Snuba: Automating Weak Supervision to Label Training Data

Paroma Varma
Stanford University
paroma@stanford.edu

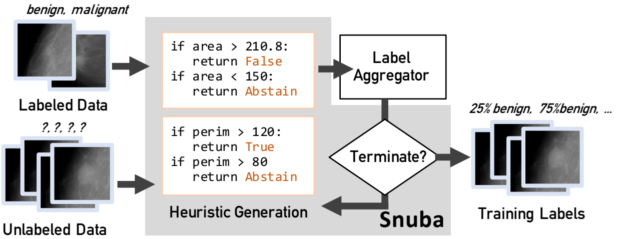
Christopher Ré
Stanford University
chrismre@cs.stanford.edu

ABSTRACT

As deep learning models are applied to increasingly diverse problems, a key bottleneck is gathering enough high-quality training labels tailored to each task. Users therefore turn to *weak supervision*, relying on imperfect sources of labels like pattern matching and user-defined heuristics. Unfortunately, users have to design these sources *for each task*. This process can be time consuming and expensive: domain experts often perform repetitive steps like guessing optimal numerical thresholds and developing informative text patterns. To address these challenges, we present Snuba, a system to automatically generate heuristics using a small labeled dataset to assign training labels to a large, unlabeled dataset in the weak supervision setting. Snuba generates heuristics that each labels the subset of the data it is accurate for, and iteratively repeats this process until the heuristics together label a large portion of the unlabeled data. We develop a statistical measure that guarantees the iterative process will automatically terminate before it degrades training label quality. Snuba automatically generates heuristics in under five minutes and performs up to 9.74 F1 points better than the best known user-defined heuristics developed over many days. In collaborations with users at research labs, Stanford Hospital, and on open source datasets, Snuba outperforms other automated approaches like semi-supervised learning by up to 14.35 F1 points.

1. INTRODUCTION

The success of machine learning for tasks like image recognition and natural language processing [12, 15] has ignited interest in using similar techniques for a variety of tasks. However, gathering enough training labels is a major bottleneck in applying machine learning to new tasks. In response, there has been a shift towards relying on *weak supervision*, or methods that can assign noisy training labels to unlabeled data, like crowdsourcing [9, 23, 61], distant supervision [8, 33], and user-defined heuristics [39, 40, 52]. Over the past few years, we have been part of the broader effort to enhance methods based on user-defined heuristics to extend their applicability to text, image, and video data for tasks in computer vision, medical imaging, bioinformatics and knowledge base construction [40, 4, 52].



```
if area > 210.8:
   return False
if area < 150:
   return Abstain
```

```
if perim > 120:
   return True
if perim > 80
   return Abstain
```

Figure 1: Snuba uses a small labeled and a large unlabeled dataset to iteratively generate heuristics. It uses existing label aggregators to assign training labels to the large dataset.

Through our engagements with users at large companies, we find that experts spend a significant amount of time *designing* these weak supervision sources. As deep learning techniques are adopted for unconventional tasks like analyzing codebases and now commodity tasks like driving marketing campaigns, the few domain experts with required knowledge to write heuristics cannot reasonably keep up with the demand for several specialized, labeled training datasets. Even machine learning experts, such as researchers at the computer vision lab at Stanford, are impeded by the need to crowdsource labels before even starting to build models for novel visual prediction tasks [26, 24]. This raises an important question: *can we make weak supervision techniques easier to adopt by automating the process of generating heuristics that assign training labels to unlabeled data?*

The key challenge in automating weak supervision lies in replacing the human reasoning that drives heuristic development. In our collaborations with users with varying levels of machine learning expertise, we noticed that the process to develop these weak supervision sources can be fairly repetitive. For example, radiologists at the Stanford Hospital and Clinics have to guess the correct threshold for each heuristic that uses a geometric property of a tumor to determine if it is malignant (example shown in Figure 1). We instead take advantage of *a small, labeled dataset to automatically generate noisy heuristics.* Though the labeled dataset is too small to train an end model, it has enough information to generate heuristics that can assign noisy labels to a large, unlabeled dataset and improve end model performance by up to 12.12 F1 points. To aggregate labels from these heuristics, we improve over majority vote by relying on existing factor graph-based statistical techniques in weak supervision that can model the noise in and correlation among these heuristics [40, 4, 52, 2, 42, 50]. However, these techniques were intended to work with user-designed labeling sources and therefore have limits on how robust they are. Automatically generated heuristics can be noisier than what these models can account for and introduce the following challenges:

Accuracy. Users tend to develop heuristics that assign accurate labels to a *subset* of the unlabeled data. An automated method has to properly model this trade-off between accuracy and coverage for each heuristic based only on the small, labeled dataset. Empirically, we find that generating heuristics that *each* labels all the datapoints can degrade end model performance by up to 20.69 F1 points.

Diversity. Since each heuristic has limited coverage, users develop multiple heuristics that each labels a different subset to ensure a large portion of the unlabeled data receives a label. In an automated approach, we could mimic this by maximizing the number of unlabeled datapoints the heuristics label as a set. However, this approach can select heuristics that cover a large portion of the data but have poor performance. There is a need to account for both the diversity and performance of the heuristics *as a set.* Empirically, balancing both aspects improves end model performance by up to 18.20 F1 points compared to selecting the heuristic set that labels the most datapoints.

Termination Condition. Users stop generating heuristics when they have exhausted their domain knowledge. An automated method, however, can continue to generate heuristics that deteriorate the overall quality of the training labels assigned to the unlabeled data, such as heuristics that are worse than random for the unlabeled data. Not accounting for performance on the unlabeled dataset can affect end model performance by up to 7.09 F1 points.

Our Approach. To address the challenges above, we introduce Snuba, an automated system that takes as input a small labeled and a large unlabeled dataset and outputs probabilistic training labels for the unlabeled data, as shown in Figure 1. These labels can be used to train a downstream machine learning model of choice, which can operate over the raw data and generalize beyond the heuristics Snuba generates to label any datapoint. Users from research labs, hospitals and industry helped us design Snuba such that it outperforms user-defined heuristics and crowdsourced labels by up to 9.74 F1 points and 13.80 F1 points in terms of end model performance. Snuba maintains a set of heuristics that is used to assign labels to the unlabeled dataset. At each iteration, Snuba appends a new heuristic to this set after going through the following components:

Synthesizer for Accuracy. To address the trade-off between the accuracy and coverage of each heuristic, the *synthesizer* (Section 3.1) generates heuristics based on the labeled set and adjusts its labeling pattern to abstain if the heuristic has low confidence. The synthesizer relies on a small number of primitives, or features of the data, to generate multiple, simple models like decision trees, which improves over fitting a single model over primitives by 12.12 F1 points. These primitives are user-defined and part of open source libraries [36, 51] and data models in existing weak supervision frameworks [39, 59]. Primitives examples in our evaluation include bag-of-words for text and bounding box attributes for images.

Pruner for Diversity. To ensure that the set of heuristics is diverse and assigns high-quality labels to a large portion of the unlabeled data, the *pruner* (Section 3.2) ranks the heuristics the synthesizer generates by the weighted average of their performance on the labeled set and coverage on the unlabeled set. It selects the best heuristic at each iteration and adds it to the collection of existing heuristics. This method performs up to 6.57 F1 points better than ranking heuristics by performance only.

Verifier to Determine Termination Condition. The verifier uses existing statistical techniques to aggregate labels from the heuristics into probabilistic labels for the unlabeled datapoints [52, 40, 4]. However, the automated heuristic generation process can surpass the noise levels to which these techniques are robust to and degrade end model performance by up to 7.09 F1 points. We develop a statistical measure that uses the small, labeled set to determine whether the noise in the generated heuristics is below the threshold these techniques can handle (Section 4).

Contribution Summary. We describe Snuba, a system to automatically generate heuristics using a small labeled dataset to assign training labels to a large, unlabeled dataset in the weak supervision setting. A summary of our contributions are as follows:

- We describe the system architecture, the iterative process of generating heuristics, and the optimizers used in the three components (Section 3). We also show that our automated optimizers can affect end model performance by up to 20.69 F1 points (Section 5).
- We present a theoretical guarantee that Snuba will terminate the iterative process *before* the noise in heuristics surpasses the threshold to which statistical techniques are robust (Section 4). This theoretical result translates to improving end model performance by up to 7.09 F1 points compared to generating as many heuristics as possible (Section 5).
- We evaluate our system in Section 5 by using Snuba labels to train downstream models, which generalize beyond the heuristics Snuba generates. We report on collaborations with Stanford Hospital and Stanford Computer Vision Lab, analyzing text, image, and multi-modal data. We show that heuristics from Snuba can improve over hand-crafted heuristics developed over several days by up to 9.74 F1 points. We compare to automated methods like semi-supervised learning, which Snuba outperforms by up to 14.35 F1 points.

2. SYSTEM OVERVIEW

We describe the input and output for Snuba, introduce notation used in the rest of paper, and summarize statistical techniques Snuba relies on to learn heuristic accuracies.

2.1 Input and Output Data

Input Data. The input to Snuba is a labeled dataset O_L with N_L datapoints and an unlabeled dataset O_U with N_U datapoints. Each datapoint is defined by its associated *primitives*, or characteristics of the data, and a label. The inputs to the system can be represented as

$$\{x_i, y_i^*\}_{i=1}^{N_L}, \text{ (for the labeled set } O_L\text{), and}$$

$$\{x_i\}_{i=1}^{N_U}, \text{ (for the unlabeled set } O_U\text{)}$$

where $x_i \in \mathbb{R}^D$, y^* represent the primitives for a particular object and the true label, respectively. For convenience, we focus on the binary classification setting, in which $y^* \in \{-1, 1\}$ and discuss the multi-class setting in Section 3.4.

The primitives for each datapoint $x_i \in \mathbb{R}^D$ can be viewed as features of the data — examples include numerical features such as area or perimeter of a tumor for image data. or one-hot vectors for the bag of words representation for text data. For our collaborators using Snuba, these primitives are usually part of data models in existing weak supervision systems and open source libraries [39, 59, 36, 51]. For example, Scikit-image includes functions to extract geometric properties from segmented images [51]. In our evaluation,

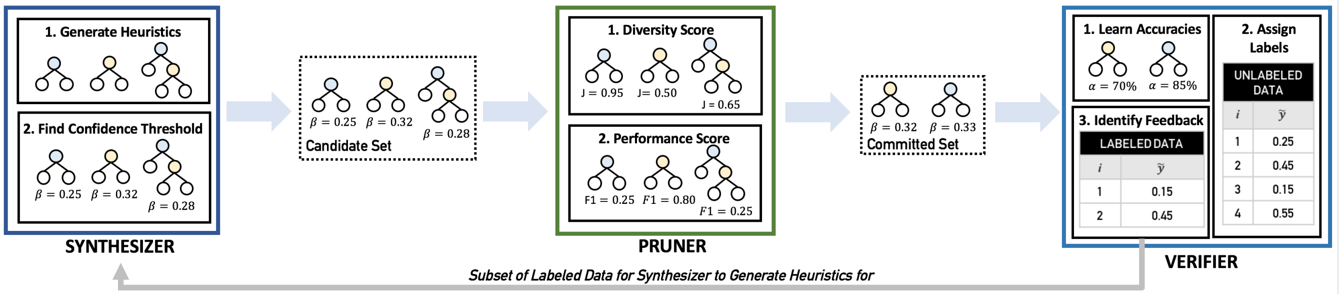


Figure 2: An overview of the Snuba system. (1) The synthesizer generates a candidate set of heuristics based on the labeled dataset. (2) The pruner selects the heuristic from the candidate set to add to the committed set. (3) The verifier learns heuristic accuracies and passes appropriate feedback to the synthesizer to continue the iterative process.

we do not allow users to extend the set of primitives beyond those present in these data models and libraries, though they could be extended in principle.

Output Data. Snuba outputs a probabilistic *training label* $\tilde{y} = P[y^* = 1] \in [0, 1]$ for each datapoint in the unlabeled set, a weighted combination of labels from different heuristics. Since Snuba only relies on information about the data encoded in the primitives and does not take advantage of a complete representation of the data, it is advantageous to train a downstream model that has access to the entire input data space using probabilistic labels from Snuba as training labels. These downstream models, such as a convolutional neural network (CNN) [27] for image classification or a long-short term memory (LSTM) architecture [20] for natural language processing tasks, can operate over the raw data (e.g., the radiology image of a tumor from Figure 1 or complete sentences). We discuss specific end models and show that the end model generalizes beyond the heuristics by improving recall by up to 61.54 points in Section 5.

2.2 Learning Heuristic Accuracies

Each heuristic Snuba generates relies on one or more primitives and outputs a binary label or abstains for each datapoint in the unlabeled dataset (Section 3.1). A single bad (but prolific) voter can compromise majority vote, which weights all heuristics equally [40]. Snuba instead relies on existing statistical techniques (Section 4) that can learn the accuracies of these heuristics without using ground truth labels and assign probabilistic labels to the unlabeled dataset accordingly [40, 4, 52, 2, 42, 50]. We treat these statistical techniques as black-box methods that learns heuristic accuracies and refer to them as *label aggregators* since they combine the labels the heuristics assign to generate a single probabilistic label per datapoint. However, since Snuba can generate heuristics that are much noisier than the label aggregator can handle, it has to determine the conditions under which the aggregator operates successfully (Section 4).

3. THE SNUBA ARCHITECTURE

The Snuba process is iterative and generates a new heuristic *specialized* to the subset of the data that did not receive high confidence labels from the existing set of heuristics at each iteration. As shown in Figure 2, the three components of Snuba are the synthesizer (Section 3.1) that generates a candidate set of heuristics, a pruner (Section 3.2) that selects a heuristic to add to an existing committed set of heuristics, and a verifier (Section 3.3) that assigns probabilistic labels to the data and passes the subset of the labeled data that received low confidence labels to the synthesizer for the next iteration. This process is repeated until the subset the verifier passes to the synthesizer is empty, or the verifier determines that the conditions for the label aggregator to operate successfully are violated (Section 4).

3.1 Synthesizer

The Snuba synthesizer takes as input the labeled set, or a subset of the labeled set after the first iteration, and outputs a *candidate* set of heuristics (Figure 2). First, we describe how the heuristics are generated using the labeled dataset and the different models the heuristic can be based on. Then, we describe how the labeling pattern of the heuristics are adjusted to assign labels to only a subset of the unlabeled dataset. Finally, we explore the trade-offs between accuracy and coverage by comparing heuristics Snuba generated to other automated methods.

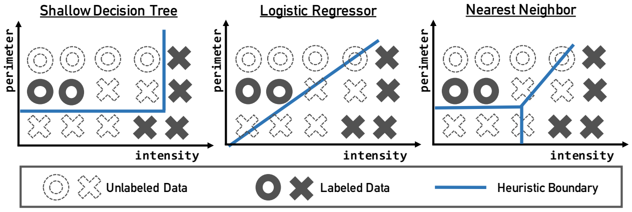


Figure 3: Heuristic models and associated boundaries.

3.1.1 Heuristic Generation

In Snuba, users can select the model they want to base their heuristics on given the heuristic h follows the input-output form: $h(x'_i) \rightarrow P[y^*_i = 1] \in [0, 1]$ where $x'_i \in \mathbb{R}^{D'}$ is a subset of primitives, $D' \leq D$ is the number of primitives in this subset, and $P[y^*_i = 1]$ is a probabilistic label.

Choosing subsets of size D' from the primitives translates to selecting D' rows from X, as shown in function GenComb in Algorithm 1. For D primitives, there will be a total of $\binom{D}{D'}$ distinct primitive subsets of size D'. These subsets of primitives can be representative of a few specific words if primitives are generated using a bag-of words model while a subset of bounding box attribute primitives could represent the x,y-coordinates of the bounding box. The synthesizer generates a heuristic for each possible combination of 1 to

D primitives, resulting in $\sum_{D'=1}^{D}\binom{D}{D'} = 2^D - 1$ total heuristics per iteration of Snuba. We find that $D' < 4$ for most real-world tasks, resulting in a maximum runtime of 14.45 minutes for Snuba on a single thread (Section 5).

Heuristic Models. In this paper, we focus on heuristics that are based on classification models that take as input one or more primitives and assign probabilistic labels $P[y_i^* = 1] \in [0, 1]$ to the unlabeled datapoints. We consider three different ways of generating heuristics given a subset of the labeled data and a subset of primitives (Figure 3).

- **Decision Stumps** mimic the nested threshold-based heuristics that users commonly write. To maintain the simplicity of the heuristic, we limit the depth of each tree to the number of primitives the heuristic depends on. The confidence each unlabeled datapoint receives is the fraction of labeled datapoints that belong to the same leaf.
- **Logistic Regressor** allows the heuristic to learn a single linear decision boundary. As shown in Figure 3, it does not have to be parallel to the primitive axes, unlike decision trees. The confidence for an unlabeled datapoint is determined by the sigmoid function, whose parameters are learned using the labeled datapoints.
- **K-Nearest Neighbor** is based on a kd-tree implementation of nearest neighbor and can lead to complex decision boundaries that neither decision trees nor logistic regressors can capture. Unlike the previous heuristic models, it does not learn a parameter per primitive, but instead relies on the distribution of the labeled datapoints to decide the decision boundaries. The confidence for a unlabeled datapoint is a function of its distance from labeled datapoints.

The user can replace the heuristic model with another function of choice as long as it follows the input-output criteria described earlier in this section. For example, decision trees that rely on bag-of-words primitives represent heuristics that check whether a particular word, represented as a primitive, exists or not.

3.1.2 Tuning Threshold for Abstains

We can improve performance of heuristics by modeling the trade-off between heuristic accuracy and coverage. Snuba forces heuristics to only assign labels to datapoints they have high confidence for and abstain for the rest. To measure confidences, Snuba relies on the probabilistic label $P[y_i^* = 1]$ that each heuristic model assigns to a datapoint. We define datapoints that heuristics have low confidence for as the points where $|P[y_i^* = 1] - 0.5| \leq \beta$, $\beta \in (0, 0.5)$. For each heuristic, Snuba selects a threshold β that determines when a heuristic assigns a label, $\hat{y} \in \{-1, 1\}$ and when it abstains, $\hat{y} = 0$. The relation between β and $\hat{y}$ can be defined as:

$$\hat{y}_i = \begin{cases} 1 \text{ if} & P[\hat{y}_i = 1] \geq 0.5 + \beta \\ 0 \text{ if} & |P[\hat{y}_i = 1] - 0.5| < \beta \\ -1 \text{ if} & P[\hat{y}_i = 1] \leq 0.5 - \beta \end{cases}$$

To choose the best threshold β, we need a metric that models the trade-offs between coverage and accuracy. We calculate the precision and recall of the heuristics on the *labeled set* with N_L datapoints as a proxy for their performance on the unlabeled dataset. We define these metrics below:

Algorithm 1: Snuba Synthesis Procedure

```
function GenerateHeuristics $(f, X, y^*)$
  Input: Heuristic model $f \in \{DT, LR, NN\}$, Primitive
         matrix $X \in \mathbb{R}^{D \times N_L}$, Labels $y^* \in \{-1, 1\}^{N_L}$
  Output: Candidate set of heuristics $H$, Primitive
          combinations $X_{comb}$
$H, X_{comb} = []$  for $D' = 1...D$ do
    //generate primitive combinations of size $D'$
    $\text{idx}_{comb} = \text{GenComb}(X, D')$
    for $i = 1...len(idx_{comb})$ do
        $X' = X[\text{idx}_{comb}, :]$
        $h = f(X', y^*)$
        $y_{prob} = \text{predictProb}(h, X')$
        $\beta = \text{FindBeta}(y_{prob}, y^*)$
        $H = H \cup \{(h, \beta)\}$
        $X_{comb} = X_{comb} \cup X'$
    end
end
return $H, X_{comb}$

function FindBeta $(y_{prob}, y^*)$
betaList = $[0, 0.05, \ldots 0.5]$
for $j$ in len(betaList) do
    $beta = \text{betaList}[j]$
    $F1[j] = \text{calcF1}(y^*, y_{prob}, beta)$
end
return betaList[argmax($F1$)]

function GenComb $(X, D')$
//get all D' length subsequences from range(D)
return all subsets of size $D'$ from $D$
```

- **Precision (P)** the fraction of correctly labeled points over the total points labeled, $\frac{\sum_{i=1}^{N_L} \mathbb{1}(\hat{y}_i = y_i^*)}{\sum_{i=1}^{N_L} \mathbb{1}(\hat{y}_i \neq 0)}$
- **Recall (R)** the fraction of correctly labeled points over the total number of points, $\frac{\sum_{i=1}^{N_L} \mathbb{1}(\hat{y}_i = y_i^*)}{N_L}$
- **F1 Score** the harmonic mean of P and R, $2\frac{P \times R}{P + R}$

To balance precision and recall, the Snuba synthesizer selects β for each heuristic that maximizes the F1 score on the labeled dataset O_L (Algorithm 1). The synthesizer iterates through (default 10) equally spaced values in $\beta \in (0, 0.5)$, calculates the F1 score the heuristic achieves, and selects the β that maximizes F1 score. In case of ties, the synthesizer chooses the lower β value for higher coverage. We find selecting β based on F1 score outperforms a constant β by up to 5.30 F1 points (Section 5).

As an example, if the synthesizer uses a decision tree as the heuristic model, it trains a normal decision tree on the small labeled dataset and learns appropriate parameters for a specific subset of primitives (e.g., D = 2 means two primitives, or two rows of X in Algorithm 1) to decide on a label. Then, the synthesizer learns β, which adjusts these decision tree thresholds to abstain for low-confidence datapoints. This adjusted decision tree is then added as a heuristic to the candidate set, and the process is repeated for different subsets of primitives as inputs to the decision tree.

3.1.3 Synthesizer Tradeoffs

We explore the trade-offs that result from allowing the heuristics to abstain in terms of the effect on end model performance. We compare to automated baseline methods (more details in Section 5.1) that assign labels *to the entire unlabeled dataset*. We generate a synthetic experiment (Figure 4) using one of the datasets from our evaluation, the Visual Genome dataset [26] (more details in Section 5.1). To study how Snuba performs given varying amounts of *unlabeled* data, we set up the following simulation: given $N_L = 100$ labeled datapoints, we varied the amount of unlabeled data available to Snuba from $N_U = 100$ to $N_U = 500$. Each of the methods assigned training labels to the unlabeled dataset, and this dataset was used to fine-tune the last layer of GoogLeNet [49].

$N_L \approx N_U$ Case: Since Snuba only labels a portion of the unlabeled data, the end model has fewer training labels to learn from compared to the other methods that do not abstain. Since the unlabeled set is small in this situation ($N_L = N_U = 100$), the baseline methods have better end model performance.

$N_L << N_U$ Case: Heuristics Snuba generates continue to only assign labels with high confidence, leading to a smaller labeled training set than other methods, but high quality training labels for that portion. This is promising for machine learning applications in which the bottleneck lies in *gathering enough training labels*, while unlabeled data is readily available. Semi-supervised learning also performs better as the amount of unlabeled data increases; however, it still performs worse than Snuba when the amount of unlabeled data is more than 3× larger than labeled data since semi-supervised methods do not abstain. Snuba also outperforms these baseline methods when the unlabeled data is between 2× to 1000× as much as labeled data (Section 5).

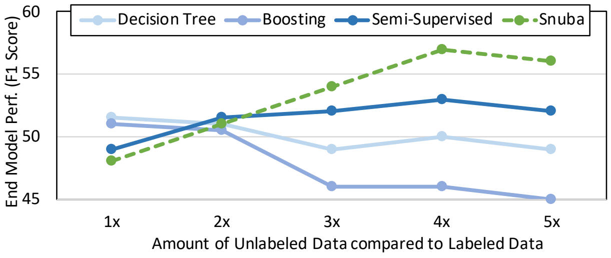


Figure 4: Linear performance increase of end model trained on labels from Snuba w.r.t. unlabeled data.

3.2 Pruner

The pruner takes as input the candidate heuristics from the synthesizer and selects a heuristic to add to the committed set of heuristics (Figure 2). We want the heuristics in the committed set to be diverse in terms of the datapoints in the unlabeled set they label, but also ensure that it performs well for the datapoints it labels in the labeled dataset.

A diverse heuristic is defined as one that labels points that have never received a label from any other heuristic. Therefore, we want to be able to maximize the *dissimilarity* between the set of datapoints a heuristic labels and the set of datapoints that previous heuristics in the committed set have already labeled. Let $n_j \in \{0,1\}^{N_U}$ represent whether heuristic j from the candidate set has assigned labels to the datapoints in the unlabeled set. Let $n \in \{0,1\}^{N_U}$ represent whether any heuristic from the committed set has assigned a label to the datapoints in the unlabeled set. To measure the distance between these two vectors, we rely on the Jaccard distance metric [21], the complement of Jaccard similarity, as a standard measure of similarity between sets. For a particular heuristic h_j in the candidate set, the generalized Jaccard distance is defined as:

$$J_j = 1 - \frac{n_j \cap n}{n_j \cup n}$$

Algorithm 2: Snuba Pruning Procedure

```
function SelectBestHeuristic $(H, H_C, y^*, X_L, X_U, n, w)$
Input: Candidate $(H)$ and committed set $(H_C)$ of
       heuristics, Labels $y^* \in \{-1,1\}^{N_L}$, Primitives
       $X_L \in \mathbb{R}^{D \times N_L}$, $X_U \in \mathbb{R}^{D \times N_U}$, If label assigned
       $n \in \{0,1\}^{N_U}$, Weight for F1 $w \in [0, 0.5]$
Output: Best heuristic in candidate set, $h_{best} \in H$
$h_{best}$ = None
bestScore = 0
for $h_i \in H$ do
    $\hat{y}^i_L = applyHeuristic(h_i, X_L)$
    $f_{score} = calcF1(\hat{y}^i_L, y^*)$

    $\hat{y}^i_U = applyHeuristic(h_i, X_U)$
    $j_{score} = calcJaccard(\hat{y}^i_U, n)$

    if $w(j_{score} + f_{score}) \geq bestScore$ then
        $h_{best} = h_i$
        //default w=0.5
        bestScore = $(1 - w) * j_{score} + w * f_{score}$
    end
end
return $h_{best}$
```

To measure performance on the labeled dataset, Snuba uses the F1 score of each heuristic in the candidate set, as defined in the previous section. As the final metric to rank heuristics, the pruner uses a weighted average of the Jaccard distance and F1 score and selects the highest ranking heuristic from the candidate set and adds it to the committed set of heuristics. This process is described in Algorithm 2. For our experiments, we use both $w = 0.5$ for a simple average and $w = \frac{\mathbb{1}^T n}{N_U}$ (percentage of unlabeled set with at least one label). The latter weights the F1 score more as coverage of the unlabeled dataset increases. We find that considering both performance on the labeled set and diversity on the unlabeled set improves over only considering diversity by up to 18.20 F1 points and over only considering performance by up to 6.57 F1 points in Section 5.

3.3 Verifier

The verifier uses the label aggregator (Section 4) to learn accuracies of the heuristics in the committed set without any ground truth labels to produce a single, probabilistic training label for each datapoint in the unlabeled dataset.

These probabilistic labels also represent how *confident* the label aggregator is about the assigned label. Datapoints that have not received a single label from heuristics in the committed set will have a probabilistic label $P[y^* = 1] = 0.5$, equal chance of belonging to either class. $P[y^* = 1]$ close to 0.5 represent datapoints with low confidence, which can result from scenarios with low accuracy heuristics labeling

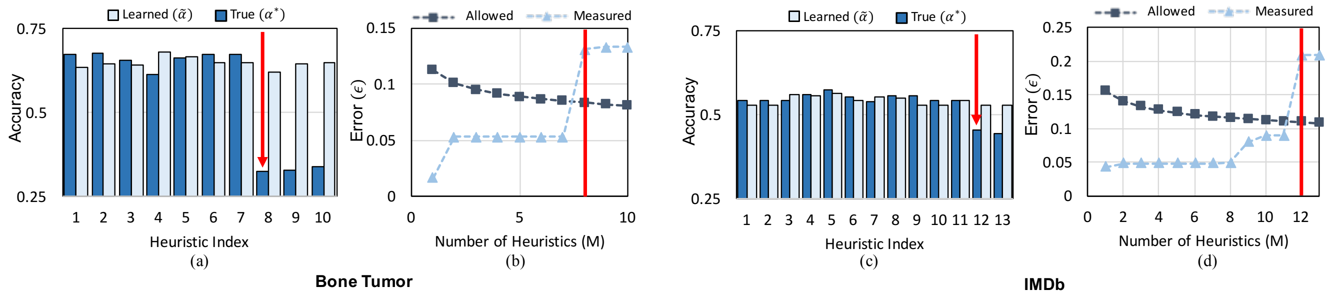


Figure 5: (a,c) show the learned and true accuracies of the committed set of heuristics at the last iteration. (b,d) show the allowed error and the measured error between learned and empirical accuracies across all iterations. The marked heuristic in each figure shows Snuba successfully stops generating heuristics when the new heuristic's *true accuracy* is worse than random.

Algorithm 3: Snuba Verifier Procedure

```
function FindFeedback $(H_C, y^*, X_L, X_U)$
Input: Committed set of heuristics $H_C, H[i] = h(X_i')$,
       Labels $y^* \in \{-1, 1\}^{N_L}$, Primitives
       $X_L \in \mathbb{R}^{D \times N_L}$, $X_U \in \mathbb{R}^{D \times N_U}$
Output: Subset of labeled set, $O_L'$, Prob. labels, $\tilde{y}_U$
$\tilde{\alpha}$ = learnAcc $(H_C, X_U)$
$\hat{\alpha}$ = calcAcc $(H_C, X_L, y^*)$
$\tilde{y}_U$ = calcLabels $(\tilde{\alpha}, X_U)$
$\tilde{y}_L$ = calcLabels $(\tilde{\alpha}, X_L)$
$\epsilon = findEps(N_U, M)$
$\nu = findNu(M)$
if $||\tilde{\alpha} - \hat{\alpha}||_\infty \geq \epsilon$ then
    return $O_L' = \emptyset$, $\tilde{y}_U$
end
else
    return $o_i \in O_L'$ if $|\tilde{y}_L^i - 0.5| \leq \nu$, $\tilde{y}_U$
end
```

that datapoint, or multiple heuristics with similar accuracies disagreeing on the label for that datapoint. Since Snuba generates a new heuristic at each iteration, we want the new heuristic to assign labels to the subset that currently has low confidence labels. Snuba identifies datapoints *in the labeled set* that receive low confidence labels from the label aggregator. It passes this subset to the synthesizer with the assumption that similar datapoints in the unlabeled dataset would have also received low confidence labels (Algorithm 3).

Formally, we define low confidence labels as $|\tilde{y}_i - 0.5| \leq \nu$ where $\tilde{y}$ is the probabilistic label assigned by the label aggregator and $\nu = \frac{1}{2} - \frac{1}{(M+1)^\eta}$ where the $\eta > 1$ parameter (default $\eta = \frac{3}{2}$) controls the rate at which the definition of low confidence changes with number of heuristics in the committed set (M). As the number of heuristics increases, we expect that fewer datapoints will have confidences near 0.5 and adjust what is considered low confidence accordingly. We also compare to a weighted feedback approach in which the weights are the inverse of the label confidence ($w_v = \frac{1}{2} - |\tilde{y} - \frac{1}{2}|$) normalized across all datapoints.

The iterative process terminates if: (1) the statistical measure discussed in Section 4 suggests the generative model in the synthesizer is not learning the accuracies of the heuristics properly, or (2) there are no low confidence datapoints, as defined by ν, in the small, labeled dataset. Empirically, we find that (1) is a more popular termination condition than (2). In both cases, it is likely for some datapoints in the large, unlabeled set to not receive a label from any heuristic in the committed set; however, since Snuba generates training labels, the downstream end model can generalize to assign labels to these datapoints.

3.4 Discussion

We discuss the extension of the Snuba architecture to the multi-class setting, intuition behind the greedy approach, alternative heuristic models, and limitations of the system.

Multi-Class Setting. While we focus on the binary setting, Snuba can be extended to the multi-class setting without additional changes. We include an example of a three-class classification task in the Appendix. Statistics like F1 and Jaccard score in the synthesizer and pruner are calculated using only overall accuracy and coverage, which apply to the multi-class setting. The label aggregator in the verifier can operate over multi-class labels [40, 39] and pass feedback using the probabilistic label of the most likely class.

Greedy Approach. Our intuition behind generating heuristics greedily was to mimic the the user process of manually developing heuristics. The iterative approach tries to ensure each heuristic labels a subset of the data that does not have labels from existing heuristics and ensure a large portion of the datapoints receive high confidence labels. We use a statistical method to determine the optimal stopping condition for the iterative approach (Section 4, Figure 5).

Alternative Heuristic Models. While we only discuss three possible heuristic models in this paper, Snuba can handle any heuristic model that follows the input-output schema described in Section 3.1. The user can therefore design different heuristic models that are specialized for their classification task. For example, the user can use a regex heuristic model that can perform more complex operations over bag-of-words primitives than a decision tree.

Limitations. First, the performance of the Snuba heuristics is bounded by the quality of the input primitives. For example, if the primitives for the tumor classification task only contained age, which was a poor signal of tumor malignancy, then the heuristics Snuba generated would not assign high quality training labels. Second, Snuba heuristics can only rely on the input primitives and no external knowledge about the task, such as knowledge bases, which is a limitation compared to user-defined heuristics (more details in Section 5.2.3). Finally, Snuba is likely to overfit and not perform well on the unlabeled dataset if the small, labeled dataset is not representative of the unlabeled dataset. For the tumor classification task, the images in the small, labeled set could be taken from one perspective while the ones in the larger, unlabeled dataset are from a different perspective. This can lead the distribution of the primitives to be significantly different across the two datasets and prevent Snuba from generating high quality heuristics.

4. SNUBA SYSTEM GUARANTEES

We provide an overview of *generative models* [40, 52, 4, 54] that serve as the label aggregator for Snuba. As discussed in Section 2, these models can learn the accuracies of the noisy heuristics without using any ground truth data and can assign probabilistic labels to the unlabeled data accordingly. However, these generative models are designed to model the noise in *user-defined* heuristics, which are much more accurate than automatically generated heuristics. Specifically, the generative model assumes that heuristics always have accuracies better than 50%; however, Snuba-generated heuristics can easily violate this assumption as described in Section 4.2. Therefore, a key challenge in Snuba is recognizing whether the committed set includes heuristics that are worse than random for the unlabeled dataset without access to ground truth labels. We introduce a statistical measure in Section 4.3 that relies on the accuracies the generative model learns and the small labeled dataset. In Section 4.4, we formally define this statistical measure and provide a theoretical guarantee that it will recognize when the generative model is not learning heuristic accuracies successfully.

4.1 Generative Model

Generative models are a popular approach to learn and model the accuracies of different labeling sources like user-defined heuristics and knowledge bases when data is labeled by a variety of sources [11, 40]. In Snuba, we could also rely on the accuracies of the heuristics on the small, labeled dataset, $\hat{\alpha}$; however, this could degrade end model performance by up to 8.43 F1 points (Section 5). Formally, the goal of the generative model is to estimate the true accuracies of the heuristics, $\alpha^* \in \mathbb{R}^M$, using the labels the heuristics assign to the unlabeled data, $\hat{Y} \in \{-1, 0, 1\}^{M \times N_U}$. It models the true class label $Y^* \in \{-1, 1\}^{N_U}$ for a datapoint as a latent variable in a probabilistic model and in the simplest case, assumes that each labeling source is independent. The generative model is expressed as a factor graph:

$$\pi_\phi(\hat{Y}, Y^*) = \frac{1}{Z_\phi} \exp\left(\phi^T \hat{Y} Y^*\right) \qquad (1)$$

where Z_ϕ is a partition function to ensure π is a normalized distribution. The parameter $\phi \in \mathbb{R}^M$ is used to calculate the learned accuracies $\tilde{\alpha} = \frac{\exp(\phi)}{1+\exp(\phi)} \in \mathbb{R}^M$ (defined pointwise). It is estimated by maximizing the marginal likelihood of the observed heuristics $\hat{Y}$, using a method similar to contrastive divergence [19], alternating between using stochastic gradient descent and Gibbs sampling [40, 4]. The generative model assigns *probabilistic training labels* by computing $\tilde{Y} = \pi_\phi(Y^* \mid \hat{Y})$ for each datapoint. These probabilistic training labels can be used to train any end model with *noise-aware* loss [39, 40]

$$\min \sum_{i=1}^{N_U} \mathbb{E}_{y \sim \tilde{Y}} \left[l(h_\theta(o_i), y) \right]$$

where $o_i \in O_U$ is an object in the unlabeled dataset and $\tilde{Y}$ are the probabilistic training labels. In our experiments, we adjust the loss functions of several popular machine learning models to the use the noise-aware variant.

4.2 Assumption Violation

Since the generative model requires no ground truth labels to learn heuristic accuracies $\tilde{\alpha}$, it has to solve an underdetermined problem where the heuristics could have accuracies $\tilde{\alpha}$ or $1 - \tilde{\alpha}$. The generative model assumes that the labeling sources always perform better than random ($\alpha^* > 0.5$), which is a reasonable assumption for user-defined heuristics [40, 4, 52]. Since Snuba generates these heuristics automatically, it is possible for the heuristics to be accurate for the labeled set but violate the generative model's assumption that $\alpha^* > 0.5$. An example of such a situation is shown in Figure 5(a),(c) for two real datasets. The 8th and 12th heuristics, respectively, have an accuracy worse than 50% on the unlabeled dataset. However, since the generative model does not know that this assumption has been violated, it *learns* an accuracy much greater than 50% in both cases. If these heuristics are included in the generative model, the generated probabilistic training labels degrade end model performance by 5.15 F1 and 4.05 F1 points, respectively.

4.3 Statistical Measure

Snuba can take advantage of the *small, labeled dataset* to *indirectly* determine whether the generated heuristics are worse than random for the unlabeled dataset. We define the empirical accuracies of the heuristics as

$$\hat{\alpha}_i = \frac{1}{N_i} \sum_{j=1}^{N_L} \mathbb{1}(\hat{Y}_{ij} = Y_j^*),$$

for $i = 1 \ldots M$. $\hat{Y}_{ij} \in \{-1, 0, 1\}$ is the label heuristic i assigned to the j-th datapoint in the labeled set O_L, and N_i is the number of datapoints where $\hat{Y}_i \in \{1, -1\}$. Our goal is to use the empirical accuracies, $\hat{\alpha}$ to estimate whether the learned accuracies, $\tilde{\alpha}$ are close to the true accuracies, α^*, defined as $||\alpha^* - \tilde{\alpha}||_\infty < \gamma$, the maximum absolute difference between the learned and true accuracies being less than γ, a positive constant to be set. Toward this end, we define the measured error between the learned and empirical accuracies as $||\hat{\alpha} - \tilde{\alpha}||_\infty$. To guarantee with high probability that the generative model learns accuracies within γ, we want to find ϵ, the largest allowed error between the learned and empirical accuracies, $||\hat{\alpha} - \tilde{\alpha}||_\infty \leq \epsilon$ at each iteration. We discuss the exact form of ϵ in Section 4.4.

We compare the measured error $||\hat{\alpha} - \tilde{\alpha}||_\infty$ to the maximum allowable value of ϵ at *each iteration*, as shown in Figure 5(b),(d). If the measured error is greater than ϵ, then we stop the iterative process of generating heuristics and use the probabilistic training labels generated at the previous iteration (since the heuristic generated at the current iteration led to measured error being greater than ϵ). As shown in Figure 5, this stopping point maps to the iteration at which the new heuristic generated has a true accuracy α^* worse than 50% for the unlabeled dataset (we only calculate α^* for demonstration since we would not have access to ground truth labels for real-world tasks). Intuitively, we expect that once the synthesizer generates a heuristic that is worse than random for the unlabeled dataset, it will never generate heuristics that will be helpful in labeling the data

Table 1: Dataset Statistics and Descriptions. N_L, N_U are size of labeled and unlabeled datasets. $\frac{N_L}{N_U}$: ratio of unlabeled to labeled data, D: number of primitives. Label sources are previous sources of training labels (DT: decision tree, UDF: user-defined functions, DS: distant supervision with knowledge base.)

Application	**Domain**	N_L	N_U	$\frac{N_U}{N_L}$	D	**Label Source**	**Task**
Bone Tumor	Image	200	400	2.0	17	DT + User	Aggressive vs. Non-aggressive Tumor
Mammogram	Image	186	1488	8.0	10	UDF	Malignant vs. Benign Tumor
Visual Genome	Image	429	903	2.1	7	UDF	Identify 'person riding bike'
MS-COCO	Multi-Modal	6693	26772	4.0	105	UDF	Identify whether person in image
IMDb	Text	284	1136	4.0	322	UDF/Crowd	Action vs. Romance Plot Descriptions
Twitter	Text	123	14551	118.3	201	Crowd	Positive vs. Negative Tweets
CDR	Text	888	8268	9.3	298	UDF + DS	Text relation extraction
Hardware	Multi-Modal	100	100,000	1000	237	UDF	Richly formatted data relation extraction

anymore. Empirically, we observe that this is indeed the case as shown for two real tasks in Figure 5(a) and (c).

4.4 Theoretical Guarantees

Assuming that the objects in the labeled set O_L are independent and identically distributed, we provide the following guarantee on the probability of the generative model learning the accuracies successfully:

Proposition 1: *Suppose we have M heuristics with empirical accuracies $\hat{\alpha}$, accuracies learned by the generative model $\tilde{\alpha}$, and measured error $||\hat{\alpha} - \tilde{\alpha}||_\infty \leq \epsilon$ for all M iterations. Then, if each heuristic labels a minimum of*

$$N \geq \frac{1}{2(\gamma - \epsilon)^2} \log\left(\frac{2M^2}{\delta}\right)$$

datapoints at each iteration, the generative model will succeed in learning accuracies within $||\alpha^* - \tilde{\alpha}||_\infty < \gamma$ across all iterations with probability $1 - \delta$.

We provide a formal proof for this proposition in the Appendix. We require each heuristic to assign labels to at least N datapoints to guarantee that the generative model will learn accuracies within γ of the true accuracies, given the measured error is less than ϵ *for all iterations.* We solve for the maximum allowed error ϵ *at each iteration*:

$$\epsilon = \gamma - \sqrt{\frac{1}{2N} \log\left(\frac{2M}{\delta}\right)}.$$

This value is plotted against the value of the measured error $||\hat{\alpha} - \tilde{\alpha}||_\infty$ in Figure 5(b,d). Snuba stops generating new heuristics when the measured error surpasses the allowed error. The above proposition relies only on the measured error to guarantee whether the generative model is learning accuracies successfully.

5. EVALUATION

We compare the performance of end models trained on labels generated by Snuba and other baseline methods. We seek to experimentally validate the following claims:

- **Training labels from Snuba outperform labels from automated baseline methods** We compare Snuba to models that generate heuristics using only the labeled data, such as boosting and decision trees, and semi-supervised methods, which utilize both labeled and unlabeled datasets. Snuba outperforms these methods by up to 14.35 F1 points. We also compare to transfer learning using only the labeled dataset, which Snuba outperforms by up to 5.74 F1 points.
- **Training labels from Snuba outperform those from user-developed heuristics** We compare the performance of heuristics generated by Snuba to heuristics developed by users. Snuba can use the *same* amount of labeled data as users to generate heuristics and improve end model performance by up to 9.74 F1 points.
- **Each component of Snuba boosts overall system performance** We evaluate separate components of the Snuba system by changing how the β parameter is chosen in the synthesizer, how the pruner selects a heuristic to add to the committed set, and different label aggregation methods in the verifier. Compared to the complete Snuba system, we observe that performance can degrade by up to 20.69 F1 points by removing these components.

5.1 Experiment Setup

We describe the datasets, baseline methods, performance metrics, and implementation details for Snuba.

5.1.1 Datasets

We consider real-world applications and tasks over open source datasets for image, text, and multi-modal classification. For each of the tasks, previous techniques to assign training labels included using crowdsourcing, user-defined functions, and decision trees based on a small, labeled dataset. Summary statistics are provided in Table 1 and additional details are in the Appendix.

Image Classification. We focus on two real-world medical image classification tasks that we collaborated on with radiologists at Stanford Hospital and Clinics. The **Bone Tumor** and **Mammogram** tumor classification tasks demonstrate how Snuba-generated heuristics compare to those developed by domain experts. The first dataset uses domain-specific primitives while the second relies on simple geometric primitives. Working with graduate students in the Stanford Computer Vision lab, we identify images of "person riding bike". We use the **Visual Genome** database [26] with bounding box characteristics as primitives and study how Snuba performs with severe class imbalance.

Text and Multi-Modal Classification. We applied Snuba to text and multi-modal datasets to study how well Snuba operated in domains where humans could easily interpret and write rules over the raw data. We generate primitives by featurizing the text using a bag-of-words representation. The **MS-COCO** dataset [31] had heuristics generated over

Table 2: Improvement of end model (F1 Score) trained on labels from Snuba compared to labels from automated baselines and UDFs (+: Snuba better). *Hardware UDFs tuned on 47,413 labeled datapoints, other baselines on 100 labeled datapoints.

		Snuba Improvement Over				
Application	**Snuba F1 Score**	**Decision Tree**	**Boosting**	**Transfer Learning**	**Semi-Supervised**	**UDF**
Bone Tumor	71.55	+6.37	+8.65	-	+6.77	+9.13
Mammogram	74.54	+5.33	+5.02	+5.74	+3.26	+9.74
Visual Genome	56.83	+7.62	+6.20	+5.58	+5.94	+6.38
MS-COCO	69.52	+1.65	+2.70	+2.51	+1.84	+2.79
IMDb	62.47	+7.78	+12.12	+3.36	+14.35	+3.67
Twitter	78.84	+5.03	+4.43	-	+3.84	+13.8
CDR	41.56	+5.65	+11.22	-	+7.49	-12.24
Hardware	68.47	+5.20	+4.16	-	+2.71	-4.75*

captions and classification performed over associated images, and the **IMDb** plot summary classification [1] is purely text-based. The **Twitter** sentiment analysis dataset relied on crowdworkers for labels [32] while the chemical-disease relation extraction task (**CDR**) [58] relies on external sources of information like knowledge bases. The **Hardware** relation extraction task over richly formatted data classifies part numbers and electrical characteristics from specification datasheets[1] as valid or not. We use visual, tabular, and structural primitives extracted using Fonduer [59].

5.1.2 Baseline Methods

We compare to pruned **decision tree [43]** and **boosting [17]** (AdaBoost), which use the labeled dataset to generate one complex or multiple, simple decision trees, respectively. We compare to **semi-supervised learning [62]**, which uses *both* the labeled and unlabeled dataset to assign training labels and represents a single 'heuristic' in the form of a black-box model. For select tasks, we perform **transfer learning** using pre-trained models. We use GLoVE embeddings [37] for IMBd and Twitter only tune the last layer of a VGGNet [45] for MS-COCO, and tune the weights of a GoogLeNet [49] pre-trained on ImageNet [12] for Visual Genome and Mammogram (more details in the Appendix).

As shown in Table 1, training labels for all tasks were previously generated by some **user-driven labeling method**, such as user-defined heuristics, distant supervision, or crowdsourcing. These were developed by users, ranging from domain experts to machine learning practitioners and input to label aggregators we developed [52, 39, 53]. For tasks like CDR, Bone Tumor, and Mammogram that required specific domain knowledge, the time taken for bioinformatics experts and radiologists to manually develop heuristics ranged from a few days to a few weeks. For tasks that did not require domain expertise, such as IMDb and Visual Genome, graduate students wrote a small number of heuristics over a period of a few hours. In all cases, users encoded their domain knowledge in heuristics and evaluated their performance on a small, held-out labeled set in an iterative manner.

5.1.3 Implementation Details

Primitives for Snuba. Since text-based tasks used a bag-of-words representation, the primitives are sparse and number in the hundreds of thousands. We filter bag-of-words primitives by only considering primitives that are active for both the labeled and unlabeled dataset, and for at least 5% of the unlabeled dataset to ensure a minimim coverage for generated heuristics. The 5% threshold had the best performance for our text datasets but this threshold can be user-defined in practice.

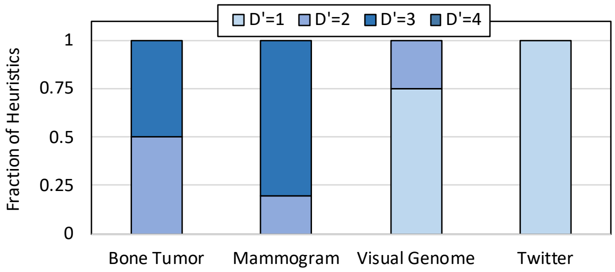


Figure 6: We observe a maximum of $D' = 1$ for our text and $D' < 4$ for our image and multi-modal tasks.

For our image-based tasks, we found that Snuba never generated heuristics that relied on more than 4 primitives as input, while for text-based tasks, it only generated heuristics that relied on a single primitive (Figure 6). Heuristics rely on a small number of primitives since this limits their complexity and prevents them from overfitting to the small, labeled dataset. Moreover, relying on multiple primitives can also lower the coverage of the heuristics, and a fairly accurate heuristic that relies on several primitives being present is filtered by the pruner, which relies on both coverage and performance. The relatively small number of primitives heuristics used as input leads to a maximum single threaded runtime of 14.45 mins for the Hardware task on a Xeon E7-4850 v3 CPU.

Performance Metrics. To measure performance, we report the F1 score of an end model trained on labels from Snuba and the baseline methods on a test set. We report F1 score instead of accuracy since some datasets have class imbalance that can lead to high accuracy by naively predicting the majority class for all datapoints. The F1 scores for the end model are defined in terms of true positives instead of correctly classified datapoints (this is different Section 3.1.2, since the end models never abstain).

End Models. While Snuba can generate training labels efficiently, they rely only on the user-defined primitives. The end model trained on these labels can use the raw data or representations of the data based on pre-trained models. For example, the end model can operate over the entire raw image, sentence or representation from a pre-trained model as

[1] https://www.digikey.com

Table 3: Precision (P), Recall (R) and F1 scores for user-defined heuristics, Snuba-generated heuristics, and end model trained on labels from Snuba-generated heuristics. Lift reported is from user to Snuba heuristics, then Snuba heuristics to end model. Snuba heuristics have lower precision than users' and end model improves recall.

Application	User Heuristics			Snuba Heuristics				Snuba + End Model			
	F1	P	R	F1	P	R	Lift(F1)	F1	P	R	Lift(F1)
Bone Tumor	30.91	89.47	18.68	31.58	33.75	29.67	+0.67	71.55	58.86	91.21	+39.97
Visual Genome	34.76	98.28	21.11	46.06	48.10	44.19	+11.30	56.83	41.34	90.91	+10.77
MS-COCO	21.43	63.66	12.88	24.41	29.40	41.49	+12.98	69.52	55.80	92.16	+35.11
IMDb	20.65	76.19	11.94	46.47	48.03	45.52	+25.82	62.47	45.42	100.	+16.00

opposed to measurements of the tumor, bag-of-words representation, or bounding box coordinates. For image classification tasks, we use popular deep learning models like GoogLeNet and VGGNet that take the raw image as input, while for text tasks we use a model composed of a single embedding and a single LSTM layer that take the raw text sentence(s) as input. These models take as input the probabilistic or binary training labels from Snuba or the baseline methods and minimize the noise-aware loss, as defined in Section 4. While the tasks explored in this section are all binary classification, the system can be easily generalized to the multi-class case (Section 3.4).

5.2 End to End System Comparison

We demonstrate that a downstream model trained on the labels from Snuba generalizes beyond the Snuba heuristics, improving recall by up to 61.54 points (Section 5.2.1), outperforms automated baseline methods by up to 12.12 F1 points (Section 5.2.2) and user-driven labeling by up to 9.74 F1 points (Section 5.2.3).

5.2.1 *Generalization beyond Heuristics*

One of the motivations for designing Snuba is to efficiently label enough training data for training powerful, downstream machine learning models like neural networks. Heuristics from Snuba are not used directly for the classification task at hand because (1) they may not label the entire dataset due to abstentions, and (2) they are based only on the user-defined primitives and fail to take advantage of the raw data representation. For datasets like MS-COCO, the end model also operates over a different modality than the heuristics. To demonstrate the advantage of training an end model, we compare the performance of Snuba heuristics to standard end models trained on labels from Snuba on a test set in Table 3. The end model improves over the heuristics' performance by up to 39.97 F1 points. The end model helps *generalize* beyond the heuristics, as a result of more powerful underlying models and access to raw data, and improves recall by up to 61.54 points.

5.2.2 *Automated Methods*

Table 2 shows that Snuba can outperform automated baseline methods by up to 14.35 F1 points. Snuba outperforms decision trees, which fit a single model to the labeled dataset, by 7.38 F1 points on average, the largest improvement compared to other baselines. The method that performs the closest to Snuba for most tasks is semi-supervised learning, which takes advantage of both the unlabeled and unlabeled dataset, but fails to account for diversity, performing worse than Snuba by 6.21 F1 points on average. Finally, compared to transfer learning which does not have to learn a representation of the data from scratch, Snuba performs up to 5.74 F1 points better using the same amount of labeled data. This demonstrates how for many tasks, using a larger training set with noisy labels is able to train a better end model from scratch than fine tuning a pre-trained model with a small labeled dataset.

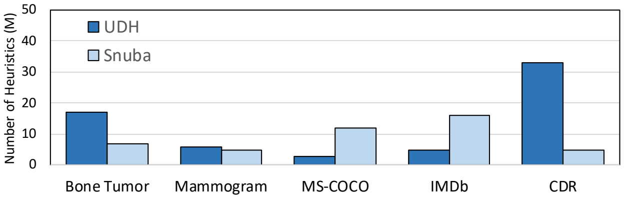


Figure 7: Snuba generates fewer heuristics than users for our image tasks and usually more for text tasks.

5.2.3 *User-Driven Labeling Methods*

We compare end model performance trained on labels Snuba generates to labels from manually generated labeling sources in Table 2 and report the precision, recall, and F1 score of Snuba-generated and user-defined heuristics in Table 3. The labels from the heuristics are combined using the Snuba label aggregator, the generative model in Section 4. Overall, Snuba generates heuristics that perform up to 25.82 F1 points better than user-defined heuristics. Note that users develop heuristics that are very high precision, up to 98.28 points. Snuba-generated heuristics, on the other hand, balance both precision and recall. This supports the design of the system since the synthesizer optimizes for F1 score, which relies on *both* precision and recall, and the pruner optimizes for both accuracy and coverage, which are related to both precision and recall.

For image domains, Snuba generates fewer heuristics (Figure 7) that depend on more primitives than user-defined heuristics. Primitives for image domains are numerical and require guessing the correct threshold for heuristics, a process Snuba automates while users guess manually. For the Bone Tumor classification task, the user-defined heuristics were manually tuned versions of decision trees fit to the labeled set. Therefore, Snuba only improves 0.67 F1 points over this partially automated approach. For text datasets (MS-COCO and IMDb), Snuba generates almost 5× as many heuristics as users since each heuristic relies only on a single primitive and improves F1 score by up to 25.82 points (Table 3). For CDR, users relied on distant supervision through the Comparative Toxicogenomics Database [10]. Snuba only relies on the primitives it has access to and cannot incorporate any external information, leading to 7.71 F1 points lower performance than user-defined heuristics using distant supervision. Finally, for Hardware, Snuba uses only 100 labeled datapoints to generate heuristics while users had ac-

Table 4: Improvement of best heuristic type over others and Snuba choosing β over never abstaining ($\beta = 0$) and midpoint value ($\beta = 0.25$). 0.00 is best heuristic type that was best for each task. DT: decision tree, LR: logistic regressor; NN: nearest neighbor.

	F1 Improvement Over				
Dataset	**DT**	**LR**	**NN**	$\beta = 0$	$\beta = 0.25$
Bone Tumor	+2.73	0.00	+4.62	+2.35	+3.77
Visual Genome	+3.22	+3.38	0.00	+7.99	+5.30
MS-COCO	0.00	0.00	+0.24	+2.51	+2.51
IMDb	0.00	0.00	+14.32	+20.69	+2.13

cess to $47,413$, which leads to Snuba performing 4.75 F1 points worse in terms of end model performance.

5.3 Micro-Benchmarking Results

We evaluate the individual components of the Snuba system and show how adjustments to each component can affect end model performance by up to 20.69 F1 points.

5.3.1 Synthesizer

First, we compare how different heuristic models perform for select tasks in Table 4 and show how much better the best heuristic type (marked as 0) performs compares to alternate heuristic types. For text-based tasks, decision tree and logistic regressor based heuristics perform the same since they both rely on a single primitive and learn the same threshold to make a binary decision. These heuristic models essentially check whether a word exists in a sentence.

Next, we set $\beta = 0$ to prevent heuristics from abstaining and set it to a constant $\beta = 0.25$, the midpoint of possible values $\beta \in (0, 0.5)$ (Table 4). Allowing heuristics to abstain can improve end model performance by up to 20.69 F1 points and choosing the correct β value can improve end model performance by up to 5.30 F1 points.

5.3.2 Pruner

We show the performance of the pruner compared to only optimizing for either performance (with F1 score) or diversity (with Jaccard distance) in Table 5. For text tasks, only optimizing for performance comes within 2.15 F1 points of the Snuba pruner since each heuristic selecting a different word automatically accounts for diversity. On the other hand, only optimizing for diversity in text domains can affect performance by up to 18.20 F1 points since it could result in a large portion of the unlabeled dataset receiving low-quality labels. We also compare to weighting the F1 score by how much of the unlabeled dataset is covered, which performs closest to the simple average case for text-based tasks. This suggests that other domain-specific weighting schemes, like weighting coverage more than accuracy given sparse primitives can further improve performance.

5.3.3 Verifier

Finally, we look at how learning heuristic accuracies for label aggregation compares to majority vote in Table 6. Text domains in which the number of heuristics generated is more than 15, the majority vote score comes within 1.80 F1 points of the Snuba verifier. With a large number of heuristics, each datapoint receives enough labels that learning accuracies has little effect on the assigned labels [29].

Table 5: Snuba pruner optimizing for only performance (F1) and diversity (Jaccard) compared to performance and diversity individually and a weighted average.

	F1 Improvement Over		
Dataset	**F1 Only**	**Jaccard Only**	**Weighted**
Bone Tumor	+3.86	+8.84	+2.74
Visual Genome	+6.57	+7.33	+3.74
MS-COCO	+2.51	+18.2	+0.80
IMDb	+2.15	+14.23	+0.37

Table 6: Snuba verifier aggregation compared to using $\hat{\alpha}$ instead of $\tilde{\alpha}$, no termination condition, majority vote (MV) across labels, feedback with weighted samples.

	F1 Improvement Over			
Dataset	$\hat{\alpha}$	**No Term.**	**MV**	**Weighted**
Bone Tumor	+5.42	+5.15	+3.78	+1.76
Visual Genome	+8.43	+7.09	+6.59	+4.42
MS-COCO	+7.98	+3.70	+3.00	+2.22
IMDb	+5.67	+4.05	+1.80	+1.63

We compare to using the empirical accuracies of the heuristics $\hat{\alpha}$ rather than learning accuracies based on labels assigned to the unlabeled data. This method performs worse than the Snuba verifier by up to 8.43 F1 points. We also generate heuristics till there are no more datapoints in the small, labeled dataset with low confidence labels and find that this can degrade end model performance by up to 7.09 F1 points as shown in Table 6.

We compare to passing a *weighted* version of the small, labeled dataset as feedback to the synthesizer instead of a subset and find it performs up to 4.42 F1 points worse than passing a subset. We posit that heuristics fit to a weighted set can lead to more low confidence labels and eventually a higher rate of abstentions for the unlabeled dataset.

6. RELATED WORK

We provide an overview of methods that label data automatically based on heuristics, use both labeled an unlabeled data, and aggregate noisy sources of labels.

Rule Learning. The inspiration for Snuba comes from program synthesis, where programs are generated given access to a set of input-output pairs [16, 46], reference implementations [3], or demonstrations [22]. The design is based loosely on counter-example guided inductive synthesis (CEGIS) in which a synthesizer generates programs, passes it to the verifier that decides whether the candidate program satisfies the given specifications, and passes relevant feedback to the synthesizer [46, 16, 22, 48]. However, unlike Snuba, such models only synthesize programs that match *all* the specified input-output pairs. Other works also generate heuristics to help interpret the underlying data labels [56, 55], but neither methods use *unlabeled data* since the programs generated either mimic the desired program perfectly or provide interpretations for existing labels. While Snuba focuses on generating training labels for various domains, rule learning has been widely studied in the context of information extraction [47, 34]. Recent works can learn logical rules for knowledge base reasoning [60], interleave beam search with parameter learning [25], select rules from a restricted set using lasso regression [28], and use alternate gradient-based search to find parameters for probailistic logic [57]. While these methods are more sophisticated than Snuba, they use a large amount of training data and rely directly on the generated rules for prediction. Incorporating these methods into the Snuba synthesizer could be interesting for future work, especially for text-based tasks.

Training Label Generation. Focusing on the problem of generating training data, Snorkel [39] is a system that relies on domain experts manually developing heuristics, patterns, or distant supervision rules to label data noisily. While users in Snorkel rely on a small, labeled dataset to evaluate and refine their heuristics, Snuba automatically generates heuristics using the labeled and unlabeled data it has access to. Snorkel and Snuba both use the generative model to aggregate heuristic labels, but Snuba can generate heuristics that are noisier than the generative model can account for. Therefore, it uses a statistical measure to determine when the generative model can be used (Section 4). Other methods that rely on imperfect sources of labels that are partially user-defined include heuristic patterns [18, 6] and distant supervision [8, 33], which relies on information present in knowledge bases.

Utilizing Labeled and Unlabeled Data. To train a deep learning model with a small, labeled dataset, a common approach is using transfer learning, or retraining models that have been trained for different tasks that have abundant training data in the same domain [35]. However, this approach does not take advantage of any unlabeled data available. Semi-supervised learning leverages both labeled and unlabeled data, along with assumptions about low-dimensional structure and smoothness of the data to automatically assign labels to the unlabeled data [7, 62]. Unlike semi-supervised learning, which generates a single black-box model, Snuba generates multiple, diverse heuristics to label the unlabeled data. Moreover, as demonstrated in Section 5, Snuba performs better than a specific semi-supervised model, label spreading [62], when the amount of unlabeled data is larger than than the amount of labeled data. Co-training [5] also takes advantage of both labeled and unlabeled data and trains two *independent* models on two separate views of the data. Snuba does not require access to separate feature sets as views and can generate more than two heuristics (classifiers) that can be correlated with each other (Section 4).

Combining Noisy Labels. Combining labels from multiple sources like heuristics is well-studied problem [11], especially in the context of crowdsourcing [9, 23, 61]. However, these methods assume the labeling sources are not generated automatically and requires a labeled dataset to *learn* the accuracies of the different sources. Other methods, including our previous work [52, 40, 54], rely on generative models to learn accuracies and dependencies among labeling sources [2, 42, 50]. Areas like data fusion [13, 41, 38] and truth discovery [30] also look at the problem of estimating how reliable different data sources are while utilizing probabilistic graphical models like Snuba.

7. CONCLUSION

Snuba is a system to automatically generate heuristics using a small labeled dataset to assign training labels to a large, unlabeled dataset, which can be used to train a downstream model of choice. It iteratively generates heuristics that are accurate and diverse for the unlabeled dataset using the small, labeled dataset. Snuba relies on a statistical measure to determine when generated heuristics are too noisy and therefore when to terminate the iterative process. We demonstrate how training labels from Snuba outperform labels from semi-supervised learning by up to 14.35 F1 points and from user-defined heuristics by up to 9.74 F1 points in terms of end model performance for tasks across various domains. Our work suggests that there is potential to use a small amount of labeled data to make the process of generating training labels much more efficient.

Acknowledgments

We thank Christopher Aberger, Jared Dunnmon, Avner May, Shoumik Palkar, Theodoros Rekatsinas, Sahaana Suri, and Sandeep Tata for their valuable feedback, and Sen Wu for help with the Hardware dataset. We gratefully acknowledge the support of DARPA under Nos. FA87501720095 (D3M) and FA86501827865 (SDH), NIH under No. N000141712266 (Mobilize), NSF under Nos. CCF1763315 (Beyond Sparsity) and CCF1563078 (Volume to Velocity), ONR under No. N000141712266 (Unifying Weak Supervision), the Moore Foundation, NXP, Xilinx, LETI-CEA, Intel, Google, NEC, Toshiba, TSMC, ARM, Hitachi, BASF, Accenture, Ericsson, Qualcomm, Analog Devices, the Okawa Foundation, and American Family Insurance, the National Science Foundation Graduate Research Fellowship under Grant No. DGE-114747, the Joseph W. and Hon Mai Goodman Stanford Graduate Fellowship,and members of the Stanford DAWN project: Intel, Microsoft, Teradata, Facebook, Google, Ant Financial, NEC, SAP, and VMWare. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of DARPA, NIH, ONR, or the U.S. Government.

8. REFERENCES

[1] IMDb Dataset. `https://www.imdb.com/interfaces/`.

[2] E. Alfonseca, K. Filippova, J.-Y. Delort, and G. Garrido. Pattern learning for relation extraction with a hierarchical topic model. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 54–59. Association for Computational Linguistics, 2012.

[3] R. Alur, R. Bodik, G. Juniwal, M. M. Martin, M. Raghothaman, S. A. Seshia, R. Singh, A. Solar-Lezama, E. Torlak, and A. Udupa. Syntax-guided synthesis. In *Formal Methods in Computer-Aided Design (FMCAD), 2013*, pages 1–8. IEEE, 2013.

[4] S. H. Bach, B. He, A. Ratner, and C. Ré. Learning the structure of generative models without labeled data. In *International Conference on Machine Learning*, pages 273–282, 2017.

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

[6] R. Bunescu and R. Mooney. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 576–583, 2007.

[7] O. Chapelle, B. Scholkopf, and A. Zien. Semi-supervised learning (Chapelle, O. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[8] M. Craven, J. Kumlien, et al. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*, volume 1999, pages 77–86, 1999.

[9] N. Dalvi, A. Dasgupta, R. Kumar, and V. Rastogi. Aggregating crowdsourced binary ratings. In *Proceedings of the 22nd international conference on World Wide Web*, pages 285–294. ACM, 2013.

[10] A. P. Davis, C. J. Grondin, R. J. Johnson, D. Sciaky, B. L. King, R. McMorran, J. Wiegers, T. C. Wiegers, and C. J. Mattingly. The comparative toxicogenomics database: update 2017. *Nucleic acids research*, 45(D1):D972–D978, 2016.

[11] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied statistics*, pages 20–28, 1979.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[13] X. L. Dong and D. Srivastava. Big data integration. *Synthesis Lectures on Data Management*, 7(1):1–198, 2015.

[14] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[15] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[16] S. Gulwani. Synthesis from examples: Interaction models and algorithms. In *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2012 14th International Symposium on*, pages 8–14. IEEE, 2012.

[17] T. Hastie, S. Rosset, J. Zhu, and H. Zou. Multi-class AdaBoost. *Statistics and its Interface*, 2(3):349–360, 2009.

[18] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics, 1992.

[19] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[20] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[21] P. Jaccard. Lois de distribution florale dans la zone alpine. *Bull Soc Vaudoise Sci Nat*, 38:69–130, 1902.

[22] S. Jha, S. Gulwani, S. A. Seshia, and A. Tiwari. Oracle-guided component-based program synthesis. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, pages 215–224. ACM, 2010.

[23] M. Joglekar, H. Garcia-Molina, and A. Parameswaran. Comprehensive and reliable crowd assessment algorithms. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pages 195–206. IEEE, 2015.

[24] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 1988–1997. IEEE, 2017.

[25] S. Kok and P. Domingos. Statistical predicate invention. In *Proceedings of the 24th international conference on Machine learning*, pages 433–440. ACM, 2007.

[26] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, et al. Visual Genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73, 2017.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[28] N. Lao and W. W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67, 2010.

[29] H. Li, B. Yu, and D. Zhou. Error rate analysis of labeling by crowdsourcing. In *ICML Workshop: Machine Learning Meets Crowdsourcing. Atalanta, Georgia, USA*. Citeseer, 2013.

[30] Y. Li, J. Gao, C. Meng, Q. Li, L. Su, B. Zhao, W. Fan, and J. Han. A survey on truth discovery. *ACM Sigkdd Explorations Newsletter*, 17(2):1–16, 2016.

[31] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[32] C. Metz. Crowdflower dataset: Airline Twitter sentiment, 2015. `https://www.crowdflower.com/data/airline-twitter-sentiment/`.

[33] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics, 2009.

[34] R. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, volume 334, 1999.

[35] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[37] J. Pennington, R. Socher, and C. Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[38] R. Pochampally, A. Das Sarma, X. L. Dong, A. Meliou, and D. Srivastava. Fusing data with correlations. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 433–444. ACM, 2014.

[39] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré. Snorkel: Rapid training data creation with weak supervision. *PVLDB*, 11(3):269–282, 2017.

[40] A. J. Ratner, C. M. De Sa, S. Wu, D. Selsam, and C. Ré. Data programming: Creating large training sets, quickly. In *Advances in Neural Information Processing Systems*, pages 3567–3575, 2016.

[41] T. Rekatsinas, M. Joglekar, H. Garcia-Molina, A. Parameswaran, and C. Ré. SLiMFast: Guaranteed results for data fusion and source reliability. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1399–1414. ACM, 2017.

[42] B. Roth and D. Klakow. Combining generative and discriminative model scores for distant supervision. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 24–29, 2013.

[43] S. R. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[44] R. Sawyer-Lee, F. Gimenez, A. Hoogi, and D. Rubin. Curated breast imaging subset of DDSM, 2016.

[45] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[46] R. Singh and S. Gulwani. Synthesizing number transformations from input-output examples. In *International Conference on Computer Aided Verification*, pages 634–651. Springer, 2012.

[47] S. Soderland. Learning information extraction rules for

semi-structured and free text. *Machine learning*, 34(1-3):233–272, 1999.

[48] A. Solar-Lezama, L. Tancau, R. Bodik, S. Seshia, and V. Saraswat. Combinatorial sketching for finite programs. *ACM Sigplan Notices*, 41(11):404–415, 2006.

[49] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015.

[50] S. Takamatsu, I. Sato, and H. Nakagawa. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 721–729. Association for Computational Linguistics, 2012.

[51] S. Van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu. scikit-image: image processing in Python. *PeerJ*, 2:e453, 2014.

[52] P. Varma, B. D. He, P. Bajaj, N. Khandwala, I. Banerjee, D. Rubin, and C. Ré. Inferring generative model structure with static analysis. In *Advances in Neural Information Processing Systems*, pages 239–249, 2017.

[53] P. Varma, D. Iter, C. De Sa, and C. Ré. Flipper: A systematic approach to debugging training sets. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics*, page 5. ACM, 2017.

[54] P. Varma, R. Yu, D. Iter, C. De Sa, and C. Ré. Socratic learning: Correcting misspecified generative models using discriminative models. *arXiv preprint arXiv:1610.08123*, 2017.

[55] F. Wang and C. Rudin. Falling rule lists. In *Artificial Intelligence and Statistics*, pages 1013–1022, 2015.

[56] T. Wang, C. Rudin, F. Doshi-Velez, Y. Liu, E. Klampfl, and P. MacNeille. Or's of and's for interpretable classification, with application to context-aware recommender systems. *arXiv preprint arXiv:1504.07614*, 2015.

[57] W. Y. Wang, K. Mazaitis, and W. W. Cohen. Structure learning via parameter learning. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1199–1208. ACM, 2014.

[58] C.-H. Wei, Y. Peng, R. Leaman, A. P. Davis, C. J. Mattingly, J. Li, T. C. Wiegers, and Z. Lu. Overview of the biocreative v chemical disease relation (CDR) task. In *Proceedings of the fifth BioCreative challenge evaluation workshop*, pages 154–166. Sevilla Spain, 2015.

[59] S. Wu, L. Hsiao, X. Cheng, B. Hancock, T. Rekatsinas, P. Levis, and C. Ré. Fonduer: Knowledge base construction from richly formatted data. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1301–1316. ACM, 2018.

[60] F. Yang, Z. Yang, and W. W. Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems*, pages 2319–2328, 2017.

[61] Y. Zhang, X. Chen, D. Zhou, and M. I. Jordan. Spectral methods meet EM: A provably optimal algorithm for crowdsourcing. In *Advances in neural information processing systems*, pages 1260–1268, 2014.

[62] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in neural information processing systems*, pages 321–328, 2004.

APPENDIX

A. EVALUATION DETAILS

We describe the tasks and baselines from Section 5 in more detail, along with an expanded version of Table 3 .

A.1 Datasets

Bone Tumor The first dataset in the medical domain consists of 800 X-rays of bones from various parts of the body, and we give Snuba access to 200 hand-labeled examples. Each X-ray also contains an associated binary mask which denotes where the tumor occurs in the image. Radiologists extract 400 domain-specific primitives based on the shape, texture, and intensity of the tumor, out of which we use 17 as primitives for Snuba and the rest as features for a logistic regression model. *We use this dataset to show how Snuba can efficiently assign labels to previously unlabeled images and improve over the oracle score.*

Mammogram The second medical dataset is based on the publicly available DDSM-CBIS [44] dataset, in which each CT scan has an associated segmentation binary mask that outlines the location of the tumor in the mammogram and a label for whether the tumor is malignant or benign. Given this segmentation, we extract simple features such as area and perimeter, which require no domain expertise, unlike the Bone Tumor dataset. *We use this dataset to demonstrate how Snuba performs well even when the quality of the primitives is mediocre and not domain-specific.*

Visual Genome [26] We define a image query of whether an image contains a person riding a bike or not by defining primitives over bounding box characteristics of the objects in the image. This query results in significant class imabalance in the labeled and unlabeled dataset, with the labeled dataset containing only 18% positive examples. *We use this dataset to measure how Snuba performs with a small number of easily interpretable primitives and with a skewed labeled dataset to learn heuristics based on.*

MS-COCO [31] In this multi-modal dataset, each image has five text caption associated with it and the goal was to build a classifier to determine whether there was a person in the image. Snuba only has access to the bag-of-words representations of these captions as primitives and the deep learning end model has access only to the images associated with the captions. *We use this dataset to demonstrate how Snuba can operate over the text domain and how the domain that heuristics are generated for can be different from the domain the final classifier is running on.*

IMDb The task was to classify plot summaries of movies as describing action or romance movies. Users had previously developed a collection of 6 heuristics to label this data; however, we found that Snuba was able to outperform these hand-written heuristics for a task that did not require any domain expertise. *We use this dataset to demonstrate how Snuba can outperform user-based heuristics for text-based tasks that do not require any domain expertise.*

Twitter [32] The originally crowdsourced task is to determine whether a specific tweet has a positive or negative connotation. Since less than 1% of the data was labeled by crowdworkers, Snuba used that subset as the labeled set. With such a significant difference between the labeled and unlabeled dataset, we wanted to see how well Snuba can generalize. Moreover, results of this task has interesting follow on work for how to make crowdsourcing more efficient. *We use this dataset to demonstrate how Snuba can work even when the labeled dataset is signficantly smaller than the unlabeled set and how this can help make crowdsourcing more efficient.*

Chemical-Disease Relation Extraction (CDR) [58] The task is to detect relations among chemicals and diseases mentioned in PubMed abstracts [58]. Previously, a combination of distant supervision from the Comparative Toxicogenomics Database [10] and user-defined heuristics were used to build a training set for this task. Unlike previous datasets, the heuristics contain additional, structured information from the knowledge base, which puts Snuba at a disadvantage. *We use this dataset to demonstrate how Snuba generated heuristics compare to these complex rules and* external sources of information.

A.2 Baselines

Decision Tree [43]: We pass in all the primitives associated with the labeled datapoints and prune the tree structure by limiting the number of datapoints at the leaves and the number of datapoints required to split a node. This baseline does not use the unlabeled dataset and represents generating a single heuristic using only the labeled dataset.

Boosting [17]: We compare to AdaBoost, which generates a collection of weak classifiers, which are imperfect decision trees in our evaluation. This method also takes as input all the primitives associated with the labeled datapoints for generating the imperfect decision trees. This baseline does not use the unlabeled dataset and represents generating a multiple, noisy heuristics using only the labeled dataset.

Semi-Supervised Learning [62]: We compare to label spreading [62], a semi-supervised method that uses *both* the labeled and unlabeled dataset to assign training labels. This baseline represents generating a single 'heuristic' in the form of a black-box model using both the labeled and unlabeled datasets.

Transfer Learning: For select tasks, there exist pre-trained models that performed well on a similar task for the same data modality. In these cases, we also compared against transfer learning, or fine-tuning the weights from a pre-trained model using only the labeled datapoints. For IMDb and Twitter, we use GLoVE embeddings [37] and only learn weights for a single LSTM layer on top of the embeddings, only tune the last layer of a VGGNet [45] for MS-COCO, and tune the weights of a GoogLeNet [49] pre-trained on ImageNet [12] over a few iterations for the Visual Genome and Mammogram datasets. This baseline represents a popular method that users rely on when they have a small amount of labeled data.

A.3 Extended Generalization Comparison

We extend Table 3 in Table 7 and discuss two special cases, Twitter and CDR. For Twitter, the "user-heuristics" are different crowdworkers. Unlike other tasks for which precision is much higher than recall, the precision and recall are balanced in this case. Snuba heuristics improve over the precision slightly and boost recall to 100. In terms of the end model, it lowers recall but improves precision significantly over the heuristics, which is not the case for any other dataset. We hypothesize this is because a few words were enough to detect sentiment accurately, leading the Snuba heuristics to perform extremely well.

For CDR, Snuba heuristics have lower recall than user-defined heuristics, since users could access knowledge bases through distant supervision while Snuba only relies on input primitives. Snuba heuristics therefore perform 16.29 F1 points worse than user-defined heuristics and distant supervision. Like other datasets, the end model improves recall and improves F1 score by 2.43 F1 points.

A.4 Multi-Class Extension

We use the iris dataset [14] to demonstrate that Snuba can also generate labels for multi-class classification tasks. We use four features associated with the task as primitives and report micro F1 scores for the generated training labels in Table 8. We use 25 labeled and 125 unlabeled datapoints and report scores for predictions on the 125 unlabeled datapoints. Note that Snuba performs worse than semi-supervised learning for this simple task, but matches the performance of a pruned decision tree and boosting. Moreover, since this task is simple, each datapoint received at least one label from Snuba heuristics, resulting in full coverage of the unlabeled set.

B. PROOF OF PROPOSITION 1

Proposition 1: *Suppose we have M heuristics with empirical accuracies $\hat{\alpha}$, accuracies learned by the generative model $\tilde{\alpha}$, and measured error $||\hat{\alpha}-\tilde{\alpha}||_\infty \leq \epsilon$ for all M iterations. Then, if each*

Table 7: Precision (P), Recall (R) and F1 scores for user-defined heuristics, Snuba-generated heuristics, and end model trained on labels from Snuba-generated heuristics. Lift reported is from user to Reef heuristics, then Reef heuristics to end model trained on labels from Snuba.

Application	User Heuristics			Reef Heuristics				Reef + End Model			
	F1	P	R	F1	P	R	Lift	F1	P	R	Lift
Bone Tumor	30.91	89.47	18.68	31.58	33.75	29.67	+0.67	71.55	58.86	91.21	+39.97
Mammogram	38.02	79.31	25.00	68.16	70.11	66.30	+30.14	74.54	61.80	93.91	+6.38
Visual Genome	34.76	98.28	21.11	46.06	48.10	44.19	+11.30	56.83	41.34	90.91	+10.77
MS-COCO	21.43	63.66	12.88	24.41	29.40	41.49	+12.98	69.52	55.80	92.16	+35.11
IMDb	20.65	76.19	11.94	46.47	48.03	45.52	+25.82	62.47	45.42	100.	+16.00
Twitter	36.17	30.91	43.59	48.15	31.71	100.	+11.98	78.84	75.40	82.61	+30.69
CDR	55.42	81.13	42.09	39.13	80.31	25.87	-16.29	41.56	32.11	58.96	+2.43

Table 8: Scores for training labels over unlabeled data for multi-class Iris dataset.

	Decision Tree	Boosting	Semi-Supervised	Reef
F1	95.20	95.20	96.80	95.20

heuristic labels a minimum of

$$N \geq \frac{1}{2(\gamma - \epsilon)^2} \log\left(\frac{2M^2}{\delta}\right)$$

datapoints at each iteration, the generative model will succeed in learning accuracies within $||\alpha^* - \tilde{\alpha}||_\infty < \gamma$ *across all iterations with probability* $1 - \delta$.

Proof: We first start by using the triangle inequality to bound the probability of the l_∞ norm error between the learned and true accuracies being larger than γ, $||\alpha^* - \tilde{\alpha}||_\infty \leq \gamma$.

$$\begin{aligned} P(||\alpha^* - \tilde{\alpha}||_\infty > \gamma) &\leq P(||\alpha^* - \hat{\alpha}||_\infty + ||\hat{\alpha} - \tilde{\alpha}||_\infty > \gamma) \\ &\leq P(||\alpha^* - \hat{\alpha}||_\infty + \epsilon > \gamma) \end{aligned} \tag{2}$$

where ϵ is the l_∞ norm error between the learned and empirical accuracies.

We then bound the probability of the l_∞ norm error between the empirical and true accuracies being greater than γ. By the union bound,

$$P(||\alpha^* - \hat{\alpha}||_\infty + \epsilon > \gamma) \leq \sum_{i=1}^{M} P(||\alpha_i^* - \hat{\alpha}_i|| + \epsilon > \gamma) \tag{3}$$

We rewrite $\hat{\alpha}_i$ for heuristics $i = 1, \ldots, M$ as

$$\hat{\alpha}_i = \frac{1}{N_i} \sum_{j : y_j \in \{-1,1\}} \mathbb{1}(\hat{y}_{ij} = y_j^*)$$

where $\hat{y}_{ij}$ is the label heuristic i assigned to datapoint j, y_j^* is the true label for datapoint j, and N_i is the number of datapoints where $\hat{y}_{ij} \in \{1, -1\}$ and did not abstain.

Substituting the above expression in (3), we get

$$\begin{aligned} &P\left(||\alpha_i^* - \hat{\alpha}_i||_\infty + \epsilon > \gamma\right) = P\left(||\alpha_i^* - \hat{\alpha}_i||_\infty > \gamma - \epsilon\right) \\ &\leq \sum_{i=1}^{M} P\left(|\alpha_i^* - \frac{1}{N_i} \sum_{j=1, y_j \in \{-1,1\}} \mathbb{1}(\hat{y}_{ij} = y_j^*)| > \gamma - \epsilon\right) \\ &\leq \sum_{i=1}^{M} 2 \exp\left(-2(\gamma - \epsilon)^2 N_i\right) \\ &\leq 2M \exp\left(-2(\gamma - \epsilon)^2 \min(N_1, \ldots, N_M)\right) \end{aligned} \tag{4}$$

where the second step uses Hoeffding's inequality and assumes that the datapoints are independent.

The above expression bounds the probability of the generative model failing *per iteration* of Snuba. To bound the probability of failure across all iterations, we use the union bound:

$$\begin{aligned} &P(||\alpha_i^* - \hat{\alpha}_i||_\infty > \gamma - \epsilon \text{ for any iteration}) \\ &\qquad \leq \sum_{i=1}^{M} P(||\alpha_i^* - \hat{\alpha}_i||_\infty > \gamma - \epsilon \text{ for one iteration}) \\ &\qquad \leq \sum_{i=1}^{M} p \\ &\qquad = Mp \end{aligned}$$

where $P(||\alpha_i^* - \hat{\alpha}_i||_\infty > \gamma - \epsilon) = p$ is the failure probability for a single iteration.

With the failure probability over all M iterations, $\delta = P(||\alpha_i^* - \hat{\alpha}_i||_\infty > \gamma - \epsilon \text{ for any iteration})$, we can express the failure probability of a single iteration as $p = \frac{\delta}{M}$. Substituting into (4), we get

$$\begin{aligned} \frac{\delta}{M} &= P\left(||\alpha^* - \tilde{\alpha}||_\infty > \gamma - \epsilon\right) \\ \delta &= MP\left(||\alpha^* - \tilde{\alpha}||_\infty > \gamma - \epsilon\right) \\ &\leq 2M^2 \exp\left(-2(\gamma - \epsilon)^2 \min(N_1, \ldots, N_M)\right) \end{aligned}$$