the entropy

scoring is badly miscalibrated. For a well-calibrated score, the subset accuracy should decrease as β increases. These results suggest that the cut statistic is able to use the geometric information encoded in ϕ to select a more accurate subset of the weakly-labeled training data. However, it does *not* indicate whether that better subset actually leads to a more accurate end model. Since we could also use ϕ for the end model—e.g., by fine-tuning the full neural network or training a linear model on top of ϕ—it's possible that the training step (1) will already perform the same corrections as the cut statistic, and the end model trained on the selected subset will perform no differently from the end model trained with $\beta = 1.0$. In the following section, we focus on the cut statistic and conduct large-scale empirical evaluation on the WRENCH benchmark to measure whether subset selection improves end model performance. Our empirical results suggest that subset selection and the end model training step are complementary: even when we use powerful representations for the end model, subset selection further improves performance, sometimes by a large margin.

4 Experiments

Having established that the cut statistic can effectively select weakly-labeled training *sub*sets that are higher-quality than the original training set, we now turn to a wider empirical study to see whether this approach actually improves the performance of end models in practice.

Datasets and Models. We evaluate our approach on the WRENCH benchmark [42] for weak supervision. We compare the status-quo of full coverage ($\beta = 1.0$) to β chosen from $\{0.1, 0.2, \ldots, 1.0\}$. We evaluate our approach with five different label models: Majority Vote (**MV**), the original Snorkel/Data Programming (**DP**), [30], Dawid-Skene (**DS**) [8], FlyingSquid (**FS**) [10], and **MeTaL** [29]. Following Zhang et al. [42], we use pretrained `roberta-base` and `bert-base-cased`[1] as the end model representation for text data, and hand-specified representations for tabular data. We performed all model training on NVIDIA A100 GPUs. We primarily evaluate on seven textual datasets from the WRENCH benchmark: **IMDb** (sentiment analysis), **Yelp** (sentiment analysis), **Youtube** (spam classification), **TREC** (question classification), **SemEval** (relation extraction), **ChemProt** (relation extraction), and **AGNews** (text classification). Full details for the datasets and the weak label sources are available in [42] Table 5 and reproduced here in Appendix B.1. We explore other several other datasets and data modalities in Sections 4.2-4.3.

Cut statistic. For the representation ϕ, for text datasets we used the `[CLS]` token representation of a large pretrained model such as BERT or RoBERTa. For relation extraction tasks, we followed [42] and used the concatenation of the `[CLS]` token and the average contextual representation of the tokens in each entity span. In Section 4.3, for the tabular **Census** dataset we use the raw data features for ϕ. Unless otherwise specified, we used *the same representation* for ϕ and for the initial end model. For example, when training `bert-base-cased` as the end model, we used `bert-base-cased` as ϕ for the cut statistic. We explore several alternatives to this choice in Section 4.2.

Hyperparameter tuning. Our subset selection approach introduces a new hyperparameter, β—the fraction of covered data to retain for training the classifier. To keep the hyperparameter tuning burden low, we first tune all other hyperparameters identically to Zhang et al. [42] holding β fixed at 1.0. We then use the optimal hyperparameters (learning rate, batch size, weight decay, etc.) from $\beta = 1.0$ for a grid search over values of $\beta \in \{0.1, 0.2, \ldots, 1.0\}$, choosing the value with the best (ground-truth) validation performance. Better results could be achieved by tuning all the hyperparameters together, but this approach limits the number of possible combinations, and it matches the setting where an existing, tuned weak supervision pipeline (with $\beta = 1.0$) is adapted to use subset selection. In all of our experiments, we used $K = 20$ nearest neighbors to compute the cut statistic and performed no tuning on this value. Appendix B contains an ablation showing that performance is not sensitive to this choice.

4.1 WRENCH Benchmark Performance

Table 1 compares the test performance of full coverage ($\beta = 1.0$) to the performance of the cut statistic with β chosen according to validation performance. Standard deviations across five random initializations are shown in parentheses.

[1] We refer to pretrained models by their names on the HuggingFace Datasets Hub. All model weights were downloaded from the hub: https://huggingface.co/datasets

Table 1: End model test accuracy (stddev) for weak supervision with $\beta = 1$ versus weak supervision with β selected from $\{0.1, 0.2, 0.3, \ldots, 1.0\}$ using a validation set ("+ cutstat"), shown for BERT (*B*) and RoBERTa (*RB*) end models. For these results, the cut statistic uses the same representation as the end model for ϕ. The cut statistic broadly improves the performance of weak supervision for many (label model, dataset, end model) combinations.

	Label model	**imdb**	**yelp**	**youtube**	**trec**	**semeval**	**chemprot**	**agnews**
B	Majority Vote	$78.32_{2.62}$	$86.85_{1.42}$	$95.12_{1.27}$	$66.76_{1.46}$	$85.17_{0.89}$	$57.44_{2.01}$	$86.59_{0.47}$
	+ *cutstat*	$\mathbf{81.86}_{1.36}$	$\mathbf{89.49}_{0.78}$	$\mathbf{95.60}_{0.72}$	$\mathbf{71.84}_{3.00}$	$\mathbf{92.47}_{0.49}$	$\mathbf{57.47}_{1.00}$	$86.26_{0.43}$
	Data Programming	$75.90_{1.44}$	$76.43_{1.29}$	$92.48_{1.30}$	$71.20_{1.78}$	$71.97_{1.57}$	$51.89_{1.60}$	$86.01_{0.63}$
	+ *cutstat*	$\mathbf{79.07}_{2.52}$	$\mathbf{88.13}_{1.46}$	$\mathbf{93.92}_{0.93}$	$\mathbf{76.76}_{1.92}$	$\mathbf{91.07}_{0.90}$	$\mathbf{55.10}_{1.49}$	$85.89_{0.45}$
	Dawid-Skene	$78.86_{1.34}$	$88.45_{1.42}$	$88.45_{1.42}$	$51.04_{1.71}$	$72.40_{1.53}$	$44.08_{1.37}$	$86.26_{0.56}$
	+ *cutstat*	$\mathbf{80.22}_{1.69}$	$\mathbf{89.04}_{1.10}$	$\mathbf{90.72}_{1.27}$	$\mathbf{57.28}_{2.91}$	$\mathbf{89.07}_{1.62}$	$\mathbf{49.07}_{1.48}$	$\mathbf{86.93}_{0.22}$
	FlyingSquid	$77.46_{1.88}$	$84.98_{1.44}$	$91.52_{2.90}$	$31.12_{2.39}$	$31.83_{0.00}$	$46.72_{0.96}$	$86.10_{0.80}$
	+ *cutstat*	$\mathbf{80.85}_{1.50}$	$\mathbf{88.75}_{1.13}$	$91.04_{1.23}$	$\mathbf{33.84}_{3.17}$	$31.83_{0.00}$	$\mathbf{48.65}_{0.99}$	$85.90_{0.39}$
	MeTaL	$78.97_{2.57}$	$83.05_{1.69}$	$93.36_{1.15}$	$58.88_{1.22}$	$58.17_{1.77}$	$55.61_{1.35}$	$86.06_{0.82}$
	+ *cutstat*	$\mathbf{81.49}_{1.51}$	$\mathbf{88.41}_{1.19}$	$92.64_{0.41}$	$\mathbf{63.80}_{2.28}$	$\mathbf{65.23}_{0.91}$	$\mathbf{58.33}_{0.81}$	$\mathbf{86.16}_{0.48}$
RB	Majority Vote	$86.99_{0.55}$	$88.51_{3.25}$	$95.84_{1.18}$	$67.60_{2.38}$	$85.83_{1.22}$	$57.06_{1.12}$	$87.46_{0.53}$
	+ *cutstat*	$86.69_{0.75}$	$\mathbf{95.19}_{0.23}$	$\mathbf{96.00}_{1.10}$	$\mathbf{72.92}_{1.31}$	$\mathbf{92.07}_{0.80}$	$\mathbf{59.05}_{0.56}$	$\mathbf{88.01}_{0.47}$
	Data Programming	$86.31_{1.53}$	$88.73_{5.07}$	$94.08_{1.48}$	$71.40_{3.30}$	$71.07_{1.66}$	$52.52_{0.69}$	$86.75_{0.24}$
	+ *cutstat*	$\mathbf{86.46}_{1.82}$	$\mathbf{93.95}_{0.93}$	$93.04_{1.30}$	$\mathbf{76.84}_{4.09}$	$\mathbf{86.07}_{1.82}$	$\mathbf{56.43}_{1.37}$	$\mathbf{87.76}_{0.17}$
	Dawid-Skene	$85.50_{1.68}$	$92.42_{1.41}$	$92.48_{1.44}$	$51.24_{3.50}$	$70.83_{0.75}$	$45.61_{2.60}$	$87.29_{0.40}$
	+ *cutstat*	$\mathbf{86.14}_{0.60}$	$\mathbf{93.81}_{0.69}$	$\mathbf{93.84}_{0.70}$	$\mathbf{58.48}_{2.75}$	$\mathbf{81.67}_{1.33}$	$\mathbf{52.93}_{1.67}$	$\mathbf{88.35}_{0.22}$
	FlyingSquid	$85.25_{1.96}$	$92.14_{2.76}$	$93.52_{2.11}$	$35.40_{1.32}$	$31.83_{0.00}$	$47.23_{1.04}$	$86.56_{0.55}$
	+ *cutstat*	$\mathbf{87.71}_{0.76}$	$\mathbf{94.50}_{0.74}$	$\mathbf{95.84}_{0.54}$	$\mathbf{38.16}_{0.43}$	$31.83_{0.00}$	$\mathbf{50.55}_{1.05}$	$\mathbf{87.49}_{0.13}$
	MeTaL	$86.16_{1.13}$	$88.41_{3.25}$	$92.40_{1.19}$	$55.44_{1.08}$	$59.53_{1.87}$	$56.74_{0.58}$	$86.74_{0.60}$
	+ *cutstat*	$\mathbf{87.46}_{0.65}$	$\mathbf{94.03}_{0.53}$	$\mathbf{93.84}_{1.38}$	$\mathbf{69.72}_{2.39}$	$\mathbf{66.70}_{0.90}$	$\mathbf{57.40}_{0.98}$	$\mathbf{88.40}_{0.38}$

The cut statistic improves the mean performance (across runs) compared to $\beta = 1.0$ in 61/70 cases, sometimes by 10–20 accuracy points (e.g., BERT SemEval DP). Since $\beta = 1.0$ is included in the hyperparameter search over β, the only cases where the cut statistic performs worse than $\beta = 1.0$ are due to differences in the performance on the validation and test sets. The mean accuracy gain from setting $\beta < 1.0$ across all 70 trials is 3.65 points, indicating that the cut statistic is complementary to the end model training. If no validation data is available to select β, we found that $\beta = 0.6$ had the best median performance gain over all label model, dataset, and end model combinations: $+1.7$ accuracy points compared to $\beta = 1.0$. However, we show in Section 4.4 that very small validation sets are good enough to select β. The end model trains using ϕ, but using ϕ to *first* select a good training set further improves performance. Figure 3 displays a box plot of the accuracy gain from using sub-selection. Appendix B.2 contains plots of the end model performance versus the coverage fraction β. In some cases, the cut statistic is competitive with COSINE [41] , which does multiple rounds of self-training on the unlabeled data. Table 7 compares the two methods, and we show in Appendix B.7 how to combine them to improve performance.

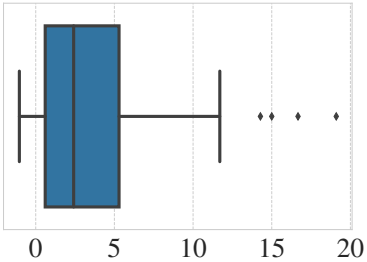


Figure 3: Test accuracy gain from setting $\beta < 1$ across all WRENCH trials.

For Table 1, we used the same representation for ϕ and for the initial end model. These results indicate that representations from very large models such as GPT-3 or CLIP are not needed to improve end model performance. However, there is no *a priori* reason to use the same representation for ϕ and for the end model initialization. Using a much larger model for ϕ may improve the cut statistic performance without drastically slowing down training, since we only need to perform *inference* on the larger model. We examine the role of the representation choice more thoroughly in Section 4.2.

4.2 Choice of Representation for Cut Statistic

How important is the quality of ϕ (the representation used for the cut statistic) for the performance of subset selection? In this section we experiment with different choices of ϕ. To isolate the effect of ϕ

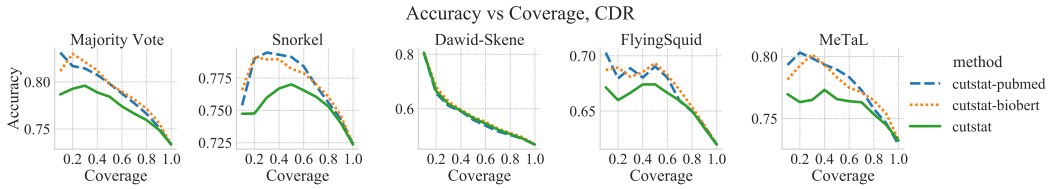


Figure 4: Domain-specific pretraining versus general-domain pretraining for ϕ. Stock BERT (*cutstat*) compared to BioBERT (*cutstat-biobert*), and PubMedBERT (*cutstat-pubmed*), two models pretrained on biomedical text. The domain-specific models select more accurate subsets than the generic model.

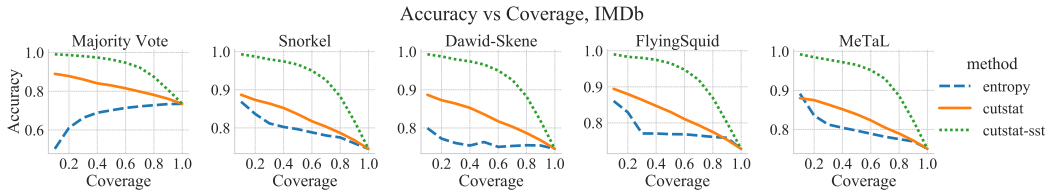


Figure 5: Comparison of IMDb training subset accuracy for the cut statistic with generic BERT (*cutstat*) and a BERT fine-tuned for sentiment analysis on the SST-2 dataset (*cutstat-sst*). The fine-tuned representation gives very high-quality training subsets when used with the cut statistic.

on performance, we use the generic BERT as the end model. Performance can improve further when using more powerful representations for the end model as well, but we use a fixed end model here to explore how the choice of ϕ affects final performance. Our results indicate that (i) for weak supervision tasks in a specific domain (e.g., biomedical text), models pretrained on that domain perform better than general-domain models as ϕ and (ii) for a specific task (e.g., sentiment analysis), models pretrained on that task, but on different data, perform very well as ϕ. We also show in Appendix B that using a larger generic model for ϕ can improve performance when the end model is held fixed.

Domain-specific pretraining can help. The **CDR** dataset in WRENCH has biomedical abstracts as input data. Instead of using general-domain ϕ such as BERT and RoBERTa, does using a domain-specific version improve performance? Figure 4 shows that domain specific models *do* improve over general-domain models when used in the cut statistic. We compare `bert-base-cased` to `PubMedBERT-base-uncased-abstract-fulltext` [11] and `biobert-base-cased-v1.2` [17]. The latter two models were pretrained on biomedical text. The domain-specific models lead to higher quality training datasets for all label models except Dawid-Skene. These gains in training dataset accuracy translate to gains in end-model performance. Trained with $\hat{Y}$ from majority vote and using BioBERT as ϕ, a general-domain BERT end model obtains test F1 score 61.14 (0.64), compared to 59.63 (0.84) using BERT for both ϕ and the end model. Both methods improve over 58.20 (0.55) obtained from training generic BERT with $\beta = 1.0$ (no sub-selection).

Representations can transfer. If a model is trained for a particular task (e..g, sentiment analysis) on one dataset, can we use it as ϕ to perform weakly-supervised learning on a different dataset? We compare two choices for ϕ on IMDb: regular `bert-base-cased`, and `bert-base-cased` fine-tuned with fully-supervised learning on the Stanford Sentiment Treebank (SST) dataset [37]. As indicated in Figure 5, the fine-tuned BERT representation selects a far higher-quality subset for training. This translates to better end model performance as well. Using majority vote with the fine-tuned BERT as ϕ leads to test performance of 87.22 (0.57), compared to 81.86 in Table 1. These results suggest that if we have a representation ϕ that's already useful for a task, we can effectively combine it with the cut statistic to improve performance on a different dataset.

4.3 Other Data Modalities

Our experiments so far have used text data, where large pretrained models like BERT and RoBERTa are natural choices for ϕ. Here we briefly study the cut statistic on *tabular* and *image* data. The **Census** dataset in WRENCH consists of tabular data where the goal is to classify whether a person's income is greater than \$50k from a set of 13 features. We also use these hand-crafted features for ϕ and train a linear model on top of the features for the end model. The **Basketball** dataset is a set of still images obtained from videos, and the goal is to classify whether basketball is being

Table 2: Test F1 of a weakly-supervised linear model (*LR*) on the **Census** dataset, which consists of 13 hand-created features. Even though the representation does not come from a large, pretrained neural network, the cut statistic improves the performance of weak supervision for every label model. Test F1 of a 1-hidden-layer network (*MLP*) on CLIP representations of the **Basketball** dataset, where the results are noisy, but the cut statistic improves the mean performance of every label model.

	Dataset	**Majority Vote**	**Data Programming**	**Dawid-Skene**	**FlyingSquid**	**MeTaL**
LR	Census	$50.71_{2.18}$	$21.67_{17.32}$	$49.90_{0.61}$	$38.23_{3.96}$	$51.41_{1.45}$
	+ *cutstat*	$\mathbf{57.98}_{0.68}$	$\mathbf{28.04}_{17.38}$	$\mathbf{58.49}_{0.23}$	$\mathbf{40.53}_{2.26}$	$\mathbf{54.99}_{1.54}$
MLP	Basketball	$52.29_{6.62}$	$52.14_{4.80}$	$22.59_{9.83}$	$54.04_{12.15}$	$32.99_{12.98}$
	+ *cutstat*	$\mathbf{55.82}_{3.98}$	$\mathbf{54.59}_{14.80}$	$\mathbf{43.77}_{12.47}$	$\mathbf{56.60}_{6.13}$	$\mathbf{47.97}_{8.89}$

Table 3: Comparison between using the full validation set to choose β and the model checkpoint versus using a randomly selected validation subset of 100 examples. These results use the majority vote (MV) label model. Standard deviation is reported over five random seeds used to select the validation set (not to be confused with Table 1, where standard deviation is reported over random seeds controlling the deep model initialization). Most of the drop in performance is due to the noisier checkpoint selection when using the small validation set. I.e., the difference between $\beta =$ best and $\beta = 1.0$ is similar for the full validation and random validation cases.

	Val. size	β	**imdb**	**yelp**	**youtube**	**trec**	**semeval**	**chemprot**	**agnews**
BERT	full	1.0	78.32	86.85	95.12	66.76	85.17	57.44	86.59
	full	best	81.86	89.49	95.60	71.84	92.47	57.47	86.26
	100	1.0	$79.17_{2.80}$	$84.88_{1.97}$	$94.50_{0.32}$	$65.33_{1.84}$	$85.62_{0.64}$	$54.99_{1.63}$	$84.77_{1.11}$
	100	best	$79.75_{2.18}$	$87.96_{1.00}$	$94.40_{0.63}$	$74.50_{1.83}$	$92.40_{1.81}$	$54.40_{2.35}$	$84.96_{1.20}$
RoBERTa	full	1.0	86.99	88.51	95.84	67.60	85.83	57.06	87.46
	full	best	86.69	95.19	96.00	72.92	92.07	59.05	88.01
	100	1.0	$85.74_{1.11}$	$89.32_{1.49}$	$95.24_{1.11}$	$66.40_{1.56}$	$84.38_{1.18}$	$56.71_{0.55}$	$85.79_{0.75}$
	100	best	$85.24_{0.78}$	$93.82_{0.48}$	$96.06_{0.70}$	$75.15_{2.89}$	$91.24_{0.76}$	$56.52_{0.44}$	$87.20_{0.31}$

played in the image using the output of an off-the-shelf object detector in the Λ_k's. We used CLIP [27] representations of the video frames and trained a 1-hidden-layer neural network using the hyperparameter tuning space from [42]. Table 2 shows the results for these datasets. The cut statistic improves the end model performance for every label model even with the small, hand-crafted representation, and also improves for the **Basketball** data.

4.4 Using a Smaller Validation Set

Many datasets used to evaluate weak supervision methods actually come with large labeled validation sets. For example, the average validation set size of the WRENCH datasets from Table 1 is over 2,500 examples. However, assuming access to a large amount of labeled validation data partially defeats the purpose of weak supervision. In this section, we show that the coverage parameter β for the cut statistic can be selected using a much smaller validation set without compromising the performance gain over $\beta = 1.0$. We compare choosing the best model checkpoint and picking the best coverage fraction β using (i) the *full* validation set and (ii) a randomly-sampled validation set of 100 examples. Table 3 shows the results for the majority vote label model. The full validation numbers come from Table 1. The difference between selecting data with the validation-optimal β and using $\beta = 1.0$ is broadly similar between the full validation and small validation cases. This suggests that most of the drop in performance from full validation to small validation is due to the noisier choice of the best model checkpoint, not due to choosing a suboptimal β.

4.5 Discussion

Why not *correct* pseudolabels with nearest neighbor? Consider an example x_i whose weak label $\hat{Y}(x_i)$ disagrees with the weak label $\hat{Y}(x_j)$ of most neighbors $j \in N(i)$. This example would get thrown out by the cut statistic selection. Instead of throwing such data points out, we could try to *re-label* them with the majority weak label from the neighbors. However, throwing data out is a more conservative (and hence possibly more robust) approach. For example, if the weak labels are

mostly *wrong* on hard examples close to the true unknown decision boundary, relabeling makes the training set worse, whereas the cut statistic ignores these points. Appendix B.3 contains an empirical comparison between subset selection and relabeling. For the representations studied in this work, relabeling largely fails to improve training set quality and end model performance.

Why does sub-selection work? As suggested above, subset selection can change the distribution of data points in the training set shown to the end model. For example, it may only select "easy" examples. However, this is already a problem in today's weak supervision methods: the full weakly-labeled training set $\mathcal{T}$ is already biased. For example, many labeling functions are keyword-based, such as those in Section 2 ("good"→positive sentiment, "bad"→negative). In these examples, $\mathcal{T}$ itself is a biased subset of the input distribution (only sentences that contain "good" or "bad", versus all sentences). Theoretical understanding for why weak supervision methods perform well on the uncovered set $\mathcal{X}\backslash\{x : \mathbf{\Lambda}(x) \neq \varnothing\}$ is currently lacking, and existing generalization bounds for the end model do not capture this phenomenon. In the following section we present a special (but practically-motivated) case where this bias can be avoided. In this case, we prove a closed form for the coverage-precision tradeoff of selection methods, giving subset selection some theoretical motivation.

5 Theoretical Results: Why Does Subset Selection Work?

We begin by presenting a theoretical setup motivated by the CheXPert [13] and MIMIC-CXR [14] datasets, where the weak labels are derived from radiology notes and the goal is to learn an end model for classifying X-ray images. Suppose for this section that we have two (possibly related) *views* $\psi_0(X)$, $\psi_1(X)$ of the data X, i.e., $\psi_0 : \mathcal{X} \to \Psi_0$, $\psi_1 : \mathcal{X} \to \Psi_1$. We use ψ here to distinguish from ϕ, the representation used to compute nearest neighbors for the cut statistic. For example, if the input space $\mathcal{X}$ is multi-modal, and each $x_i = (x_i^{(0)}, x_i^{(1)})$, then we can set ψ_0 and ψ_1 to project onto the individual modes (e.g., $\phi_0(X)$ the clinical note and $\phi_1(X)$ the X-ray). We will assume that the labeling functions $\Lambda_k(x_i)$ only depend on $\psi_0(x_i)$, and that the end model f only depends on $\psi_1(x_i)$. In the multi-modal example, this means the labeling functions are defined on one view, and the end model is trained on the other view. To prove a closed form for the precision/coverage tradeoff, we make the following strong assumption relating the two views ψ_0 and ψ_1:

Assumption 1 (Conditional independence)**.** *The random variables $\psi_0(X)$, $\psi_1(X)$ are conditionally independent given the true (unobserved) label Y. That is, for any sets $A \subset \Psi_0$, $B \subset \Psi_1$,*

$$\mathbb{P}_{X,Y}[\psi_0(X) \in A, \psi_1(X) \in B|Y] = \mathbb{P}_{X,Y}[\psi_0(X) \in A|Y]\mathbb{P}_{X,Y}[\psi_1(X) \in B|Y].$$

Note since every Λ_k only depends on $\psi_0(X)$, the pseudolabel $\hat{Y}$ only depends on $\psi_0(X)$. Hence $\hat{Y}(X) = \hat{Y}(\psi_0(X))$, and likewise for an end model f, $f(X) = f(\psi_1(X))$. Assumption 1 implies:

$$\mathbb{P}_{X,Y}[\mathbb{I}[\hat{Y}(X) \neq Y], \psi_1(X) \in B|Y] = \mathbb{P}_{X,Y}[\mathbb{I}[\hat{Y}(X) \neq Y]|Y]\mathbb{P}_{X,Y}[\psi_1(X) \in B|Y]$$

for every $B \subset \Psi_1$. In this special case, the end model training reduces to learning with class-conditional noise (CCN), since the errors $\mathbb{I}[\hat{Y}(X) \neq Y]$ are conditionally independent of the representation $\psi_1(X)$ being used for the end model. This assumption is most natural for the case of multi-modal data and ψ_0, ψ_1 that project onto each mode, but it may also roughly apply when the representation being used for the end model (such as a BERT representation) is "suitably orthogonal" to the input X. While very restrictive, this assumption allows us to make the coverage-precision tradeoff precise.

Theorem 1. *Suppose Assumption 1 holds, and that $\mathcal{Y} = \{0, 1\}$. Define the* balanced error *of a classifier f on labels Y as: $err_{bal}(f, Y) = \frac{1}{2}(\mathbb{P}[f(X) = 0|Y = 1] + \mathbb{P}[f(X) = 1|Y = 0])$. We write $f(X)$ instead of $f(\psi_1(X))$ for convenience. Let $\hat{Y} : \Psi_0(X) \to \{0, 1\}$ be an arbitrary label model. Define $\alpha = \mathbb{P}[Y = 0|\hat{Y} = 1]$ and $\gamma = \mathbb{P}[Y = 1|\hat{Y} = 0]$ and suppose $\alpha + \gamma < 1$, $\mathbb{P}[\hat{Y} = y] > 0$ for $y \in \{0, 1\}$. These parameters measure the amount of noise in $\hat{Y}$. Define $f^* := \inf_{f \in \mathcal{F}} err_{bal}(f, Y)$. Let $\hat{f}$ be the classifier obtained by minimizing the empirical balanced accuracy on $\{(x_i, \hat{Y}(x_i))\}_{i=1}^n$. Then the following holds w.p. $1 - \delta$ over the sampling of the data:*

$$err_{bal}(\hat{f}, Y) - err_{bal}(f^*, Y) \leqslant \tilde{\mathcal{O}}\left(\frac{1}{1 - \alpha - \gamma}\sqrt{\frac{\mathrm{VC}(\mathcal{F}) + \log\frac{1}{\delta}}{n\mathbb{P}[\hat{Y} \neq \varnothing]\min_y \mathbb{P}[\hat{Y} = y|\hat{Y} \neq \varnothing]}}\right),$$

where $\tilde{\mathcal{O}}$ hides log factors in m and $\mathrm{VC}(\mathcal{F})$.

Proof. For space, we defer the proof and and bibliographic commentary to Appendix A. □

This bound formalizes the tradeoff between the *precision* of the weak labels, measured by α and γ, and the *coverage*, measured by $n\mathbb{P}[Y \neq \varnothing]$, which for large enough samples is very close to the size of the *covered* training set $\mathcal{T} = \{(x_i, \hat{Y}(x_i)) : \hat{Y}(x_i) \neq \varnothing\}$. Suppose we have a label model $\hat{Y}$ and an alternative label model $\hat{Y}'$ that abstains more often than $\hat{Y}$ (so $\mathbb{P}[Y' \neq \varnothing] < \mathbb{P}[\hat{Y} \neq \varnothing]$) but also has smaller values of α and γ. Then according to the bound, an end model trained with $\hat{Y}'$ can have better performance, and the empirical results in Section 4 confirm this trend.

This bound is useful for comparing two fixed label models $\hat{Y}, \hat{Y}'$ with different abstention rates and (α, γ) values. However, we have been concerned in this paper with selecting a subset $\mathcal{T}'$ of $\mathcal{T}$ based on a single label model $\hat{Y}$, and training using $\mathcal{T}'$. We can represent this subset selection with a new set of pseudolabels $\{\tilde{Y}(x_i) : x_i \in \mathcal{T}\}$ that abstains more than $\hat{Y}(x_i)$—i.e., points not chosen for $\mathcal{T}'$ get $\tilde{Y}(x_i) = \varnothing$. However, selection for $\mathcal{T}'$ depends on sample-level statistics, so the $\tilde{Y}$ values are not i.i.d., which complicates the generalization bound. We show in Appendix A that this can be remedied by a sample-splitting procedure: we use half of $\mathcal{T}$ to define a refined label model $\tilde{Y} : \Psi_0 \to \mathcal{Y} \cup \{\varnothing\}$, and then use the other half of $\mathcal{T}$ as the initial training set. This allows us to effectively reduce to the case of two fixed label models $\hat{Y}, \tilde{Y}$ and apply Theorem 1. We include the simpler $\hat{Y}$ versus $\hat{Y}'$ bound here because it captures the essential tradeoff without the technical difficulties.

6 Limitations, Societal Impact, and Conclusion

Surprisingly, using *less* data can greatly improve the performance of weak supervision pipelines when that data is carefully selected. By exploring the tradeoff between weak label precision and coverage, subset selection allows us to select a higher-quality training set without compromising generalization performance to the population pseudolabeling function. This improves the accuracy of the end model on the true labels. In Section 5, we showed that this tradeoff can be formalized in the special setting of conditional independence. By combining the cut statistic with good data representations, we developed a technique that improves performance for five different label models, over ten datasets, and three data modalities. Additionally, the hyperparameter tuning burden is low. We introduced one new hyperparameter β (the coverage fraction) and showed that all other hyperparameters can be re-used from the full-coverage $\beta = 1.0$ case, so existing tuned weak supervision pipelines can be easily adapted to use this technique.

However, this approach is not without limitations. The cut statistic requires a good representation ϕ of the input data to work well. Such a representation may not be available. However, for image or text data, pretrained representations provide natural choices for ϕ. Our results on the Census dataset in Section 4.3 indicate that using hand-crafted features as ϕ can also work well. Finally, as discussed at the end of Section 4.5, subset selection can further bias the input distribution (except in special cases like the one in Section 5). However, this is already an issue with current weak supervision methods. Most methods only train on the covered data $\mathcal{T}$. Labeling functions are typically deterministic functions of the input example, (such as functions based on the presence of certain tokens) and so the support of the full training set $\mathcal{T}$ is a strict subset of the support of the true input distribution, and $\mathcal{T}$ may additionally have a skewed distribution over its support. This underscores the need for (i) the use of a ground-truth validation set to ensure that the end model is an accurate predictor on the full distribution (ii) in high stakes settings, sub-group analyses such as those performed by [35], to ensure that the pseudolabels have not introduced bias against protected subgroups and (iii) the need for further theoretical understanding on why weakly supervised end models are able to perform well on the uncovered set $\{x : \mathbf{\Lambda}(x) = \varnothing\}$.

Acknowledgments and Disclosure of Funding

This work was supported by NSF AitF awards CCF-1637585 and CCF-1723344. Thanks to Hussein Mozannar for helpful conversations on Section 5 and the pointer to Woodworth et al. [39]. Thanks to Dr. Steven Horng for generously donating GPU-time on the BIDMC computing cluster [12] and to NVIDIA for their donation of GPUs also used in this work.

References

[1] Anelia Angelova, Yaser Abu-Mostafam, and Pietro Perona. Pruning training sets for learning of object categories. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 494–501. IEEE, 2005.

[2] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4): 343–370, 1988.

[3] Chidubem Arachie and Bert Huang. Adversarial label learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3183–3190, 2019.

[4] Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to statistical learning theory. In *Summer school on machine learning*, pages 169–207. Springer, 2003.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[6] Mayee F Chen, Daniel Yang Fu, Dyah Adila, Michael Zhang, Frederic Sala, Kayvon Fatahalian, and Christopher Re. Shoring up the foundations: Fusing model embeddings and weak supervision. In *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022.

[7] Hao Cheng, Zhaowei Zhu, Xingyu Li, Yifei Gong, Xing Sun, and Yang Liu. Learning with instance-dependent label noise: A sample sieve approach. In *International Conference on Learning Representations*, 2021.

[8] Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1):20–28, 1979.

[9] 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://aclanthology.org/N19-1423.

[10] Daniel Fu, Mayee Chen, Frederic Sala, Sarah Hooper, Kayvon Fatahalian, and Christopher Ré. Fast and three-rious: Speeding up weak supervision with triplet methods. In *International Conference on Machine Learning*, pages 3280–3291. PMLR, 2020.

[11] Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare (HEALTH)*, 3(1):1–23, 2021.

[12] Steven Horng. Machine Learning Core. 2 2022. doi: 10.6084/m9.figshare.19104917.v2. URL https://figshare.com/articles/preprint/Machine_Learning_Core/19104917.

[13] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, et al. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 590–597, 2019.

[14] Alistair EW Johnson, Tom J Pollard, Seth J Berkowitz, Nathaniel R Greenbaum, Matthew P Lungren, Chih-ying Deng, Roger G Mark, and Steven Horng. Mimic-cxr, a de-identified publicly available database of chest radiographs with free-text reports. *Scientific data*, 6(1):1–8, 2019.

[15] Giannis Karamanolakis, Subhabrata Mukherjee, Guoqing Zheng, and Ahmed Hassan. Self-training with weak supervision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 845–863, 2021.

[16] Hunter Lang, Monica N Agrawal, Yoon Kim, and David Sontag. Co-training improves prompt-based learning for large language models. In *International Conference on Machine Learning*, pages 11985–12003. PMLR, 2022.

[17] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[18] Junnan Li, Richard Socher, and Steven CH Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *International Conference on Learning Representations*, 2019.

[19] Ming Li and Zhi-Hua Zhou. Setred: Self-training with editing. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 611–621. Springer, 2005.

[20] 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. *arXiv preprint arXiv:1907.11692*, 2019.

[21] Ayush Maheshwari, Oishik Chatterjee, Krishnateja Killamsetty, Ganesh Ramakrishnan, and Rishabh Iyer. Semi-supervised data programming with subset selection. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4640–4651, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.408. URL https://aclanthology.org/2021.findings-acl.408.

[22] Aditya Menon, Brendan Van Rooyen, Cheng Soon Ong, and Bob Williamson. Learning from corrupted binary labels via class-probability estimation. In *International conference on machine learning*, pages 125–134. PMLR, 2015.

[23] Fabrice Muhlenbach, Stéphane Lallich, and Djamel A Zighed. Identifying and handling mislabelled instances. *Journal of Intelligent Information Systems*, 22(1):89–109, 2004.

[24] Subhabrata Mukherjee and Ahmed Awadallah. Uncertainty-aware self-training for few-shot text classification. *Advances in Neural Information Processing Systems*, 33:21199–21212, 2020.

[25] Curtis Northcutt, Lu Jiang, and Isaac Chuang. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70:1373–1411, 2021.

[26] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1944–1952, 2017.

[27] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[28] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 11, page 269. NIH Public Access, 2017.

[29] Alexander Ratner, Braden Hancock, Jared Dunnmon, Frederic Sala, Shreyash Pandey, and Christopher Ré. Training complex models with multi-task weak supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4763–4771, 2019.

[30] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. *Advances in neural information processing systems*, 29:3567–3575, 2016.

[31] Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations*, 2022.

[32] Norbert Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1):145–147, 1972.

[33] Clayton Scott, Gilles Blanchard, and Gregory Handy. Classification with asymmetric label noise: Consistency and maximal denoising. In Shai Shalev-Shwartz and Ingo Steinwart, editors, *Proceedings of the 26th Annual Conference on Learning Theory*, volume 30 of *Proceedings of Machine Learning Research*, pages 489–511, Princeton, NJ, USA, 12–14 Jun 2013. PMLR. URL https://proceedings.mlr.press/v30/Scott13.html.

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

[35] Laleh Seyyed-Kalantari, Guanxiong Liu, Matthew McDermott, Irene Y Chen, and Marzyeh Ghassemi. Chexclusion: Fairness gaps in deep chest x-ray classifiers. In *BIOCOMPUTING 2021: Proceedings of the Pacific Symposium*, pages 232–243. World Scientific, 2020.

[36] Saharon Shelah. A combinatorial problem; stability and order for models and theories in infinitary languages. *Pacific Journal of Mathematics*, 41(1):247–261, 1972.

[37] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[38] VN Vapnik. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–281, 1971.

[39] Blake Woodworth, Suriya Gunasekar, Mesrob I Ohannessian, and Nathan Srebro. Learning non-discriminatory predictors. In *Conference on Learning Theory*, pages 1920–1953. PMLR, 2017.

[40] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA, June 1995. Association for Computational Linguistics. doi: 10.3115/981658.981684. URL https://aclanthology.org/P95-1026.

[41] Yue Yu, Simiao Zuo, Haoming Jiang, Wendi Ren, Tuo Zhao, and Chao Zhang. Fine-tuning pre-trained language model with weak supervision: A contrastive-regularized self-training approach. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1063–1077, 2021.

[42] Jieyu Zhang, Yue Yu, Yinghao Li, Yujing Wang, Yaming Yang, Mao Yang, and Alexander Ratner. Wrench: A comprehensive benchmark for weak supervision. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[43] Min-Ling Zhang and Zhi-Hua Zhou. Cotrade: Confident co-training with data editing. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(6):1612–1626, 2011.

[44] Yan Zhou, Murat Kantarcioglu, and Bhavani Thuraisingham. Self-training with selection-by-rejection. In *2012 IEEE 12th international conference on data mining*, pages 795–803. IEEE, 2012.

[45] Zhaowei Zhu, Zihao Dong, Hao Cheng, and Yang Liu. A good representation detects noisy labels. *arXiv preprint arXiv:2110.06283*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We show both empirical and theoretical benefits of using *less* weakly-labeled data to improve performance.
   (b) Did you describe the limitations of your work? [Yes] In Sections 4.5 and 6.

(c) Did you discuss any potential negative societal impacts of your work? [Yes] In Section 6.

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [Yes] In Assumption 1 and in the statement of Theorem 1.

(b) Did you include complete proofs of all theoretical results? [Yes] In Appendix A.

3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] In Appendix C and in a ZIP file in the supplementary material.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] In Section 4.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Our tables in Section 4 include standard deviations across 5 runs.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Described in Section 4.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] We primarily used the WRENCH benchmark [42].

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include the code for reproducing our empirical results in the supplementary material.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Proofs from Section 5

To ease notation in this section, we consider the multimodal setting $\mathcal{X} = \mathcal{X}^{(0)} \times \mathcal{X}^{(1)}$. The extension to arbitrary $\phi_0(X)$, $\phi_1(X)$ is straightforward. First, we fix an arbitrary label model $\hat{Y}$ and, we assume for this part that $\hat{Y}$ can be written as a function mapping single examples to pseudolabels: $\hat{Y} : \mathcal{X}^{(0)} \to \mathcal{Y} \cup \{\varnothing\}$. We first prove Theorem 1 for this case. Second, we discuss the extention of this theorem to the case where the pseudolabels $\tilde{Y}$ come from some label model $\hat{Y}$ plus a subset selector such as the cut statistic. This complicates the situation because the post-subselection pseudolabels $\{\tilde{Y}(x_i)\}$ (i.e., the output of the cut statistic on the training set) cannot be written as i.i.d. samples of a "refined label model" $\tilde{Y} : \mathcal{X}^{(0)} \to \mathcal{Y} \cup \{\varnothing\}$. We show how to use sample splitting to suitably define the population-level function $\tilde{Y} : \mathcal{X}^{(0)} \to \mathcal{Y} \cup \{\varnothing\}$, which allows us to directly apply Theorem 1.

A.1 Proof of Theorem 1.

Suppose that $\mathcal{Y} = \{0, 1\}$. Recall the conditional independence assumption:

Assumption (Conditional independence)**.** *The random variables $X^{(0)}$, $X^{(1)}$ are conditionally independent given the true (unobserved) label Y. That is, for any $A \subset \mathcal{X}^{(0)}$, $B \subset \mathcal{X}^{(1)}$,*

$$\mathbb{P}_{X,Y}[X^{(0)} \in A, X^{(1)} \in B|Y] = \mathbb{P}_{X,Y}[X^{(0)} \in A|Y]\mathbb{P}_{X,Y}[X^{(1)} \in B|Y].$$

Let $\hat{Y} : \mathcal{X}^{(0)} \to \mathcal{Y} \cup \{\varnothing\}$ be an arbitrary label model, and define:

$$\alpha = \mathbb{P}_{X,Y}[Y = 0|\hat{Y}(X^{(0)}) = 1]$$
$$\gamma = \mathbb{P}_{X,Y}[Y = 1|\hat{Y}(X^{(0)}) = 0]$$

These parameters measure the amount of noise in $\hat{Y}$. We assume throughout that $\mathbb{P}[\hat{Y} = y] > 0$ for $y \in \{0, 1\}$ and that $\alpha + \gamma < 1$. Note that this implies $\mathbb{P}[\hat{Y} \neq \varnothing|Y = y] > 0$ for all y, since otherwise either α or γ is 1. So there are pseudolabeled examples from both conditional distributions $\mathbb{P}[X|Y = y]$.

Theorem. *Suppose we observe a weakly labeled training set $\bar{\mathcal{T}} := \{x_i^{(0)}, x_i^{(1)}, \hat{Y}(x_i^{(0)})\}_{i=1}^n$, where the x_i's are drawn i.i.d. from the marginal distribution $\mathbb{P}_X$. This differs from $\mathcal{T}$ in the main text because we include both views $x^{(0)}, x^{(1)}$ and also include the points where $\hat{Y}(x_i^{(0)}) = \varnothing$.*

Let $\mathcal{F}$ be a hypothesis class consisting of functions $f : \mathcal{X}^{(1)} \to \mathcal{Y}$. Suppose Assumption 1 holds, and that $\mathbb{P}[\hat{Y} = y] > 0$ for $y \in \{0, 1\}$ and that $\alpha + \gamma < 1$. Define the balanced error *of a classifier f on labels $Z \in \{0, 1\}$ as:*

$$err_{bal}(f, Z) = \frac{1}{2}(\mathbb{P}[f(X) = 0|Z = 1] + \mathbb{P}[f(X) = 1|Z = 0]).$$

We will consider both $Z = Y$ and $Z = \hat{Y}$. Let $f \in \mathcal{F}$ be an arbitrary classifier. Then f's true balanced error $err_{bal}(f, Y)$ and f's pseudolabel balanced error $err_{bal}(f, \hat{Y})$ satisfy:

$$err_{bal}(f, Y) = \frac{1}{1 - \alpha - \gamma}\left[err_{bal}(f, \hat{Y}) - \frac{\alpha + \gamma}{2}\right].$$

Now let $f^* := \inf_{f \in \mathcal{F}} err_{bal}(f, Y)$ be the classifier with optimal balanced accuracy on the true labels. Suppose that $\hat{f}$ is the classifier obtained by minimizing the empirical balanced accuracy on the dataset $\mathcal{T} = \{(x_i^{(1)}, \hat{Y}(x_i^{(0)}))\}$:

$$\hat{f} := \underset{f \in \mathcal{F}}{argmin}\, \widehat{err}_{bal}(f, \hat{Y}) = \frac{1}{2}\left(\frac{\sum_{i=1}^n \mathbb{1}_{f(x_i^{(1)})=0}\mathbb{1}_{\hat{Y}(x_i^{(0)})=1}}{\sum_{i=1}^n \mathbb{1}_{\hat{Y}(x_i^{(0)})=1}} + \frac{\sum_{i=1}^n \mathbb{1}_{f(x_i^{(1)})=1}\mathbb{1}_{\hat{Y}(x_i^{(0)})=0}}{\sum_{i=1}^n \mathbb{1}_{\hat{Y}(x_i^{(0)})=0}}\right).$$

Note that the training points $\{x_i : \hat{Y}(x_i^{(0)}) = \varnothing\}$ do not feature at all in the above expression, so we can safely discard them for the training step. Then for any $\delta > 0$ the following holds with probability

at least $1-\delta$ *over the sampling of* $\overline{\mathcal{T}}$*:*

$$err_{bal}(f,Y) - err_{bal}(f^*,Y) \leqslant \widetilde{\mathcal{O}}\left(\frac{1}{1-\alpha-\gamma}\sqrt{\frac{VC(\mathcal{F})+\log\frac{1}{\delta}}{n\mathbb{P}[\hat{Y}\neq\varnothing]\min_y \mathbb{P}[\hat{Y}=y|\hat{Y}\neq\varnothing]}}\right),$$

where $\widetilde{\mathcal{O}}$ *hides log factors in* m *and* $\mathrm{VC}(\mathcal{F})$*.*

Proof. Lemma 1 proves that for any $f \in \mathcal{F}$,

$$\mathrm{err}_{bal}(f,Y) = \frac{1}{1-\alpha-\gamma}\left[\mathrm{err}_{bal}(f,\hat{Y}) - \frac{\alpha+\gamma}{2}\right].$$

Subtracting $\mathrm{err}_{bal}(f^*,\hat{Y})$ from both sides:

$$\mathrm{err}_{bal}(\hat{f},Y) - \mathrm{err}_{bal}(f^*,Y) = \frac{1}{1-\alpha-\gamma}\left[\mathrm{err}_{bal}(\hat{f},\hat{Y}) - \mathrm{err}_{bal}(f^*,\hat{Y})\right].$$

Let $\hat{f}^*$ be the classifier in $\mathcal{F}$ with optimal population-level balanced error on $\hat{Y}$:

$$\hat{f}^* := \operatorname*{argmin}_{f\in\mathcal{F}} \mathrm{err}_{bal}(f,\hat{Y}).$$

Then

$$\mathrm{err}_{bal}(\hat{f},Y) - \mathrm{err}_{bal}(f^*,Y) \leqslant \frac{1}{1-\alpha-\gamma}\left[\mathrm{err}_{bal}(\hat{f},\hat{Y}) - \mathrm{err}_{bal}(\hat{f}^*,\hat{Y})\right]. \tag{2}$$

Now we need to control $\mathrm{err}_{bal}(\hat{f},\hat{Y}) - \mathrm{err}_{bal}(\hat{f}^*,\hat{Y})$, the excess risk of $\hat{f}$ on the weak labels $\hat{Y}$. From $\overline{T}$, we can form a sample of n i.i.d. points $\{(x_i^{(1)},\hat{y}_i)\}_{i=1}^n$ from the joint distribution: $\mathbb{P}_X[X^{(1)},\hat{Y}(X^{(0)})]$.

Theorem 2 implies that for any $\delta > 0$, with probability at least $1-\delta$ over the sampling of $\{(x_i^{(1)},\hat{y}_i)\}_{i=1}^n$, we have the following deviation bound:

$$\sup_{f\in\mathcal{F}} \mathbb{P}[|\widehat{\mathrm{err}}_{bal}(f,\hat{Y}) - \mathrm{err}_{bal}(f,\hat{Y})|] \leqslant \widetilde{\mathcal{O}}\left(\sqrt{\frac{VC(\mathcal{F})+\log\frac{1}{\delta}}{n\mathbb{P}[\hat{Y}\neq\varnothing]\min_y \mathbb{P}[\hat{Y}=y|\hat{Y}\neq\varnothing]}}\right).$$

We prove Theorem 2 in the self-contained Section A.3. We can easily turn this uniform convergence result into an excess risk bound for $\hat{f}$ with a standard sequence of inequalities. We drop the subscript and remove $\hat{Y}$ from the error arguments for convenience, so $\mathrm{err}(\cdot)$ in the following refers to $\mathrm{err}_{bal}(\cdot,\hat{Y})$:

$$\begin{aligned}
\mathrm{err}(\hat{f}) - \mathrm{err}(\hat{f}^*) &= \widehat{\mathrm{err}}(\hat{f}) - \widehat{\mathrm{err}}(\hat{f}^*) + \mathrm{err}(f) - \widehat{\mathrm{err}}(f) + \widehat{\mathrm{err}}(\hat{f}^*) - \mathrm{err}(\hat{f}^*) \\
&\leqslant \mathrm{err}(f) - \widehat{\mathrm{err}}(f) + \widehat{\mathrm{err}}(\hat{f}^*) - \mathrm{err}(\hat{f}^*) \\
&\leqslant |\mathrm{err}(f) - \widehat{\mathrm{err}}(f)| + |\widehat{\mathrm{err}}(\hat{f}^*) - \mathrm{err}(\hat{f}^*)| \\
&\leqslant_{\text{w.p. } 1-\delta} \widetilde{\mathcal{O}}\left(\sqrt{\frac{VC(\mathcal{F})+\log\frac{1}{\delta}}{n\mathbb{P}[\hat{Y}\neq\varnothing]\min_y \mathbb{P}[\hat{Y}=y|\hat{Y}\neq\varnothing]}}\right).
\end{aligned}$$

The first inequality used that $\widehat{\mathrm{err}}(\hat{f}) - \widehat{\mathrm{err}}(\hat{f}^*) \leqslant 0$ since $\hat{f}$ is the empirical minimizer, and the last inequality applied the deviation bound to each term. Hence we have shown that with probability at least $1-\delta$,

$$\mathrm{err}_{bal}(\hat{f},\hat{Y}) - \mathrm{err}_{bal}(\hat{f}^*,\hat{Y}) \leqslant \widetilde{\mathcal{O}}\left(\sqrt{\frac{VC(\mathcal{F})+\log\frac{1}{\delta}}{n\mathbb{P}[\hat{Y}\neq\varnothing]\min_y \mathbb{P}[\hat{Y}=y|\hat{Y}\neq\varnothing]}}\right).$$

Plugging this in to (2) completes the proof of Theorem 1. □

Lemma 1 ([33], [22]). *Suppose Assumption 1 holds and that $\alpha+\gamma < 1$, $\mathbb{P}[\hat{Y} = y] > 0$ for $y \in \{0, 1\}$. Then for any $f \in \mathcal{F}$, the balanced errors on $\hat{Y}$ and Y satisfy the following relationship:*

$$err_{bal}(f, Y) = \frac{1}{1-\alpha-\gamma}\left[err_{bal}(f, \hat{Y}) - \frac{\alpha+\gamma}{2}\right].$$

Proof. The formula relating $\text{err}_{bal}(f, \hat{Y})$ and $\text{err}_{bal}(f, Y)$ is due to Scott et al. [33], Menon et al. [22]. We reprove it here to show that $\mathbb{P}[\hat{Y} = \varnothing] > 0$ does not affect the result, since those works consider $\hat{Y}(x_i^{(0)}) \in \{0, 1\}$. Define:

$$\begin{aligned}
\widehat{\text{FNR}}(f) &= \mathbb{P}[f(X^{(1)}) = 0 | \hat{Y}(X^{(0)}) = 1] \\
\widehat{\text{FPR}}(f) &= \mathbb{P}[f(X^{(1)}) = 1 | \hat{Y}(X^{(0)}) = 0] \\
\text{FNR}(f) &= \mathbb{P}[f(X^{(1)}) = 0 | Y = 1] \\
\text{FPR}(f) &= \mathbb{P}[f(X^{(1)}) = 1 | Y = 0].
\end{aligned}$$

Observe that:

$$\begin{aligned}
\widehat{\text{FNR}}(f) &= \mathbb{P}[f(X^{(1)}) = 0 | \hat{Y}(X^{(0)}) = 1] = \sum_y \mathbb{P}[f(X^{(1)}) = 0, Y = y | \hat{Y}(X^{(0)}) = 1] \\
&= \sum_y \frac{\mathbb{P}[f(X^{(1)}) = 0, \hat{Y}(X^{(0)}) = 1 | Y = y]\mathbb{P}[Y = y]}{\mathbb{P}[\hat{Y}(X^{(0)}) = 1]} \\
&= \sum_y \frac{\mathbb{P}[f(X^{(1)}) = 0 | Y = y]\mathbb{P}[\hat{Y}(X^{(0)}) = 1 | Y = y]\mathbb{P}[Y = y]}{\mathbb{P}[\hat{Y}(X^{(0)}) = 1]} \\
&= \sum_y \mathbb{P}[f(X^{(1)}) = 0 | Y = y]\mathbb{P}[Y = y | \hat{Y}(X^{(0)}) = 1] \\
&= \mathbb{P}[f = 0 | Y = 0]\mathbb{P}[Y = 0 | \hat{Y} = 1] + \mathbb{P}[f = 0 | Y = 1]\mathbb{P}[Y = 1 | \hat{Y} = 1] \\
&= (1 - \text{FPR}(f))\alpha + \text{FNR}(f)(1 - \alpha).
\end{aligned}$$

Similarly,

$$\widehat{\text{FPR}}(f) = (1 - \text{FNR}(f))\gamma + \text{FPR}(f)(1 - \gamma).$$

Collecting these equalities gives:

$$\begin{bmatrix} (1-\gamma) & -\gamma \\ -\alpha & (1-\alpha) \end{bmatrix} \begin{bmatrix} \text{FPR}(f) \\ \text{FNR}(f) \end{bmatrix} + \begin{bmatrix} \gamma \\ \alpha \end{bmatrix} = \begin{bmatrix} \widehat{\text{FPR}}(f) \\ \widehat{\text{FNR}}(f) \end{bmatrix}$$

The coefficient matrix is invertible since we assumed $\alpha + \gamma < 1$. Multiplying both sides by its inverse gives:

$$\begin{aligned}
\text{FPR}(f) &= \frac{1}{1-\alpha-\gamma}\left((1-\alpha)\widehat{\text{FPR}}(f) + \gamma\widehat{\text{FNR}}(f) - \gamma\right) \\
\text{FNR}(f) &= \frac{1}{1-\alpha-\gamma}\left(\alpha\widehat{\text{FPR}}(f) + (1-\gamma)\widehat{\text{FNR}}(f) - \alpha\right)
\end{aligned}$$

Finally, plugging these in to $\text{err}_{bal}(f, Y) = \frac{1}{2}\left(\text{FPR}(f) + \text{FNR}(f)\right)$ gives the first result. □

A.2 Dealing with subset selection

Suppose $\hat{Y}$ is some fixed label model (such as majority vote). Let $\bar{\mathcal{T}} = \{(x_i^{(0)}, x_i^{(1)}, \hat{Y}(x_i^{(0)}))\}_{i=1}^n$ be the full weakly-labeled sample, including points where $\hat{Y} = \varnothing$. We have assumed the x_i's are drawn i.i.d. from some distribution $\mathbb{P}_X$ over $\mathcal{X}$ satisfying:

$$\mathbb{P}[X^{(0)} \in A, X^{(1)} \in B | Y] = \mathbb{P}[X^{(0)} \in A | Y]\mathbb{P}[X^{(1)} \in B | Y]$$

for any $A \subset \mathcal{X}^{(0)}$, $B \subset \mathcal{X}^{(1)}$. Let $\mathcal{S}$ be a function that maps sets $\{(x_i^{(0)}, x_i^{(1)}, \hat{Y}(x_i^{(0)}))\}_{i=1}^n$ to $\{0,1\}^n$, where $S(\bar{\mathcal{T}})_i$ indicates whether to include example i in the subset. We assume that $\mathcal{S}$ only uses the information $\{(x_i^{(0)}, \hat{Y}(x_i^{(0)}))\}$, i.e., that membership in the subset does not depend directly on $x_i^{(1)}$. We can also consider $\mathcal{S}$ as defining a new set of pseudolabels $\tilde{Y}(x_i^{(0)})$ for $\bar{\mathcal{T}}$. Define these new pseudolabels $\tilde{Y}(x_i^{(0)}) \in \mathcal{Y} \cup \{\varnothing\}$ as:

$$\tilde{Y}(x_i^{(0)}) = \begin{cases} \hat{Y}(x_i^{(0)}) & \mathcal{S}(\mathcal{T})_i = 1 \\ \varnothing & \mathcal{S}(\mathcal{T})_i = 0. \end{cases}$$

We then use the refined set $\mathcal{T}' = \{(x_i^{(1)}, \tilde{Y}(x_i^{(0)}))\}$ to train the end model (recall that the end model only uses $x^{(1)}$, and that points where $\tilde{Y}(x_i^{(0)}) = \varnothing$ can be safely ignored during the training process, since they do not appear in the loss function.

Importantly, the $\mathcal{S}$ we have studied in this work operate at the set level, and so *cannot* be written as

$$\mathcal{S}(\bar{\mathcal{T}}) = \{\tilde{\mathcal{S}}(x_1^{(0)}), \tilde{\mathcal{S}}(x_2^{(0)}), \ldots, \tilde{\mathcal{S}}(x_m^{(0)})\} \in \{0,1\}^n$$

for an *example*-level selector function $\tilde{\mathcal{S}} : \mathcal{X} \times (\mathcal{Y} \cup \{\varnothing\}) \to \{0,1\}$. For example, both the cut statistic and entropy scoring use a *percentile-based* ranking of examples, and choose the top β fraction of examples in $\bar{\mathcal{T}}$ to have $\mathcal{S}(\bar{\mathcal{T}})_i = 1$. The value of the threshold for inclusion in the subset $\mathcal{T}'$ thus depends on the entire sample $\bar{\mathcal{T}}$ and not just on a single example.

Ultimately, we would like to obtain a generalization bound for an end model trained with the refined pseudolabels $\tilde{Y}$. However, the set-level form of $\mathcal{S}$ presents an issue, because it is unclear how to even define the population-level quantities considered in Theorem 1, such as $\alpha = \mathbb{P}_{X,Y}[Y = 0 | \tilde{Y} = 1]$. How do we define $\tilde{Y}(x)$ for x that do not appear in the sample $\bar{\mathcal{T}}$? Additionally, the samples $\{(x_i^{(1)}, \tilde{Y}(x_i^{(0)}))\}_{i=1}^n$ are not i.i.d., which means we can't directly apply Theorem 1. To see this, consider the case where $\beta = \frac{1}{m} + \epsilon$, i.e., the selection percentile is set so that only one example is chosen to have $\tilde{Y}(x_i^{(0)}) \neq \varnothing$. Then observing a sample $(x_i^{(1)}, \tilde{Y}(x_i^{(0)}))$ with $\tilde{Y}(x_i^{(0)}) \neq \varnothing$ implies that $\tilde{Y}(x_j^{(0)}) = \varnothing$ for all $j \neq i$. To resolve these issues, we use a straightforward sample-splitting scheme.

Suppose we partition $\bar{\mathcal{T}}$ into two halves, $\bar{\mathcal{T}} = (\bar{\mathcal{T}}_0, \bar{\mathcal{T}}_1)$. We can use $\bar{\mathcal{T}}_0$ to produce an example-level selector $\mathcal{S}$, i.e., a selector such that for any $U \subset (\mathcal{X} \times \mathcal{Y})^n$, $U = \{(x_i, \hat{Y}(x_i))\}_{i=1}^n$:

$$\mathcal{S}(U) = \{\tilde{\mathcal{S}}(x_1^{(0)}), \tilde{\mathcal{S}}(x_2^{(0)}), \ldots, \tilde{\mathcal{S}}(x_n^{(0)})\} \in \{0,1\}^m.$$

for some $\tilde{\mathcal{S}} : \mathcal{X}^{(0)} \to \{0,1\}$.

For example, for the cut statistic, to compute $\tilde{\mathcal{S}}(x^{(0)})$ for an arbitrary x, we first compute $\hat{Y}(x^{(0)})$, and then compute the K-nearest neighbors of x among the non-abstaining points in $\bar{\mathcal{T}}_0$. I.e., we insert x into the nearest-neighbor graph over $\bar{\mathcal{T}}_0$ with the pseudolabel $\hat{Y}(x)$. Next, we use this graph and the empirical distribution of $\hat{Y}$ from $\bar{\mathcal{T}}_0$ to compute the Z-score for x. The last step is to threshold the Z-score to decide whether to set $\tilde{\mathcal{S}}(x) = 1$. We compute the quantiles of Z-score for the examples $\bar{\mathcal{T}}_0$, and we set $\tilde{\mathcal{S}}(x^{(0)}) = 1$ if x's Z-score would have been in the top β fraction in $\bar{\mathcal{T}}_0$. To summarize, to compute $\tilde{Y}(x^{(0)})$ we essentially perform the cut statistic on the set $\bar{\mathcal{T}}_0 \cup \{(x^{(0)}, x^{(1)}, \hat{Y}(x^{(0)}))\}$ (but where x is left out of the percentile computations, and its Z-score is compared to the quantiles computed on $\bar{\mathcal{T}}_0$).

This sample-splitting trick allows us to extend the selection beyond the training sample to easily define a population-level $\tilde{Y}(X^{(0)})$:

$$\tilde{Y}(X^{(0)}) = \begin{cases} \hat{Y}(X^{(0)}) & \tilde{\mathcal{S}}(X^{(0)}) = 1 \\ \varnothing & \tilde{\mathcal{S}}(X^{(0)}) = 0 \end{cases}$$

We can now compute the relevant α and γ parameters for $\tilde{Y}$, i.e., $\mathbb{P}[Y = 0 | \tilde{Y} = 1]$ and $\mathbb{P}[Y = 1 | \tilde{Y} = 0]$ are well-defined. Note that these parameters depend on $\bar{\mathcal{T}}_0$. We can then use $\bar{\mathcal{T}}_1$ in place of

$\bar{\mathcal{T}}$ as the full training data, and treat $\tilde{Y}$ like an arbitrary label model and applying Theorem 1. Note that the generalization bound now holds with respect to the sampling of $\bar{\mathcal{T}}_1$, while holding $\bar{\mathcal{T}}_0$ fixed. As a final check, observe that subset selection using $\mathcal{S}$ does not affect conditional independence: for $y \neq \varnothing$ and $B \subset \mathcal{X}^{(1)}$,

$$\begin{aligned}\mathbb{P}[X^{(1)} \in B, \tilde{Y}(X) = y|Y] &= \mathbb{P}[X^{(1)} \in B, \hat{Y}(X^{(0)}) = y, \mathcal{S}(X^{(0)}) = 1|Y] \\ &= \mathbb{P}[X^{(1)} \in B|Y]\mathbb{P}[\hat{Y}(X^{(0)}) = y, \mathcal{S}(X^{(0)}) = 1|Y] \\ &= \mathbb{P}[X^{(1)} \in B|Y]\mathbb{P}[\tilde{Y}|X].\end{aligned}$$

The first equality recalled that $\hat{Y}(X)$ and $\mathcal{S}(X)$ only depend on ϕ_0, and the second used conditional independence of $X^{(0)}$ and $X^{(1)}$ given Y. The proof for $y = \varnothing$ is similar. Therefore, we can still apply Lemma 1 with $\tilde{Y}$. The samples $\mathcal{T}_1 = \{(x_i^{(1)}, \tilde{Y}(x_i^{(0)})\}$ (which we obtained from $\bar{\mathcal{T}}_1$ by discarding $x_i^{(0)}$) are clearly i.i.d. because we assumed that $\{(x_i^{(1)}, x_i^{(0)})\}_{i=1}^n$ were i.i.d. samples. Hence, we can still apply the Theorem 2 result in the proof of Theorem 1. This sample-splitting construction of $\tilde{Y}$ allowed us to reduce to the case of a fixed, population level label model and directly use Theorem 1 to give bounds on the end model error when training with the refined pseudolabels.

A.3 Balanced error generalization bound: Notation and result

The notation in this section is self-contained and slightly differs from that of previous sections. Let $\mathcal{X}$ be an input space and $\mathcal{Y} = \{0, 1, \varnothing\}$ be the (binary) label space + an abstention symbol. Let $\mathcal{H}$ be a class of functions mapping $\mathcal{X} \to \{0, 1\}$. We assume $(X, Y) \in \mathcal{X} \times \mathcal{Y}$ is a pair of random variables distributed according to an unknown distribution $\mathbb{P}$. We observe a sequence of n i.i.d. pairs (X_i, Y_i) sampled according to $\mathbb{P}$, and the goal is to learn a classifier $h \in \mathcal{H}$ with low *balanced error*:

$$R(h) := \frac{1}{2}\left(\mathbb{P}[h(X) = 1|Y = 0] + \mathbb{P}[h(X) = 0|Y = 1]\right),$$

To measure classifier performance from our finite sample, we use the *empirical balanced error*:

$$R_n(h) := \frac{1}{2}\left(\frac{\sum_{i=1}^n \mathbb{1}_{h(X_i)=1}\mathbb{1}_{Y_i=0}}{\sum_{i=1}^n \mathbb{1}_{Y_i=0}} + \frac{\sum_{i=1}^n \mathbb{1}_{h(X_i)=0}\mathbb{1}_{Y_i=1}}{\sum_{i=1}^n \mathbb{1}_{Y_i=1}}\right)$$

Note that the points where $Y = \varnothing$ do not appear in either expression. The goal is to derive a bound on the *generalization gap* $R(h) - R_n(h)$ for a classifier $\hat{h}$ that is learned from (and hence, depends on) the the finite sample $\{(X_i, Y_i) : i\}$. The challenge lies in the presence of random variables in the denominator of $R_n(h)$, so unlike the empirical zero-one loss, it cannot be written simply as $\frac{1}{n}\sum_{i=1}^n g(h, x, y)$ for some indicator function g.

The following theorem gives a uniform (over $\mathcal{H}$) convergence result for the balanced error. The key technique used in the proof is essentially due to Woodworth et al. [39].

Theorem 2. *For any $\delta \in (0, 1)$ and any distribution P, with probability at least $1 - \delta$ over the sampling of $\{(X_i, Y_i)\}_{i=1}^n$,*

$$\sup_{h\in\mathcal{H}} R(h) - R_n(h) \leqslant \tilde{\mathcal{O}}\left(\sqrt{\frac{\mathrm{VC}(\mathcal{H}) + \log\frac{1}{\delta}}{n\mathbb{P}[Y \neq \varnothing]\min_{y\in\{0,1\}}\mathbb{P}[Y = y|Y \neq \varnothing]}}\right),$$

where $\tilde{\mathcal{O}}$ hides log factors in n and $\mathrm{VC}(\mathcal{H})$.

Proof. Let $S = \{(X_i, Y_i)\}_{i=1}^n$ refer to the empirical sample. For convenience, for each $(\bar{y}, y) \in \{0, 1\} \times \{0, 1\}$, define:

$$\gamma_{\bar{y}y}(h) = \mathbb{P}[h(X) = \bar{y}|Y = y],$$

and its empirical analogue:

$$\gamma_{\bar{y}y}^S(h) = \frac{\sum_{i=1}^n \mathbb{1}_{h(X_i)=\bar{y}}\mathbb{1}_{Y_i=y}}{\sum_{i=1}^n \mathbb{1}_{Y_i=y}}.$$

Then $R(h) = \frac{1}{2}(\gamma_{01} + \gamma_{10})$ and $R_n(h) = \frac{1}{2}(\gamma_{01}^S + \gamma_{10}^S)$. For sample S, let

$$\mathcal{I}_y = \{i \in [n] : Y_i = y\}$$

be the set of indices i where $Y_i = y$, and set $n_y^S = |\mathcal{I}_y|$. Conditioned on $\mathcal{I}_y$, the γ^S variables are distributed as:

$$\gamma_{\bar{y}y}^S(h)|\mathcal{I}_y \sim \frac{1}{n_y^S}\text{Binomial}(\gamma_{\bar{y}y}, n_y^S),$$

since the randomness over X in the sample gives n_y^S independent trials to make $h(X_i)$ equal to $\bar{y}$. We hide the argument h below for convenience. Observe that $\mathbb{E}\left[\gamma_{\bar{y}y}^S \mid \mathcal{I}_y\right] = \gamma_{\bar{y}y}$. Then for every $\eta > 0$,

$$\begin{aligned}
\mathbb{P}[|\gamma_{\bar{y}y}^S - \gamma_{\bar{y}y}| > t] &= \sum_{\mathcal{I}_y} \mathbb{P}\left[|\gamma_{\bar{y}y}^S - \gamma_{\bar{y}y}| > t \mid \mathcal{I}_y\right] \mathbb{P}[\mathcal{I}_y] \\
&\leqslant \mathbb{P}\left[n_y^S < (1-\eta)n\mathbb{P}[Y=y]\right] + \sum_{\mathcal{I}_y : n_y^S \geqslant (1-\eta)n\mathbb{P}[Y=y]} \mathbb{P}\left[|\gamma_{\bar{y}y}^S - \gamma_{\bar{y}y}| > t \mid \mathcal{I}_y\right] \mathbb{P}[\mathcal{I}_y] \\
&\leqslant \exp\left(-\frac{\eta^2 n\mathbb{P}[Y=y]}{2}\right) + \sum_{\mathcal{I}_y : n_y^S \geqslant (1-\eta)n\mathbb{P}[Y=y]} 2\exp(-2n_y^S t^2)\mathbb{P}[\mathcal{I}_y] \\
&\leqslant \exp\left(-\frac{\eta^2 n\mathbb{P}[Y=y]}{2}\right) + 2\exp(-2t^2(1-\eta)n\mathbb{P}[Y=y])
\end{aligned}$$

The first inequality comes from simplifying the sum over all 2^n possible values of $\mathcal{I}_y$. The second comes from applying a Chernoff bound to $\text{Binomial}(\mathbb{P}[Y=y], n)$ and Hoeffding's inequality to $\gamma_{\bar{y}y}$. We can set η to balance these terms:

$$\frac{\eta^2}{2} = 2t^2(1-\eta),$$

which yields:

$$\eta = 2\left(\sqrt{t^4+t^2} - t^2\right),$$

since the other root is negative. Substituting η gives:

$$\mathbb{P}[|\gamma_{\bar{y}y}^S - \gamma_{\bar{y}y}| > t] \leqslant 3\exp\left(-2\left(\sqrt{t^4+t^2} - t^2\right)^2 n\mathbb{P}[Y=y]\right)$$

For $t \in (0,1)$, $\sqrt{t^4+t^2} - t^2 \geqslant t/4$, so

$$\mathbb{P}[|\gamma_{\bar{y}y}^S - \gamma_{\bar{y}y}| > t] \leqslant 3\exp\left(-\frac{t^2}{8}n\mathbb{P}[Y=y]\right)$$

Hence, for $t \in (0,1)$,

$$\begin{aligned}
\mathbb{P}[|R(h) - R_n(h)| > t] &\leqslant \mathbb{P}[|\gamma_{01} - \gamma_{01}^S| + |\gamma_{10} - \gamma_{10}^S| > 2t] \\
&\leqslant \mathbb{P}[|\gamma_{01} - \gamma_{01}^S| > t] + \mathbb{P}[|\gamma_{10} - \gamma_{10}^S| > t] \\
&\leqslant 6\exp\left(-\frac{t^2}{8}n \min_{y\in\{0,1\}} \mathbb{P}[Y=y]\right). \\
&= 6\exp\left(-\frac{t^2}{8}n\mathbb{P}[Y \neq \varnothing] \min_{y\in\{0,1\}} \mathbb{P}[Y=y|Y\neq\varnothing]\right).
\end{aligned}$$

Now we show how to apply this deviation bound for R in place of Hoeffding's inequality in the symmetrization argument from Bousquet et al. [4].

Lemma 2 (Symmetrization)**.** *Let $Z = (X, Y)$ and suppose we have a* ghost sample *of n additional points Z_i' drawn i.i.d. from P. Let $R_n'(h)$ denote the empirical balanced error of classifier h on the ghost sample. Then for any $t > 0$ such that $nt^2 \geqslant \frac{32\log 12}{\min_{y\in\{0,1\}}\mathbb{P}[Y=y]}$:*

$$\mathbb{P}\left[\sup_{h\in\mathcal{H}} R(h) - R_n(h) > t\right] \leqslant 2\mathbb{P}\left[\sup_{h\in\mathcal{H}} R_n'(h) - R_n(h) > t/2\right].$$

Proof of Lemma 2. This follows Bousquet et al. [4] exactly, except we replace the application of one inequality with the deviation bound derived above. Let h_n be the function achieving the supremum on the left-hand-side. This depends on the sample $(Z_1, \ldots, Z_n)$.

$$\begin{aligned}\mathbb{1}_{R(h_n)-R_n(h_n)>t}\mathbb{1}_{R(h_n)-R'_n(h_n)<t/2} &= \mathbb{1}_{R(h_n)-R_n(h_n)>t \wedge R'_n(h_n)-R(h_n)\geqslant -t/2} \\ &\leqslant \mathbb{1}_{R'_n(h_n)-R_n(h_n)>t/2}\end{aligned}$$

Taking the expectation over the second sample $(Z'_1, \ldots, Z'_n)$,

$$\mathbb{1}_{R(h_n)-R_n(h_n)>t}\mathbb{P}'[R(h_n) - R'_n(h_n) < t/2] \leqslant \mathbb{P}'[R'_n(h_n) - R_n(h_n) > t/2]$$

From the result above,

$$\begin{aligned}P'[R(h_n) - R'_n(h_n) \geqslant t/2] &\leqslant 6\exp\left(-\frac{t^2}{32}n \min_{y\in\{0,1\}} \mathbb{P}[Y = y]\right) \\ &\leqslant \frac{1}{2}\end{aligned}$$

by the condition on nt^2. Hence

$$\mathbb{1}_{R(h_n)-R_n(h_n)>t} \leqslant 2\mathbb{P}'[R'_n(h_n) - R_n(h_n) > t/2],$$

and taking the expectation over the original sample $(Z_1, \ldots, Z_n)$ finishes the proof. □

Define $\mathcal{H}_{Z_1,\ldots,Z_n} = \{(h(x_1), \ldots h(x_n)) : h \in \mathcal{H}\}$. Recall that the *growth function* of class $\mathcal{H}$ is defined as $\mathcal{S}_{\mathcal{H}}(n) = \sup_{(Z_1,\ldots,Z_n)} |\mathcal{H}_{Z_1,\ldots,Z_n}|$. Now to finish the proof of Theorem 2, observe that the sup in the right-hand-side of the Lemma 2 result only depends on the *finite* set of vectors $\mathcal{H}_{Z_1,\ldots,Z_n,Z'_1,\ldots,Z'_n}$ That is,

$$\begin{aligned}\mathbb{P}\left[\sup_{h\in\mathcal{H}} R(h) - R_n(h) > t\right] &\leqslant 2\mathbb{P}\left[\sup_{h\in\mathcal{H}_{Z_1,\ldots,Z_n,Z'_1,\ldots,Z'_n}} R'_n(h) - R_n(h) > t/2\right] \\ &\leqslant 2\mathcal{S}_{\mathcal{H}}(2n) \max_{h\in\mathcal{H}_{Z_1,\ldots,Z_n,Z'_1,\ldots,Z'_n}} \mathbb{P}[R'_n(h) - R_n(h) > t/2] \\ &\leqslant 4\mathcal{S}_{\mathcal{H}}(2n)\mathbb{P}[R(h) - R_n(h) > t/4] \\ &\leqslant 24\mathcal{S}_{\mathcal{H}}(2n)\exp\left(-\frac{t^2}{128}n \min_{y\in\{0,1\}} \mathbb{P}[Y = y]\right),\end{aligned}$$

where in the first line we applied the definition of the growth function and used the union bound, and in the last line we applied the concentration result for fixed h. Recall that the Sauer-Shelah lemma [38, 32, 36] implies that for any class $\mathcal{H}$ with $\mathrm{VC}(\mathcal{H}) = d$, $\mathcal{S}_{\mathcal{H}}(n) \leqslant \left(\frac{en}{d}\right)^d$. Then setting:

$$t \geqslant 8\sqrt{2\frac{\mathrm{VC}(\mathcal{H})\log\frac{2en}{\mathrm{VC}(\mathcal{H})} + \log\frac{24}{\delta}}{n\min_{y\in\{0,1\}}\mathbb{P}[Y = y]}}$$

completes the proof. Note that this choice of t ensures that $nt^2 \geqslant \frac{32\log 12}{\min_{y\in\{0,1\}}\mathbb{P}[Y=y]}$ for any $\delta \in (0, 1)$. □

Table 4: Details for the WRENCH datasets used in this work.

Task	Domain	Dataset	Num. Labels	# Λ's	Train	Val	Test
Sentiment	Movie	IMDb	2	5	20,000	2,500	2,500
	Review	Yelp	2	8	30,400	3,800	3,800
Spam Classification	Comments	Youtube	2	10	1,586	200	250
Question Classification	Web Query	TREC	6	68	4,965	500	500
Relation Classification	Web Text	SemEval	9	164	1,749	200	692
	Chemical	ChemProt	10	26	12,861	1,607	1,607
	Biomedical	CDR	2	33	8,430	920	4,673
Image Classification	Video Frames	Basketball	2	4	17,970	1,064	1,222
Topic Classification	News	AGNews	4	9	96,000	12,000	12,000

Table 5: Hyperparameter search spaces for label models and end models.

Model	Parameters	Searched Values
MeTaL	learning rate	1e-5, 1e-4, 1e-3, 1e-2, 1e-1
	weight decaay	1e-5, 1e-4, 1e-3, 1e-2, 1e-1
	training epochs	5, 10, 50, 100, 200
Data Programming	learning rate	1e-5, 5e-5, 1e-4
	weight decaay	1e-5, 1e-4, 1e-3, 1e-2, 1e-1
	training epochs	5, 10, 50, 100, 200
Logistic Regression	learning rate	1e-5, 1e-4, 1e-3, 1e-2, 1e-1
	weight decaay	1e-5, 1e-4, 1e-3, 1e-2, 1e-1
	batch size	32, 128, 512
	training steps	10000
MLP	learning rate	1e-5, 1e-4, 1e-3, 1e-2, 1e-1
	weight decaay	1e-5, 1e-4, 1e-3, 1e-2, 1e-1
	batch size	32, 128, 512
	training steps	10000
	hidden layers	1
	hidden size	100
BERT, RoBERTa	learning rate	2e-5,3e-5,5e-5
	weight decay	1e-4
	batch size	16, 32
	training steps	10000

B All Empirical Results

B.1 Dataset and hyperparameter details

Table 4 is a reproduction of Zhang et al. [42]'s Table 5 for the datasets used in this paper. Zhang et al. [42]'s Table 5 contains more statistics on the labeling functions, including average coverage and accuracy.

Table 5 shows the hyperparameter search spaces for the label models and end models. We used the same search spaces and tuning procedure as Zhang et al. [42] (see their Table 10), choosing the values that obtain the best mean performance on the gold-labeled validation set across five trial runs. As discussed in Section 4, we do *not* re-tune these hyperparameters for $\beta < 1.0$; we used fixed values to show that simply tuning β on its own can improve performance.

B.2 End model performance and β

Figure 6 shows how the end model test performance changes with β for TREC and SemEval datasets and Majority Vote and Dawid-Skene label models. At low coverage fractions, the end model performs

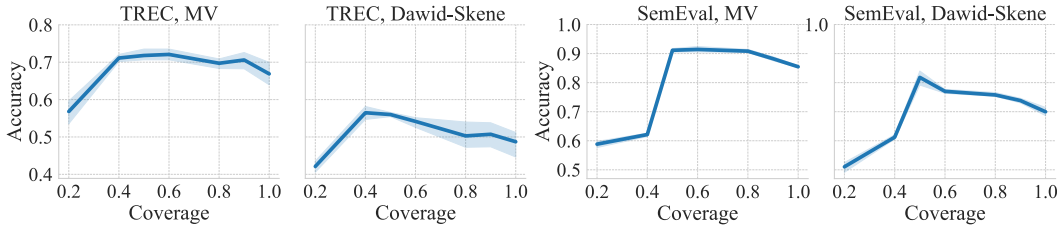


Figure 6: End model performance versus coverage fraction β for TREC and SemEval, Majority Vote and Dawid-Skene.

worse than in the $\beta = 1.0$ case because there is less training data and because the training subsets can be imbalanced (recall that we do not use stratified subset selection, and that the generalization bound in Theorem 1 depends on the *minimum* coverage for each class). At the intermediate coverage fractions, the end model performs *better* than the $\beta = 1.0$ case. An interesting direction for future work is to determine methods for automatically selecting the best value of β.

B.3 Subset selection versus relabeling

Why not *correct* pseudolabels with nearest neighbor? Consider an example x_i whose weak label $\hat{Y}(x_i)$ disagrees with the weak label $\hat{Y}(x_j)$ of most neighbors $j \in N(i)$. This example would get thrown out by the cut statistic selection. Instead of throwing such data points out, we could try to *re-label* them with the majority weak label from the neighbors. However, throwing data out is a more conservative (and hence possibly more robust) approach: Figure 7a shows a simple example where relabeling performs much worse than sub-selection. For the representations studied in this work, relabeling largely fails to improve training set quality and end model performance.

If the weak labels are mostly inaccurate close to the true unknown decision boundary (e.g., on hard examples), relabeling can actually make the training set *worse*. This is also borne out on real empirical examples. Figure 7b shows the weak label accuracy on a relabeled Yelp training set where the β fraction of examples with the *largest* cut statistic score score Z_i—examples with many more cut edges than expected—are relabeled according to the majority vote of their neighbors. Relabeling largely fails to improve over the quality of the $\beta = 1.0$ full training set. However, we note that instead of relabeling, [6] obtained good results using nearest-neighbor to *expand* the pseudolabeled training set by labeling some of the unlabeled examples $\{x_i : \mathbf{\Lambda}(x_i) = \varnothing\}$. If most uncovered examples $\{x : \hat{Y}(x) = \varnothing\}$ are closer to *correctly* pseudolabeled examples than incorrectly labeled ones, this nearest-neighbor *expansion* can improve performance.

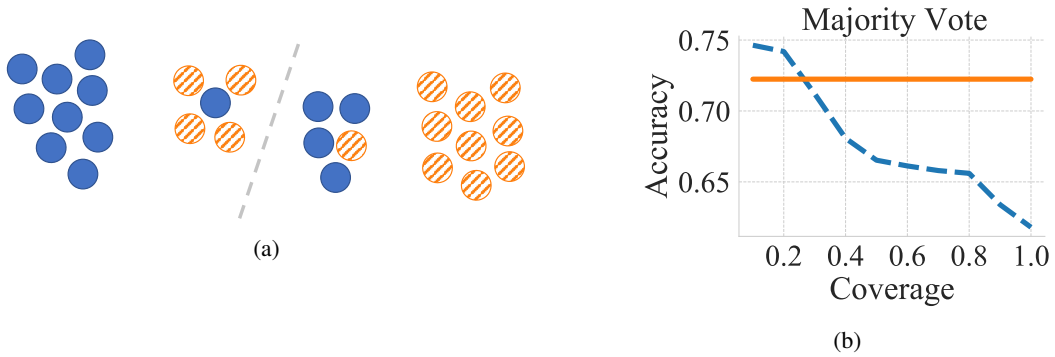
B.4 Additional selection accuracy plots

Figure 8 contain analogous plots to Figure 2 for every dataset in Table 1. These figures compare the quality of the training subsets selected by the cut statistic and entropy scoring.

B.5 Ablation for number of cut statistic neighbors

K value	Test accuracy
5	76.92 (2.57)
10	73.72 (2.59)
20	72.92 (1.31)
40	73.12 (2.39)
80	72.16 (1.89)

This table shows how the results change when varying the number of nearest-neighbors K used in the cut statistic, using majority vote and training a RoBERTa end model on TREC. The performance gain over $\beta = 1.0$ (which obtains 66.28% mean accuracy) is not sensitive to the choice of K. As in all of our results, we re-used hyperparameters from the $\beta = 1.0$ case and chose the best value of $\beta = 1.0$ according to gold-labeled validation performance. The best value for β was stable across all choices of K: $\beta = 0.4$ had the optimal validation performance in all of these experiments. As indicated in the table, better results may be obtained by tuning over K, but our results showed the same value of K obtains good performance across a wide variety of datasets and end models.



(a) (b)

Figure 7: Left: example where subset selection performs better than re-labeling using ϕ. In this example, the true labels Y are recoverable by an unknown linear classifier (the dotted line). The weak labels (solid versus striped) are mostly incorrect close to this unknown decision boundary and always correct farther from the decision boundary. Relabeling the noisy points using nearest-neighbor (e.g., 4-nearest neighbor) actually makes the weak label accuracy *worse*, whereas selecting based on the cut statistic yields a subset of examples with 100% accuracy. Right: relabeling performance on Yelp. Points (β, Y) show the accuracy of the pseudo-labeled training set obtained by relabeling the noisiest β fraction of points (ranked by the cut statistic Z_i) with the majority vote of their neighbors in ϕ (dotted blue), compared to the accuracy when $\beta = 1$ (solid orange). Relabeling largely fails to improve accuracy over the $\beta = 1.0$ case.

Table 6: RoBERTa results using $\beta = 1.0$ (no sub-selection), reported from Zhang et al. [42]

	imdb	**yelp**	**youtube**	**trec**	**semeval**	**chemprot**	**agnews**
MV [42]	85.76 (0.70)	89.91 (1.76)	96.56 (0.86)	66.28 (1.21)	84.00 (0.84)	56.85 (1.91)	86.88 (0.98)
MV (ours)	86.99 (0.55)	88.51 (3.25)	95.84 (1.18)	67.60 (2.38)	85.83 (1.22)	57.06 (1.12)	87.46 (0.53)
DP [42]	86.26 (1.02)	89.59 (2.87)	95.60 (0.80)	72.12 (4.58)	70.57 (0.83)	39.91 (9.33)	86.81 (0.42)
DP (ours)	86.31 (1.53)	88.73 (5.07)	94.08 (1.48)	71.40 (3.30)	71.07 (1.66)	52.52 (0.69)	86.75 (0.24)
DS [42]	84.74 (1.41)	92.30 (1.75)	93.52 (1.39)	48.32 (1.50)	69.67 (1.18)	45.69 (0.86)	87.16 (0.58)
DS (ours)	85.50 (1.68)	92.42 (1.41)	92.48 (1.44)	51.24 (3.50)	70.83 (0.75)	45.61 (2.60)	87.29 (0.40)
FS [42]	86.95 (0.58)	92.08 (2.63)	93.84 (1.57)	30.44 (3.48)	31.83 (0.00)	39.95 (6.50)	86.69 (0.29)
FS (ours)	85.25 (1.96)	92.14 (2.76)	93.52 (2.11)	35.40 (1.32)	31.83 (0.00)	47.23 (1.04)	86.56 (0.55)
MeTaL [42]	84.98 (1.07)	89.08 (3.71)	94.56 (0.65)	60.04 (1.18)	70.73 (0.68)	54.59 (0.77)	87.18 (0.45)
MeTaL (ours)	86.16 (1.13)	88.41 (3.25)	92.40 (1.19)	55.44 (1.08)	59.53 (1.87)	56.74 (0.58)	86.74 (0.60)

B.6 Original WRENCH results

Our $\beta = 1.0$ results closely matched the $\beta = 1.0$ results from Zhang et al. [42], but not in every case, despite using the same hyperparameter search space and tuning scheme for both the label model and the end model. Table 6 shows our results for RoBERTa and $\beta = 1.0$ in line with the same results reported in Zhang et al. [42]. Table 7 compares the performance of our method (i.e., selection with the cut statistic) against the performance of COSINE [41], which performs multiple rounds of self-training on the unlabeled data.

B.7 Combining the Cut Statistic with Weakly-Supervised Self-Training Methods

COSINE. COSINE [41] combines an initial set of pseudolabeled data with a self-training procedure to make better use of the *unlabeled* data that is not covered by weak rules. In each round of self-training, a subset of the data is chosen to use as the training set for the next round by using the confidence score of the trained end model. Instead of using the confidence score, we can instead use the cut statistic to select the training data for each round. This is analogous to switching from the standard self-training algorithm to SETRED [19], which uses the cut statistic to select data for each self-training round. Intuitively, replacing the poorly-calibrated confidence score with the cut statistic

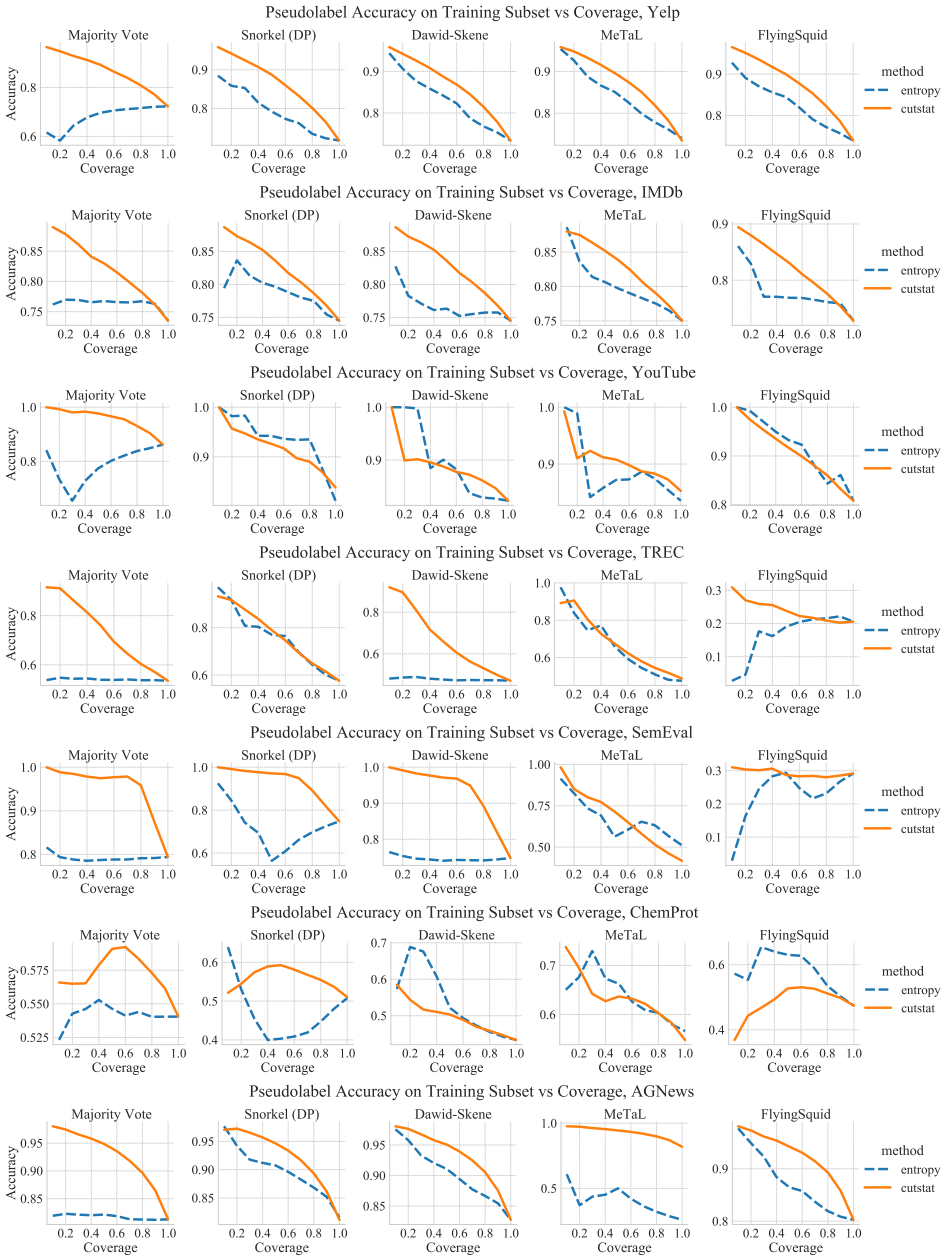


Figure 8: Accuracy of the pseudolabeled training set versus the selection fraction β for five different label models and seven datasets. A pretrained BERT model is used as ϕ for the cut statistic. The accuracy of the weak training labels is better for $\beta < 1$, indicating that sub-selection can select higher-quality training sets. The two curves should always agree at $\beta = 1.0$, but don't always do so for MeTaL due to noise in the MeTaL training procedure.

in each round should lead to higher quality training data and increased performance. Our previous experiments show this is true for the *first* round.

ASTRA. ASTRA [15] is a semi-weakly supervised learning method that uses weakly-labeled data plus a small set of labeled data and a large set of unlabeled data. There are two networks: the *teacher model*, also called the Rule Attention Network (RAN), and the *student model*, which is analogous to our *end model* (BERT, RoBERTa, etc.). Training proceeds in rounds. In the first step, the student model is fine-tuned on the small labeled dataset and used to pseudolabel the unlabeled

Table 7: Cut statistic versus the COSINE method [41]. We report the tuned COSINE performance from [42]. COSINE performs multiple rounds of self-training on the unlabeled data, whereas the cut statistic method performs one round of training on a carefully chosen subset of the weakly-labeled data. Surprisingly, the cut statistic is sometimes competitive with COSINE despite not using any of the unlabeled data and only requiring one round of training. We show in Section B.7 how to combine two appraoches.

End Model	Method	**imdb**	**yelp**	**youtube**	**trec**	**semeval**	**chemprot**	**agnews**
BERT	MV + COSINE	82.98 (0.05)	89.22 (0.05)	98.00 (0.00)	76.56 (0.08)	86.80 (0.46)	58.47 (0.08)	87.03 (0.00)
	MV + cutstat	81.86 (1.36)	89.49 (0.78)	95.60 (0.72)	71.84 (3.00)	92.47 (0.49)	57.47 (1.00)	86.26 (0.43)
	DP + COSINE	84.58 (0.08)	88.44 (0.03)	96.32 (0.16)	78.72 (0.43)	75.77 (1.33)	57.51 (0.02)	86.98 (0.39)
	DP + cutstat	79.07 (2.52)	88.13 (1.46)	93.92 (0.93)	76.76 (1.92)	91.07 (0.90)	55.10 (1.49)	85.89 (0.45)
	DS + COSINE	91.54 (0.54)	90.84 (0.30)	94.16 (0.20)	53.36 (0.29)	72.50 (0.00)	49.65 (0.68)	87.19 (0.00)
	DS + cutstat	80.22 (1.69)	89.04 (1.10)	90.72 (1.27)	57.28 (2.91)	89.07 (1.62)	49.07 (1.48)	86.93 (0.22)
	FS + COSINE	84.40 (0.00)	89.05 (0.07)	94.80 (0.00)	27.60 (0.00)	31.83 (0.00)	48.10 (0.60)	87.16 (0.16)
	FS + cutstat	80.85 (1.50)	88.75 (1.13)	91.04 (1.23)	33.84 (3.17)	31.83 (0.00)	48.65 (0.99)	85.90 (0.39)
	MeTaL + COSINE	83.47 (0.12)	89.76 (0.00)	94.88 (0.53)	61.80 (0.00)	79.20 (2.33)	55.46 (0.12)	87.26 (0.02)
	MeTaL + cutstat	81.49 (1.51)	88.41 (1.19)	92.64 (0.41)	63.80 (2.28)	65.23 (0.91)	58.33 (0.81)	86.16 (0.48)
RoBERTa	MV + COSINE	88.22 (0.22)	94.23 (0.20)	97.60 (0.00)	77.96 (0.34)	86.20 (0.07)	59.43 (0.00)	88.15 (0.30)
	MV + *cutstat*	86.69 (0.75)	95.19 (0.23)	96.00 (1.10)	72.92 (1.31)	92.07 (0.80)	59.05 (0.56)	88.01 (0.47)
	DP + COSINE	87.91 (0.15)	94.09 (0.06)	96.80 (0.00)	82.36 (0.08)	75.17 (0.95)	52.86 (0.06)	87.53 (0.03)
	DP + *cutstat*	86.46 (1.82)	93.95 (0.93)	93.04 (1.30)	76.84 (4.09)	86.07 (1.82)	56.43 (1.37)	87.76 (0.17)
	DS + COSINE	88.01 (0.56)	94.19 (0.18)	96.24 (0.41)	59.40 (0.42)	71.70 (0.07)	46.75 (0.27)	88.20 (0.11)
	DS + *cutstat*	86.14 (0.60)	93.81 (0.69)	93.84 (0.70)	58.48 (2.75)	81.67 (1.33)	52.93 (1.67)	88.35 (0.22)
	FS + COSINE	88.48 (0.00)	95.33 (0.06)	96.80 (0.00)	33.80 (0.00)	31.83 (0.00)	39.89 (0.00)	87.23 (0.00)
	FS + *cutstat*	87.71 (0.76)	94.50 (0.74)	95.84 (0.54)	38.16 (0.43)	31.83 (0.00)	50.55 (1.05)	87.49 (0.13)
	MeTaL + COSINE	86.46 (0.11)	93.11 (0.01)	97.04 (0.20)	71.64 (0.59)	70.90 (0.08)	53.32 (0.19)	87.85 (0.02)
	MeTaL + *cutstat*	87.46 (0.65)	94.03 (0.53)	93.84 (1.38)	69.72 (2.39)	66.70 (0.90)	57.40 (0.98)	88.40 (0.38)

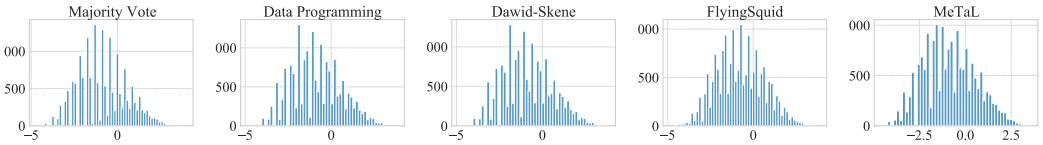


Figure 9: Histograms for the cut statistic score Z_i on IMDb using BERT as ϕ.

dataset. Next, the teacher is trained on the weakly-labeled training data plus pseudolabeled data from the gold-fine-tuned student. The teacher then pseudolabels the unlabeled data using a learned instance-specific weighting procedure, so that high-quality examples are upweighted. Finally, the student model is trained on this data.

We can insert the cut statistic in multiple parts of this procedure. First, in each round, ASTRA trains the teacher model on *all* of the pseudolabeled data from the gold-fine-tuned student. Instead, we can use the cut statistic to select a high-quality subset of this data for the teacher model to train on. Second, the student model is trained on a subset of the data selected by the teacher model; we can further filter this subset with the cut statistic. Applying the cut statistic in this step is somewhat less necessary, since ASTRA already has a (soft) instance-specific selection procedure built-in.

Table 8 shows plain ASTRA versus ASTRA + cutstat on SemEval and TREC using a RoBERTa-base end model. Standard deviations are reported across five random seeds for choosing the labeled subset. Following the best constant β from Section 4, we set $\beta = 0.6$ for the first round of training, then increase by 0.1 in each round to use more of the unlabeled data each time. So β for the t-th round of ASTRA is $\min(1, 0.6 + 0.1t)$, $t \in \{0, \ldots, 24\}$. Following Karamanolakis et al. [15], we train ASTRA for up to 25 iterations using a patience of 3 iterations. In each step, the model checkpoint with best validation performance is kept. We did not perform hyperparameter tuning on the end model parameters and used a fixed learning rate of 2e-5 and batch size 128. The cut statistic improves the ASTRA performance for nearly every labeled data size despite us not tuning β on the validation set. Tuning β on the validation set, as in Table 1, would likely result in even better performance gains.

Table 8: Combining the cut statistic with ASTRA [15] boosts performance by selecting a higher-quality set of training data for the teacher model in each round. These results use fixed end-model hyperparameters and a fixed choice for the cut statistic fraction β in each round.

Method	|Labeled set|	**trec**	**semeval**
ASTRA	10	65.60 (5.19)	82.70 (3.04)
+ *cutstat*	10	67.40 (5.78)	91.10 (0.92)
ASTRA	20	74.40 (3.35)	86.53 (1.17)
+ *cutstat*	20	75.04 (1.63)	90.27 (2.09)
ASTRA	40	85.72 (1.32)	87.60 (1.22)
+ *cutstat*	40	84.52 (3.17)	91.20 (1.11)

C Cut statistic code

For simplicity in computing the graph G for the cut statistic, we provide code where the neighborhoods sets $N(i)$ are not necessarily symmetric, so $i \in N(j) \not\Rightarrow j \in N(i)$. This does not change the empirical performance of the algorithm.

```
import torch

def get_conf_inds(labels, features, coverage, device='cuda'):
    features = torch.FloatTensor(features).to(device)
    labels = torch.LongTensor(labels).to(device)

    # move to CPU for memory issues on large dset
    pairwise_dists = torch.cdist(features, features, p=2).to('cpu')

    N = labels.shape[0]
    dists_sorted = torch.argsort(pairwise_dists)
    neighbors = dists_sorted[:,:20]
    dists_nn = pairwise_dists[torch.arange(N)[:,None], neighbors]
    weights = 1/(1 + dists_nn)

    neighbors = neighbors.to(device)
    dists_nn = dists_nn.to(device)
    weights = weights.to(device)

    cut_vals = (labels[:,None] != labels[None,:]).long()
    cut_neighbors = cut_vals[torch.arange(N)[:,None], neighbors]
    Jp = (weights * cut_neighbors).sum(dim=1)

    weak_counts = torch.bincount(labels)
    weak_pct = weak_counts / weak_counts.sum()

    prior_probs = weak_pct[labels]
    mu_vals = (1-prior_probs) * weights.sum(dim=1)
    sigma_vals = prior_probs * (1-prior_probs) * torch.pow(weights, 2).sum(dim=1)
    sigma_vals = torch.sqrt(sigma_vals)
    normalized = (Jp - mu_vals) / sigma_vals

    normalized = normalized.cpu()
    inds_sorted = torch.argsort(normalized)

    N_select = int(coverage * N)
    conf_inds = inds_sorted[:N_select]
    conf_inds = list(set(conf_inds.tolist()))
    return conf_inds
```