Asterisk: Generating Large Training Datasets with Automatic Active Supervision

MONA NASHAAT, AINDRILA GHOSH, and JAMES MILLER, Electrical and Computer Engineering, University of Alberta, Canada
SHAIKH QUADER, IBM Canada Software Lab, IBM Canada, Canada

Labeling datasets is one of the most expensive bottlenecks in data preprocessing tasks in machine learning. Therefore, organizations, in many domains, are applying weak supervision to produce noisy labels. However, since weak supervision relies on cheaper sources, the quality of the generated labels is problematic. Therefore, in this article, we present *Asterisk*, an end-to-end framework to generate high-quality, large-scale labeled datasets. The system, first, automatically generates heuristics to assign initial labels. Then, the framework applies a novel data-driven active learning process to enhance the labeling quality. We present an algorithm that learns the selection policy by accommodating the modeled accuracies of the heuristics, along with the outcome of the generative model. Finally, the system employs the output of the active learning process to enhance the quality of the labels. To evaluate the proposed system, we report its performance against four state-of-the-art techniques. In collaboration with our industrial partner, IBM, we test the framework within a wide range of real-world applications. The experiments include 10 datasets of varying sizes with a maximum size of 11 million records. The results illustrate the effectiveness of the framework in producing high-quality labels and achieving high classification accuracy with minimal annotation efforts.

CCS Concepts: • **Theory of computation** → **Active learning**; • **Computing methodologies** → **Supervised learning**;

Additional Key Words and Phrases: Machine learning, Heuristics design, Data labeling, Active learning

ACM Reference format:

Mona Nashaat, Aindrila Ghosh, James Miller, and Shaikh Quader. 2020. Asterisk: Generating Large Training Datasets with Automatic Active Supervision. *ACM/IMS Trans. Data Sci.* 1, 2, Article 13 (May 2020), 28 pages. <https://doi.org/10.1145/3385188>

1 INTRODUCTION

Organizations in different domains are increasingly investing in machine learning to empower their data-driven decisions. However, one of the most tedious tasks in creating machine learning models is obtaining hand-labeled training data, especially with the new revolutionary advances that deep learning methods bring to the field of machine learning. Since such techniques require

Authors' addresses: M. Nashaat (corresponding author), A. Ghosh, and J. Miller, Electrical and Computer Engineering, University of Alberta, 116 St and 85 Ave, Edmonton, AB T6G 2R3, Canada; emails: {nashaata, aindrila, jimm}@ualberta.ca; S. Quader, IBM Canada Software Lab, IBM Canada, 8200 Warden Ave, Unionville, ON L6G 1C7, Canada; email: shaikhq@ca.ibm.com.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2020 Association for Computing Machinery.

2577-3224/2020/05-ART13 \$15.00

<https://doi.org/10.1145/3385188>

large training datasets [1], the cost of labeling these datasets has become a significant expense for businesses and large organizations. In real-world settings, domain experience is usually required to accomplish or at least supervise such labeling processes; this makes the process of obtaining large-scale hand-labeled training data prohibitively expensive.

For these reasons, several researchers [2–7] have proposed techniques to generate training data with minimal annotation effort. One approach that aims at generating labeled datasets at scale is weak supervision [2]. In weak supervision, practitioners turn to noisy labels [3], which are programmatically generated using cheaper annotation sources such as crowdsourcing [4], external knowledge bases [5], and user-defined heuristics [6]. Previous research [6–9] has shown that weak supervision can produce less-than-ideal training datasets at a large scale for a wide range of applications. These labels can then be used to train many complex machine learning models, such as deep learning. Alternatively, other well-studied techniques rely on semisupervised learning [10, 11]. Semisupervised techniques exploit a small labeled set to derive assumptions about the data structure and leverage a larger unlabeled dataset. For this purpose, some techniques [11] employ the concept of generative models to utilize the unlabeled data and learn the data representation. Generative models produce samples after learning the underlying data distribution; these samples can then be used as training labels for discriminative models.

On the other hand, active learning (AL) [7] is a special kind of semisupervised learning that has been used for decades to achieve a high level of classification accuracy while optimizing the annotation cost. In AL settings, instead of manually labeling an entire dataset, an algorithm iteratively selects the most valuable points to classify and asks the user to only label these points. Although AL does not aim at producing labeled datasets, it helps in reducing the annotation cost while building machine learning models that generalize beyond the training data.

A closer look at these labeling techniques, however, reveals several gaps and shortcomings [12–16]. On the one hand, since cheaper annotation methods are used in weak supervision, these sources are expected to overlap and conflict, which affects the quality of the resulting labels [12]. To estimate the level of noise in the generated labels, previous studies introduce the data programming (DP) paradigm [2, 12], which uses generative models to integrate the outcome of multiple weak supervision. Nevertheless, the uncertainty levels originating from these weak sources can complicate the process of learning the structure of these generative models [12]. Moreover, these approaches require users to design a set of user-defined heuristics [6] to encode their domain experience, which can be an expensive and time-consuming process [13].

On the other hand, active learning can be expensive when applied to high-dimensional datasets [14]. In pool-based settings [17], the active learner performs an iterative process to choose one or more points from an unlabeled pool to query the user in each iteration. This iterative process involves ranking all the points in the unlabeled pool, selecting the points for which true labels should be provided, training a model, and evaluating its performance using a held-out test set. Therefore, any imbalance between the sizes of the unlabeled pool and the labeled dataset can affect the time complexity of the process and increase the annotation cost [14]. Moreover, other studies [15, 16] show that in situations where the unlabeled data points cannot be entirely separated, active learning does not provide much superiority over passive learning.

To overcome some of these challenges, we propose Asterisk, a framework to generate high-quality training datasets at scale. An overview of the system is presented in Figure 1. As shown in the figure, instead of relying on the end-users to write user-defined heuristics, the proposed approach exploits a small set of labeled data and automatically produces a set of heuristics (weak supervision sources) to assign initial labels. In this phase, the system applies an iterative process of creating, testing, and ranking heuristics in each and every iteration to only accommodate high-quality heuristics. Then, Asterisk examines the disagreements between these heuristics to

Fig. 1. An overview of the proposed system.

model their accuracies. To enhance the quality of the generated labels, the framework improves the accuracies of the heuristics by applying a novel data-driven AL process. During the process, the system examines the generated weak labels along with the modeled accuracies of the heuristics to help the learner decide on the points for which the user should provide true labels. The process aims at enhancing the accuracy and the coverage of the training data while engaging the user in the loop to execute the enhancement process. Therefore, by incorporating the underlying data representation, the user is only queried about the points that are expected to enhance the overall labeling quality. Then, the true labels provided by the users are used to refine the initial labels generated by the heuristics. As the figure shows, the refinement process can be repeated to further enhance the quality of the generated labels. Finally, the framework examines the refined labels and outputs a set of probabilistic labels that can be used to train any downstream classifier. A prototype implementation of the proposed framework is available at <https://github.ibm.com/Mona-Nashaat-Ali-Elmowafy/Asterisk>.

To evaluate the proposed method, we compare its performance with the performances of four state-of-the-art techniques including data programming [2], automated weak supervision [13], and traditional active learning strategies [17]. During the experiments, we report the labeling accuracy, annotation cost, and performance of the end model trained with the generated labels. The primary contributions of this research can be summarized as follows:

- An end-to-end labeling framework is proposed to create high-quality, large-scale training datasets. We describe the architecture of the proposed system, which includes a novel process of automatic generation of labeling heuristics instead of relying on the end-user to manually define the weak sources.
- We propose a data-driven active learning process to enhance the accuracy of the generated weak labels. The process learns the selection policy while taking the distribution of the underlying data and the labeling confidence to optimize user engagement.
- We applied a comprehensive set of experiments to evaluate the proposed method against state-of-the-art techniques. The experimental evaluation explores a wide range of domains with 10 datasets that vary in size and dimensionality with a maximum size of 11M records. We also use a real-world business dataset of 1.5M records provided by our industrial partner, IBM. The experiments also include a micro-benchmarking to evaluate the individual components of the proposed approach.

The remainder of the article is structured as follows: Section 2 presents the background related to this research. Section 3 states, in detail, the design of the proposed solution. Section 4 presents

the performed experiments and reports the obtained results. Section 5 discusses related work, and Section 6 concludes the article.

2 BACKGROUND

In this section, we first review weak supervision and the methods of automating weak supervision sources. In the second subsection, we discuss active learning and, more specifically, different approaches for meta-active learning.

2.1 Automated Weak Supervision

In weak supervision, domain experience is encoded in the form of high-level, low-quality sources such as user-defined heuristics. Each of these sources is then used to automatically generate noisy (weak) labels for an unlabeled dataset. Since these weak supervision labels are collected from sources with different coverage and accuracies, the main challenge of weak supervision is to combine these conflicting sources into a single label for each data point. To overcome such a challenge, researchers [2, 6, 12, 18, 19] try to estimate the accuracy of different weak supervision sources and use these estimates to produce combined labels. Most of these efforts [6, 18, 19] utilize generative models to assess the accuracies of multiple weak supervision sources and model the true label as a latent variable based upon a set of noisy observations. After modeling the accuracies, the generative model can output a set of probabilistic labels to work as the training dataset for any discriminative model.

Aiming at aggregating labels from different supervision sources, the data programming paradigm [2] learns the accuracy of different weak supervision sources by examining the disagreements between them without the need for any ground truth. Data programming allows users to encode their domain experience using an ensemble of labeling functions [6]. The abstract concept of the labeling functions in data programming supports a wide range of weak supervision sources, including crowdsourcing and external knowledge bases. To denoise these sources, data programming builds a generative model to examine the dependency structure among these labeling functions and model their accuracy. However, other studies [20, 21] state that it can be difficult to estimate the noise level in the generated labels from data programming. Also, since the quality of the labels produced from the generative model affects the subsequent models being trained, it is essential to allow the user to debug and trace the output of the generative models [20]. However, this also can be a challenging task given the complex structure of such models, especially when the weak sources show a high level of dependencies [12].

Moreover, as the success of data programming depends on the quality of the weak supervision sources encoded as labeling functions, some research [13, 22] argues that the task of writing these labeling functions can be monotonous for end-users. Therefore, recent studies [13, 22] try to automate the process of creating weak supervision sources. For example, the authors in [13] present a system that can automatically generate weak supervision sources for an unlabeled dataset using a small labeled set. The system uses the small labeled dataset to iteratively create heuristics and tries to terminate this iterative process before the quality of the generated labels degrades. Also, Das et al. [22] propose an affinity coding paradigm that infers true labels of an unlabeled dataset by examining the similarity between the unlabeled points. The proposed system [22] derives the affinity scores from convolutional neural networks and uses these scores as signals to decide upon class membership.

However, although these efforts have illuminated the importance of automating weak supervision sources, there are some potentially open questions about the applicability of these techniques to real-world cases with large-scale datasets. For example, the proposed system in [13] was evaluated using a set of datasets with varying sizes but with a maximum size of 100K unlabeled

points. Also, the system presented in [22] was only evaluated using image classification tasks with a maximum size of 37,322 images. Therefore, we believe that the scalability of automated weak supervision is yet to be explored in real-world tasks with millions of unlabeled records, which represents one of the main motives driving our research.

2.2 Meta-Active Learning

Active learning optimizes the process of data collection needed to train a classifier by deciding upon which instances an oracle should label. In our research, we focus on the pool-based settings [17] in which a classifier is initially trained using a small set of labeled points (the seed). Then, the active learning algorithm iteratively selects one or more points from an unlabeled pool and asks the user to provide the correct labels for these points, and then it adds them to the labeled set to retrain the model. The model is then evaluated using a held-out test set, and the process is repeated to label more points until a target performance is reached or a predefined annotation budget is exhausted. The algorithm that decides on which points should be labeled is called the query strategy, and it is an essential part of the active learning process. Over the last decades, many query strategies have been proposed for different classification tasks [7, 17]. One of the most effective query strategies is uncertainty sampling [17]. The algorithm ranks the data instances in the unlabeled pool and chooses the point about which the current classifier is most uncertain. Another efficient algorithm is query-by-committee [17], which employs a committee of classifiers and selects the points about which the committee members disagree. Since these two methods tend to choose the points that lie on the classification boundary [17], they are known to be prone to select outliers. Therefore, to ensure that the selected points can be seen as representatives of other instances in the distribution, previous studies [23, 24] propose density-weighted uncertainty sampling in which the uncertainty sampling algorithm is augmented to consider both uncertainty and density measures simultaneously.

Although these algorithms have performed remarkably well in various tasks [17, 25, 26], previous studies [27–30] have pointed out that these strategies can be limited when dealing with different data distributions. Since these strategies apply fixed heuristics to measure the informativeness of the unlabeled points, they do not employ characteristics specific to the underlying learning problem [27]. Therefore, various factors, such as imbalanced classes and label noise, can make uncertainty sampling result in suboptimal decisions [30]. As a result, to overcome these limitations, recent studies [27, 29–32] propose the use of meta-active learning. In meta-active learning, the choice and the design of the query strategy change depending on the underlying data distribution. For example, some studies [26, 31] propose combining existing query strategies to reduce overfitting [26] or transfer the active learning experience [31]. Alternatively, other research [27, 29, 30, 32] goes further and tries to learn the query strategy; for instance, the technique in [29] structures the process of designing the query strategy as a reinforcement learning problem to learn the data selection policy. Similarly, Konyushkova et al. [30] propose an active learning algorithm in which the query strategy is replaced with a regression function to decide which points to label. Also, Pang et al. [32] model the query strategy algorithm as a deep neural network to predict the next best point to query the user.

However, most of these studies have been only applied to certain domains and specific classification settings; for example, the techniques proposed in [30, 32] only consider a binary base classifier. Also, the research in [30] tries to learn an active learning strategy for the aim of transferring query strategies to new domains; hence, their experiments focus more on the computational cost rather than reducing the annotation budget. Moreover, none of these works has been evaluated using massive datasets with millions of records. For instance, the approach in [29] could cut the annotation cost by up to 30% with a labeling budget of 200 instances. Similarly, the technique

in [32] achieved more than 95% accuracy on a dataset with 70,000 records. Also, the approach in [30] could reach 95% accuracy with a dataset of 3,190 records. In short, among all these algorithms [29, 30, 32] that aim at learning the strategy of AL, a maximum of 70,000 records [32] was used in the evaluation. Also, with the increasing popularity of weak supervision, a critical research question rises, which is whether any of these meta-active learning techniques can work with the high level of noise in the labels collected from the weak sources.

3 ASTERISK ARCHITECTURE

In the following two subsections, we describe the architecture of the proposed system. While Section 3.1 formulates the input and output for Asterisk, Section 3.2 describes in detail the individual components of the proposed system.

3.1 Input and Output

The input to Asterisk is an unlabeled dataset D_U of size N and a small labeled dataset D_L with size M , where $M \ll N$. The unlabeled dataset D_U consists of data points that are described as $\{\mathbf{x}_i, y_i\}_{i=1}^N$, where $\mathbf{x}_i$ represents a set of features describing the i th observation (data point) in the dataset and y_i is the unknown label associated with this point. Similarly, data points in the labeled dataset D_L are defined with a set of points $\{\mathbf{x}_i^*, y_i^*\}_{i=1}^M$ denoting the set of features $\mathbf{x}_i^*$ and the corresponding known label y_i^* that describe the i th data point. Both $\mathbf{x}_i, \mathbf{x}_i^* \in \mathbb{R}^F$ are viewed as F features representing the data. Features are a set of measurable properties of the observed data points. Hence, a set of numerical values describing the i th point can be described by a feature vector $\mathbf{x}_i$. For example, if the classification task is to predict the default payment of a client, $\mathbf{x}_i$ can be a set of numeric features describing the previous credit payments. For the sake of simplicity, we consider the binary classification situation, hence, $y_i, y_i^* \in \{-1, 1\}$. As for the output data, the proposed system generates probabilistic training labels $\bar{y} = P[y = 1] \in [0, 1]$ for the points in the unlabeled dataset D_U , which can be further used to train any noise-aware classifier.

3.2 Asterisk Design

The proposed system exploits the small labeled dataset D_L to produce a set of probabilistic labels $\bar{y}$ for the data points in the unlabeled dataset D_U . An overview of the system is shown in Figure 2. As the figure depicts, the system consists of three main components, namely *the heuristics generator*, *the data-driven learner*, and *the probabilistic labels generator*. The heuristics generator component (Section 3.2.1) aims to automatically produce a set of heuristics to assign initial labels to the points in D_U . The second component, the data-driven learner (Section 3.2.2), works with the outcomes of the first component to further examine the data and refine the initial labels. This component aims to enhance the accuracy of the generated heuristics and increase the coverage of the generated training labels. Therefore, the component tries to economically engage the user to express his or her domain experience and uses his or her input in the refinement process. Finally, the probabilistic labels generator (Section 3.2.3) is used to learn the accuracy of these labels and assign a single label for each data point in D_U .

As the figure shows, the process of denoising and generating the final labels (the second and the third components) can be repeated to enhance the accuracy of the final labels. Since each iteration outputs an improved set of heuristics (from the second component) and a refined set of probabilistic labels (from the third component), the user can decide to initiate another cycle where these outputs are fed to the data-driven learner component to enhance the quality of the heuristics. Then, the label generator component can be used to produce more accurate labels. However, this requires increasing the budget of manual labeling since the user will be queried to label more points to help with the refinement process. Nonetheless, we found that running only one iteration of the process

Fig. 2. Component overview of the Asterisk framework.

can help obtain a satisfactory level of classification accuracy for real-world tasks (Section 4) and achieve labeling accuracy of 90.09% on average (Section 4.2.1).

3.2.1 Heuristics Generator: Automating the Heuristics Production. The system starts with the heuristics generator component, which takes the labeled set D_L and the unlabeled set D_U as inputs and outputs a set of heuristics H of size K denoted as $(h_1, h_2, \dots, h_k)$ and a vector of initial probabilistic labels $\bar{y}_{\text{initial}} \in [-1, 1]^N$ for the N points in D_U . Each heuristic in H follows the form $h_j(\hat{x}_i) \rightarrow y_i \in \{-1, 0, 1\}$, where $\hat{x}_i$ is a subset of the F features and is used as an input to h_j , and y_i is the weakly supervised label for the i th point.

Since the heuristics exploit the labeled data in D_L to output labels for D_U , the examples provided in D_L are assumed to be handled by subject-matter experts and hence have a strong ground-truth value. In this component, we treat the process of generating heuristics as a process of creating a set of probabilistic classification models that take one or more features as input and calculate probability distribution over a set of classes [33]. The process utilizes D_L to train and evaluate the generated heuristics. Hence, if there is a high level of noise in the examples provided in D_L , this can affect the process of creating the initial heuristics.

Nevertheless, applying some noise-filtering techniques [3, 12] can help to eliminate the noise effect. Moreover, the proposed method utilizes meta-active learning in the second component to enhance the quality of the generated labels, which will help reverse the noise effect.

Then, the component uses the distribution provided by each heuristic to either assign labels to the unlabeled dataset (i.e., assigns either -1 or 1) or abstain (i.e., outputs 0). More specifically, in our implementation, we use an ensemble of decision stumps [34] as the inner classification model to mimic the threshold-based heuristics that users usually write [6, 13].

To create the final set of heuristics H , we follow the iterative process shown in Algorithm 1. As the algorithm indicates, the process starts with *defining the input* (features) for the potential models (steps 1–3). Then, the process continues with *creating the models (heuristics)* (steps 5–8)

ALGORITHM 1: The Procedure of the Heuristics Generator Component**Input:** Unlabeled Input dataset D_U , Small Labeled Input Dataset with Ground truth labels D_L **Output:** Set of Heuristics H and a vector of initial probabilistic labels $\tilde{y}_{\text{initial}}$

- 1: compute C as the maximum number of features for the heuristics (Equation (1))
- 2: **for** $\tilde{F} = 1 \dots C$ **do**
- 3: $F_combinations =$ all distinct subsets of features of size $\tilde{F}$ from the original set of features F
- 4: **for** $j = 1 \dots \text{length}(F_combinations)$ **do**
- 5: $\tilde{x} = F_combinations[j,:]$
- 6: use $\tilde{x}$ as input to build a heuristic h_j using an ensemble of decision stumps
- 7: create a heuristic model $h_j = \sum_{m=1}^M \omega_m f_m(x)$
- 8: apply h_j to D_L and produce predictions as y_j^*
- 9: $P_j, R_j, MCC_j = \text{calculate_performance}(y_j^*, y^*)$ (Equation (3))
- 10: apply h_j to D_U and compute the conditional probability $P(y_i = +1 | \tilde{x}_i)$
- 11: estimate the confidence interval for h_j
- 12: use the confidence interval to force h_j to abstain from labeling low confidence labels
- 13: calculate Hamming distance for h_j to estimate coverage
- 14: compute Rank_j for h_j (Equation (2))
- 15: add the heuristic with the highest Rank to H
- 16: use a generative model to learn the accuracies of H and produce $\tilde{y}_{\text{initial}}$
- 17: **return** $H, \tilde{y}_{\text{initial}}$

and *evaluating their performance and coverage* (steps 9–14). Finally, the process *ranks the heuristics* generated by each and every iteration to decide upon which heuristic to add to the set H (steps 14–17); further details about these steps are presented as follows:

Defining the Input. First, to choose the input features for the heuristics (models), we iterate over a range $\tilde{F}$ from 1 to C , where C is the maximum number of features that can be used as input to the heuristics. In each iteration, the system generates distinct subsets of features of size $\tilde{F}$. These subsets can be denoted as $\binom{F}{\tilde{F}}$, where $\tilde{F}$ is the value assigned to the size of the input of the heuristics generated in this iteration. Although the users can define the maximum size C , based on the insights obtained from the conducted real-world experiments (Section 4), we adjust a default value for C as

$$C = \ln(F) + 1 \quad (1)$$

to bound the number of iterations when the number of features F grows continuously. In other words, for high-dimensional data, the proposed method tries to control the growth rate at which C expands as the number of features increases. At this point, the component aims to limit the number of inputs to the heuristics, so it does not affect the computational complexity of the proposed method. On the other hand, the component specifies a lower bound for C as one input feature for all the possible values of F where $F \geq 1$.

The inner classifiers are implemented as an ensemble of decision stumps that try to split the training examples in D_L into two subsets based on the values of one or more features in the data. So, each classifier defines the input features and a set of thresholds to classify the training data into two groups.

Creating Heuristics. To define the input for each heuristic, the component proceeds with designing a heuristic (model) for each possible combination resulting in $\sum_{\tilde{F}=1}^C \binom{F}{\tilde{F}}$ models. As for the threshold, each classifier finds the best threshold that fits the training examples and gives the best accuracy over D_L . The generated classifier in each iteration can be formulated as

$h_j = \sum_{m=1}^M w_m f_m(x)$, where M is the number of decision stumps, w_m are the learned coefficients, and $f_m(x) \in \{-1, 1\}$ denotes a single decision stump as $f(x) = s(x_k > T)$, where $s \in \{1, -1\}$, x_k is the k^{th} element in $\hat{x}$, and T is the threshold specified by the decision stump.

Moreover, like other probabilistic classification models, the generated heuristics estimates the conditional probability of a class label $P(y_i = +1 \mid \hat{x}_i)$. The component forces the models to abstain when they are not confident about a generated label. To decide on the abstaining interval, the component examines the confidence interval for each heuristic h_j using D_L as a validation set [35]. The confidence level is adjusted according to the coverage of the heuristic (i.e., the percentage of D_U that receives a label from h_j). Since the confidence level denotes the degree at which the generated labels represent the distribution of D_U [35], we rely on the coverage achieved by each heuristic to determine the confidence level. Then, the component adjusts the abstaining interval accordingly as $\{a \mid 0 \leq a \leq 1 \text{ and } a \notin \text{CI}\}$, where CI is the confidence interval for h_j . This way, the component obliges the heuristics to only output labels for datapoints where they have high confidence, which helps to increase the accuracy of the generated heuristics.

Evaluating Heuristics. When evaluating the performance of the heuristics produced during each iteration, the component also considers the overall coverage of the heuristics when applied to D_U . The component aims to widen the range of the data points that receive labels from H in D_U . In other words, the goal of the component is to output a set of heuristics that are individually accurate while achieving high labeling coverage when combined. Therefore, to estimate the performance of the heuristics, the system computes Precision (P), Recall (R), and MCC metrics for heuristics generated during each iteration. The performance metrics are computed by applying each heuristic to D_L . Since the generative model [6] assumes that weak sources encoded by the users always perform better than random, the component holds this assumption by only including heuristics with MCC values greater than 0.60. Alternatively, to evaluate the coverage of a heuristic h_j , we examine the dissimilarity between the data points in D_U that are labeled by h_j and the points that already received labels from H . To compute the dissimilarity, we construct a vector $v_j \in \{0, 1\}^N$, which represents whether each point in D_U receives a label from heuristic h_j (1) or not (0). Then, we obtain another vector $v \in \{0, 1\}^N$ to represent whether any heuristic in H has assigned a label to the data points in D_U . Next, we compute the Hamming distance [36] between v_j and v and use it as a measure for the coverage of h_j .

Ranking Heuristics. After that, heuristics generated during each iteration are ranked based on performance (i.e., Recall, Precision, and MCC) and coverage (i.e., the Hamming distance) to decide on which heuristics to add to the final set. The ranking uses a weighted average of the performance metrics and the coverage distance as

$$\text{Rank}_j = \omega \times f_1(R, P, \text{MCC}) + (1 - \omega) \times f_2(v_j, v), \quad (2)$$

where f_1 is the harmonic mean of R, P, and MCC. The value of MCC [52] is adjusted since the harmonic mean is only calculated for positive real numbers. Therefore, f_1 is computed as

$$f_1(R, P, \text{MCC}) = ((R^{-1} + P^{-1} + (\text{MCC} + 1)^{-1})/3)^{-1}, \quad (3)$$

and f_2 is the function to calculate the Hamming distance for h_j defined in [36], and $\omega = 0.5$ to indicate equal weight between the coverage and the performance. Then, the component only chooses the highest-ranking heuristic to add it to the set H .

Finally, to combine the output of the heuristic and generate an initial vector of probabilistic labels $\bar{y}_{\text{initial}}$, we employ a generative model [12] to learn the accuracies of the heuristics in H and estimate any statistical dependency between their outputs. Then, the generative model employs the learned accuracies to produce a single probabilistic label for each data point in the unlabeled dataset.

It is essential to mention that, since the component encourages the heuristics to abstain from labeling points with low confidence labels, there might be a subset of points in the unlabeled dataset that do not receive a label from any heuristic in H , especially when $M \ll N$, which is the case in many real-world problems. On the one hand, this means that the final set of heuristics generates more accurate labels than other methods [6] since it does not produce low-accuracy, high-coverage heuristics. On the other hand, suppressed coverage may pose a problem, especially when a large amount of accurate training data is needed. Therefore, the data-driven learner component, explained in the next subsection, tries to reverse the effect of abstaining while further refining the generated labels.

3.2.2 Data-Driven Active Learner: Utilizing the Generative Model Output in AL Ranking. As mentioned earlier, the heuristics generator component outputs two outcomes: a final set of heuristics H and a vector of initial probabilistic labels $\bar{y}_{\text{initial}}$. The heuristics set H can be denoted by a sparse matrix of weakly supervised labels as

$$h_{i,j} = h_j(x_i) \text{ where } 1 \leq i \leq N, 1 \leq j \leq K. \quad (4)$$

Conversely, the vector of probabilistic labels $\bar{y}_{\text{initial}}$ represents how confident the generative model is about the assigned labels. For example, if a data point did not receive a label from the heuristics set H , the generative model will have a probabilistic label for this point that is equal to $P[y_i = 1] = 0.5$, which represents an equal probability for the data point being of either class. Moreover, when the generative model assigns a probabilistic label $P[y_i = 1]$ close to 0.5 to a data point x_i , this indicates a point with low confidence labels, which may happen when many heuristics with similar accuracies disagree on the label for that data point. Formally, we define low confidence labels as

$$|P[y_i = 1] - 0.5| \leq \alpha, \quad (5)$$

where $P[y_i = 1]$ is the probabilistic label assigned by the generative model and α is a threshold to ensure that the definition of low confidence changes according to the number of the generated heuristics in H . α is denoted as

$$\alpha = 0.3 - \left(1/e^{\sqrt{K+1}}\right), \quad (6)$$

where K is the number of heuristics generated in the first component. It is important to know that as the number of heuristics becomes larger, the value of α is expected to approach an initial value denoting fewer data points with low confidence labels (with $P[y_i=1]$ close to 0.5). Also, the formula specifies a value of 0.3 as the initial value before measuring the exponential decay as K increases. The value is determined based on the insights obtained from the experiments as the value succeeds to capture the right range of low confidence labels with a lesser number of heuristics. In other words, we expect to have fewer data points with low confidence labels when more heuristics are generated. Thus, the learner component uses the formula above to classify the data points in D_U into two groups, namely points with high confidence labels D_{HC} , where $D_{HC} \subseteq D_U, \forall x_i \in D_{HC} \{x_i | |P[y_i = 1] - 0.5| > \alpha\}$, and points with low confidence labels D_{LC} with size L_c , where $D_{LC} \subseteq D_U, \forall x_i \in D_{LC} \{x_i | |P[y_i = 1] - 0.5| \leq \alpha\}$. The learner component aims at eliminating the second group by replacing the low confidence labels with more accurate ones.

To accomplish such a goal, the component tries to integrate the user in the loop at this point by employing active learning. However, our problem settings do not impose traditional active learning scenarios where we usually have a small set of labeled points and a larger set of unlabeled data. Instead, we deal with a set of probabilistic labels that are classified based on the confidence of the generative model. Therefore, we adopt meta-active learning in this component and propose a data-driven approach to learn the query strategy. The approach formulates the process of designing the

Fig. 3. An overview of the data-driven active learner.

query strategy as a regression problem. We train a regression model to predict the reduction of the generalization error associated with adding a labeled point $\{x_i, y_i\}$ to the training data of a classifier. The idea of implementing machine learning to decide or develop the selection policies in active learning has been applied to many situations in the literature [29–31] and proved to achieve a competitive performance against traditional querying strategies. Therefore, our main hypothesis is that this regressor can serve as the query strategy in our problem settings to outperform the baseline strategies since it is customized to the underlying distribution and considers the output of the generative model.

Accordingly, the component consists of two main processes: first, *designing the AL query strategy* that fits the data distribution for a given problem, and second, applying it to the D_{LC} as a *data-driven AL process*. The overall structure of this component is illustrated in Figure 3. Detailed descriptions about each of the two processes are given in the rest of this section.

Designing the Data-Driven Strategy. Since the task of designing the AL query strategy is framed as a regression problem, the goal is to train a regressor that, when applied to a set of points, chooses the point that results in the maximum reduction to the generalization error. To set up the regression process, we need a set of labeled observations to train the regressor. Formally, at this point, we aim at creating a training dataset D_{reg} of size Q , which can be described as $\{y_i, \nabla_i\}_{i=1}^Q$, where y_i represents a set of features describing the i th observation in D_{reg} . In our implementation, we consider features that are specific to the data distribution and represent the state of the points in D_U . Therefore, we utilize the probabilistic label $P[y_x = 1]$, which is assigned to the point x by the generative model, the distance to the closest point in the dataset, and the distance to the nearest labeled point. Alternatively, ∇_i represents the label associated with the i th point in D_{reg} , which is the potential reduction to the generalization error after annotating this point and adding it to the labeled dataset.

Creating the Training Data for the Regressor. Therefore, to collect these observations and create D_{reg} , we design an experiment in which we iteratively train a classifier and evaluate it to record the corresponding generalization errors. In this scenario, we use both the labeled dataset D_L along with the points with the high confidence labels produced by the generative model D_{HC} . After combining these two datasets, we split them into a training set D_{Train} , and testing set D_{Test} . Then, we split D_{Train} into a labeled dataset D_{Train_L} of size S and an unlabeled dataset D_{Train_U} consisting of the remaining points. Next, we train a classifier with D_{Train_L} , resulting in a model m_s that can be used to output class labels for the data point in D_{Test} and calculate the corresponding classification loss L_s using the test data D_{Test} . After that, we iteratively select a new point x from D_{Train_U} and add it to the labeled set as

$$D_x = D_{\text{Train}_L} \cup \{x\}. \quad (7)$$

Then, we use D_x to train the classifier again and create a new model m_x , test the new model using D_{Test} , and calculate the new classification loss L_x . We then record the reduction in the classification loss associated with adding x to the training set as

$$\nabla_x = L_s - L_x. \quad (8)$$

Moreover, as we are recording the reduction in the generalization error ∇_x associated with labeling each point x , we compute the set of features γ_i that represent the point status in D_U . Finally, to construct the final dataset, we repeat this experiment using different initializations of D_{Train_L} with varying sizes of $S \in \{S_{\min}, \dots, S_{\max}\}$. Based on the insights obtained from the experiments (Section 4), we repeat the process with different sizes equal to 70%, 80%, and 90% of the total size of D_{Train_L} . Although S can be initialized with any range of sizes, we find that these values result in enough data points to learn the query strategy while sustaining an acceptable computational cost.

Based on the insights obtained from the experiments, the process of training the regression function does not impose a high cost on the active learning process. The total time required for learning the AL strategy for a dataset of 11M records was less than 15 minutes on an Intel i7 machine with 32 GB RAM. Also, for training the regression function for more massive datasets, the user can always adjust the sampling range to reduce the number of iterations and speed up the process. As for applying the query strategy, the component runs in $O(I \cdot L_c)$, where I is the number of queries consumed by the AL component, and L_c is the size of the unlabeled pool D_{LC} . Consequently, during each iteration, we randomly sample S points from D_{Train} and record the characteristics γ_x of different data points along with their reductions γ_x to the generalization error. This results in the new training dataset D_{reg} that is used later to train the regressor.

Training the Regressor. As the data-driven learner aims at enhancing the quality of the labels in D_{LC} , we use D_{reg} to train a regression function g to predict the potential error reduction of annotating the instances in D_{LC} . Although the distribution in D_{LC} is different from the distribution of D_U , at this point, we aim at creating an active learning strategy that considers the distribution of the unlabeled pool, which in this case, is D_{LC} . Therefore, in our implementation, a random forest regressor is used and trained using D_{reg} . Although the regressor can be implemented using any regression function, random forest regression is applied since it maintains high accuracy for large high-dimensional data while preventing overfitting. Since the random forest regressor requires a set of meta-parameters, we use cross-validation to define a grid of hyperparameter ranges and use it to choose the best model that reduces the chance of overfitting. Then, we treat the output model as our query strategy that is built while considering the outcome of the generative model. The resulting policy can be used to greedily select data points with the highest potential error

ALGORITHM 2: The Process of Designing the Data-Driven Strategy**Input:** The vector of probabilistic labels $\bar{y}_{\text{initial}}$, the labeled input dataset DL**Output:** A regressor model g

- 1: use $\bar{y}_{\text{initial}}$ to classify DU into DHC and DLC
- 2: initialize dataset D = DL $\cup$ DHC
- 3: split D into DTrain and DTest
- 4: **for** S in {Smin, ... Smax}do
- 5: split DTrain_L into DS of size S and DTrain_U
- 6: train a classification model mS
- 7: calculate the test loss LS
- 8: **for** each point x in DTrain_U **do**
- 9: form a new dataset Dx = DTrain_L $\cup$ {x}
- 10: train a classifier model mx
- 11: calculate the new test loss Lx
- 12: calculate the error reduction $\nabla x = Ls - Lx$
- 13: collect the data point parameters γx
- 14: return the labeled data point $\{\gamma x, \nabla x\}$
- 15: return Dreg of size Q as $\{\gamma x, \nabla x\}$
- 16: train and evaluate a random forest regressor g using Dreg
- 17: return g

reduction by taking the maximum of the value predicted by the regressor g as

$$x^* = \arg \max_{x \in D_{\text{Test}}} g(\gamma x). \quad (9)$$

Furthermore, the whole process is explained in Algorithm 2.

Data-Driven AL Process. The regression function g is then applied to rank the points in D_{LC} . Since the number of data points in D_{LC} is much smaller than the number of points in D_U ($L_c \ll N$), the ranking time is highly reduced. Moreover, to overcome the cold-start problem in active learning [37], the component uses the dataset with the high confidence labels D_{HC} along with the labeled dataset D_L to form the initial seed and the test set. The initial seed is used to train the classifier at the beginning of the active learning process, while the test set is used to evaluate the classifier after each iteration [17]. The component also allows the user to specify a budget B for the maximum number of points that can be manually labeled using g. The output of this component is a labeled set D_{AL} that can be described as $\{x_i, y_i^*\}_{i=1}^{\min(B, L_c)}$. In other words, D_{AL} represents the data points x in D_U that are selected by g to receive true labels y^* from the user. Since the active learning process only terminates when either all the points in D_{LC} receive a true label from the user or the labeling budget B is exceeded, the size of D_{AL} is denoted as $\min(B, L_c)$. Finally, the component uses this set to refine the sparse matrix H as

$$H_{\text{updated } i,j} = \begin{cases} y_i^* & \text{if } (x_i, y_i) \in D_{AL} \\ L_{i,j} & \text{otherwise} \end{cases} \quad j = 1, 2, \dots, K. \quad (10)$$

The matrix H_{updated} is used as an improved version of H. By utilizing these processes, a portion of the low-confidence labels are replaced by true labels. As mentioned earlier, the low-confidence points are originated when the heuristics either abstain from labeling or disagree on specific points. Therefore, the learner component enhances the quality of the labels by trying to eliminate the abstaining effect and resolve the disagreements between the heuristics to increase their accuracies.

3.2.3 Probabilistic Labels Generator: Aggregating the Output of Different Heuristics. The final component of Asterisk is the label generator, which aims at learning the accuracies of the

generated heuristics using the refined heuristics matrix H_{updated} and then combines all the output of these heuristics to produce a single probabilistic label $\tilde{y}_i$ for each point in D_U . This process is accomplished by learning the structure of a generative model Gen [12], which utilizes the refined matrix to model the process of labeling the training set. Since the generative model treats the final label Y^* as a latent variable, it learns the distribution over the labels generated by each heuristic in H_{updated} . To learn the accuracies of the refined heuristics and the correlations among them, the generative model maximizes the L_1 regularized marginal pseudolikelihood [12] of the output of each heuristic in H_{updated} . The process uses the agreements and disagreements between the refined heuristics to encode the generative model as a factor graph [38]. It employs three factors, which are labeling accuracy, labeling propensity, and the heuristics pairwise correlations. These factors formally define our model as [6]

$$\text{Gen} : \pi_{\varphi}(H_{\text{updated}}, Y^*) = \frac{1}{Z_{\varphi}} e^{\vartheta^T H_{\text{updated}} Y^*}, \quad (11)$$

where ϑ denotes the accuracy of the heuristics and represents the factor graph, and Z_{φ} is a partition function to ensure π is a distribution. As a result, the generative model Gen employs a distribution to describe the relationship between the heuristics H_{updated} and the latent variable for the true label Y^* . Hence, after learning the relative accuracies of the heuristics, the generative model can estimate $P(H_{\text{updated}} | Y^*)$ by combining their output into a single label for each data point.

As Figure 2 depicts, the processes of updating the heuristics and generating the final probabilistic labels are iterative. Therefore, at this point, the user is informed about the performance of the final heuristics, the coverage obtained in D_U , the status of the generated probabilistic labels such as the number of low-confidences labels, and the number of true labels consumed so far. Then the user decides to either terminate the process or initiate another cycle to further refine the output labels. The output of the generative model can then be used to train any noise-aware discriminative model to generalize beyond the generated observations.

4 EVALUATION

To evaluate the proposed method, we run a set of experiments to compare Asterisk to other labeling approaches. The experiments are threefold and seek to validate the following claims:

- **Labels from Asterisk outperform labels produced by weak supervision sources that are automatically created.** We compare Asterisk to another approach that automatically generates weak supervision sources and produces training labels [13]. During the experiments, we consider both the accuracy of the generated labels (compared to the ground truth) and the performance of the end model. As for the accuracy of the generated labels, Asterisk outperforms this method by 14.84% on average (labeling accuracy). Also, the experiments show that the proposed method improves the classification accuracy by 7.15% on average (end-model accuracy) when compared to this approach.
- **Labels generated from Asterisk outperform labels generated using user-defined weak supervision sources.** We compare Asterisk to other methods that allow users to express their domain experience in the form of labeling functions [6, 38]. The experimental results show that the proposed system outperforms these tools in labeling accuracy by 34.20% on average. The proposed method also enhances the learned accuracy of the generative models by 10.42% on average.
- **Labels generated from Asterisk outperform labels generated using active learning.** We compare Asterisk to baseline active learning techniques. The experiments consider four different active learning query strategies [7]. Asterisk enhances the classification accuracy by 4.15% on average while reducing the labeling cost by up to 52.77%.

Table 1. Datasets Statistics

Datasets	Domain	Data Size	F	+ / Size	M	N
Higgs	Physical	11,000,000	28	52.96	440,000	10,560,000
Renewal Sales	Business	1,354,704	11	73.06	54,188	1,300,516
Rain Prediction	Weather	142,000	24	22.42	7,100	134,900
Travel Insurance	Business	63,300	11	14.60	3,165	60,135
Bank	Business	45,211	17	11.70	2,261	42,950
News	Social	39,797	61	20.38	1,990	37,807
Credit Card	Business	30,000	24	22.12	1,500	28,500
Occupancy Detection	Physical	20,560	7	23.10	1,028	19,532
Magic	Physical	19,020	12	35.16	951	18,069
MNIST	Image	70,000	784	-	3,500	66,500

Data size is the number of records each dataset has. F is the number of features used to create the labels and train the final classifier. + / Size is the percentage of the positive class to the dataset size. M, N are the sizes of labeled and unlabeled datasets, respectively.

Although the experimental evaluation considers a wide range of 10 classification tasks and reports different evaluation metrics, the reader should not overinfer from those 10 samples. The results obtained from the experiments depict that the proposed method achieves competitive labeling results along with adequate classification performance. However, these samples cannot describe the entire sampling frame for this problem space. Hence, it is wrong to assume that the algorithm will provide significantly superior performance in every situation. Therefore, even though the proposed method outperforms both semisupervised learning and weak supervision techniques, for other tasks and with different problem settings, other paradigms such as unsupervised learning models may result in more superior performance.

The section is divided into three subsections. In the first subsection, we discuss the experimental setup. Next, we report the results of comparing Asterisk to other labeling methods. Finally, in the third subsection, we evaluate the individual components of the proposed system by experimenting with two variations of Asterisk to assess the effect of each component on the final model performance.

4.1 Experimental Setup

This section describes the datasets used in the experiments, the baseline methods, and the implementation details.

4.1.1 Datasets. We consider real-world applications and tasks over open source tabular datasets. Summary statistics are provided in Table 1. We examined classification tasks for various domains, including business, physical, social, and multiclass image classification.

Business. We use four business datasets in the experiments. First, we employ a real-world business dataset, **Renewal Sales**, that is collected from our industrial partner, IBM. The data contains more than 1.3 million records and is used to classify renewal risks in which clients decide not to renew their software licenses [39]. Second, we use the **Travel Insurance** dataset, which is collected from a third-party travel insurance company that is based in Singapore. The dataset has more than 63K records and is used to detect insurance claims. Another business dataset is the **Bank** [40] dataset, which is a business dataset that contains more than 45K instances. It represents direct marketing campaigns using phone calls of a banking institution and is used in a classification task to predict if the client will subscribe to a term deposit. Finally, we include the **Credit Card** dataset [41], which is a business dataset of 30K records for customers' credit card payments. The classification task of this dataset is to predict the default payments.

Physical. The experiments incorporate three physical datasets, namely **HIGGS**, **Occupancy Detection**, and **Magic**. **Higgs** is a large-scale dataset of 11M records and 28 features. It is used in a classification problem to distinguish between a signal process of Higgs bosons and a background process [42]. The **Occupancy Detection** [43] dataset is used for binary classification to determine if a room is occupied or not based on seven measurements, such as room temperature, humidity, and light. **Magic** [44] is a dataset of 19K records and 12 features that simulates the registration of high-energy gamma particles in an atmospheric telescope. The classification target is to discriminate between photons that are caused by primary gammas and the images of hadronic showers.

Social. We also use the **Rain Prediction** dataset, which contains daily weather observations from numerous Australian weather stations. It has more than 140K records, and 24 features to classify whether or not it will rain tomorrow. Another dataset is the News [45] dataset, which summarizes a set of 61 heterogeneous features of online articles. The dataset has more than 39K records and is used to classify the popularity of a given article.

Image. Finally, the experiments include an example of multiclass classifications tasks using the MNIST dataset [46]. The dataset contains 70K handwritten digits images with 10 classes, from “0” to “9.”

4.1.2 Baseline Methods. We compare the proposed method to the following methods:

- **Data programming** [2]: The experiments include the DP system in [6]. The system requires users to write labeling functions to express arbitrary heuristics. Then, the system denoises their outputs without access to ground truth by incorporating the DP paradigm [12].
- **Automated weak supervision:** We incorporate an automated weak supervision approach (WS-Automatic) described in [13]. The method takes advantage of a small labeled dataset to automatically create weak heuristics and generate labels to an unlabeled dataset.
- **Hybrid approaches:** We compare the proposed method to a hybrid approach (DALP) from [38]. Similar to DP, the hybrid method allows users to write labeling functions. Then, it applies active learning to enhance the accuracy of the generated labels.
- **Active learning** [7]: We compare Asterisk to different active learning strategies [17], namely uncertainty sampling (UNC), query by committee (QBC), density-weighted uncertainty sampling (DWUNC), and random sampling (RAND).
- Also, to evaluate the individual components of the proposed system, we added two variations of the proposed system, namely **Asterisk-manual** and **Asterisk-AL**. In Asterisk-manual, the component of the automatic generation of heuristics is disabled (Section 3.2.1), and the system relies on the users to write labeling functions for each classification task. On the other hand, Asterisk-AL does not incorporate the data-driven active learner component (Section 3.2.2). Instead, it applies uncertainty sampling to rank the points in D_{LC} .

4.1.3 Implementation Details.

Generating Heuristics. To write the labeling functions used for DP [6], DALP [38], and Asterisk-manual, we implemented a set of threshold-based labeling functions [6, 13] in which the labeling functions rely on numerical thresholds to output a label for each data point or abstain. To create these labeling functions, a set of gold labels is required to evaluate the quality of these functions. To ensure fairness of the experiments, the same labeling budget B specified by the proposed method is used as gold labels. Also, as for WS-Automatic, the same set of gold labels is used to develop and evaluate the generated heuristics.

Table 2. Settings for the User-Defined Labeling Functions

Dataset	# Candidates	# Labeling Functions	Labeling Functions Performance	
			Acc	F1
Higgs	8,800,000	9	0.63	0.50
Renewal Sales	1,239,849	7	0.85	0.80
Rain Prediction	113,755	11	0.87	0.75
Travel Insurance	50,662	6	0.81	0.79
Bank	36,169	8	0.78	0.67
News	31,716	10	0.69	0.74
Credit Card	24,000	7	0.73	0.75
Occupancy Detection	16,448	9	0.65	0.80
Magic	15,216	8	0.82	0.73
MNIST	56,000	12	0.68	0.74

For the renewal sales dataset, we consulted a set of sales representatives from IBM to help us write seven labeling functions. As for the rest of the datasets, we applied pattern matching to decide on the threshold values. We consider using pattern matching techniques since we do not have access to domain experts. Also, we find the technique consistent with the best practice followed in the literature [6, 13, 20, 47].

Moreover, to only append accurate sources, we calculated the empirical accuracy of the labeling functions using D_L and only included the functions that reported more than 60% accuracy. Table 2 shows the settings of the labeling functions used in the experiments including the number of candidates (data points) for which labels are created, the number of labeling functions designed for each task, and the empirical evaluation metrics (accuracy and F1 measure) calculated using D_L .

Active Learning Settings. When comparing the proposed method against different AL sampling techniques, we averaged the results over 10 runs. For each classification task, the labeled dataset D_L was used as the initial seed. The unlabeled dataset D_U was split into an unlabeled pool and a separate held-out test set to evaluate the classifier after each iteration. Moreover, to decide when to terminate the active learning process, we examined the classifier performance and stopped when the classification accuracy did not show significant improvements with additional iterations [48, 49]. More specifically, we used a threshold λ for the differences between accuracy values achieved by the classifier and stopped the process when the mean of these differences did not exceed λ for a successive number of iterations. While having a small value for λ can increase the annotation cost, choosing a larger value can result in missing useful generalizations [49]. Therefore, in earlier research [50], we experimented with a range of different values for λ to decide on the optimal value to stop the active learning process. We found that having $\lambda = 0.0001$ succeeds in catching the elbow values in the learning curves, after which the performance does not notably change. Moreover, to keep all the conditions the same throughout the experiments, we report the number of queried points consumed by the active learning process and used it as the labeling budget B for the proposed method.

Performance Metrics. Since training models with accurate labels improve their capability to generalize to unseen points in the test data [2, 6], the experiments report the classification performance of the final models trained with the generated labels along with the labeling accuracy of the generated labels. Also, reporting only one evaluation metric has been proven to be not enough to judge machine learning models [51]. Therefore, we report a handful of evaluation metrics. As for the end models, we report classification accuracy (Acc), F1 measure (F1), and Matthews correlation

coefficient (MCC). Classification accuracy represents the number of correct predictions made divided by the total number of predictions reported and is calculated as $(TP+TN)/(TP+TN+FP+FN)$, where TP, TN, FP, and FN are the numbers of true positives, true negatives, false positives, and false negatives, respectively. However, accuracy can be misleading for classifying imbalanced datasets. Since some of the datasets used in the experiments are imbalanced, we consider MCC [52] to describe the confusion matrix and the classifier performance. MCC is calculated as

$$MCC = \frac{TP*TN - FP*FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}, \quad (12)$$

for binary classifications and has been generalized to the multiclass situations [52]. Moreover, we report F1 measure as a harmonic mean of both precision and recall.

Also, since all the datasets used in the experiments contain ground-truth labels, the experiments record the labeling accuracy achieved by the proposed method, WS-Automatic, data programming, and the hybrid approach DALP. To calculate the labeling accuracy, the ground-truth labels are initially removed to create the unlabeled dataset D_U . Then, each of these approaches is used to generate labels for D_U . Later, the generated labels are compared to the ground truth to calculate the labeling accuracy. The labeling accuracy is measured as the ratio between the number of correctly labeled instances and the training set size.

Final Learning Models. Since the proposed method aims at generating labeled training datasets for any downstream model, we experimented with a wide range of classification algorithms to find the best model for each task. In the case of the renewal sales and the Higgs datasets, we used the gradient boosting algorithm XGBoost [53]. As for the bank, the rain prediction, and the travel insurance datasets, we applied random forest classifier since it has been evaluated as a reliable classifier in various classification tasks [54]. In the news and the MNIST datasets, classifiers were implemented using the linear Support Vector Machine (SVM) classifier [55]. As for the credit card, magic, and occupancy detection tasks, we chose logistic regression. The principal motive for using this range of different classification algorithms is to demonstrate the resiliency of the proposed method to the classifier and to show that, regardless of the choice of the classifier, the final model can generalize beyond the generated labels and produce predictions to the unseen points in the test set.

Handling Imbalanced Data. Moreover, some of the datasets used in the experiments are highly imbalanced, such as the renewal sales, the rain prediction, the travel insurance, and the bank datasets. To deal with such data, the experiments are designed to maintain an equal number of labeled samples for each class in the labeled dataset D_L . As a result, a balanced new dataset is retrieved and used to create the initial set of heuristics H . Then, the final learning models are tuned to learn with imbalanced data. For example, we adjust the *scale_pos_weight* parameter in the gradient boosting algorithm to control the balance of positive and negative weights before applying XGBoost to the renewal sales and the Higgs datasets. As for the random forest classifier with the bank, the travel insurance, and the rain prediction datasets, we assign higher weights to the minority class and penalize the misclassification for instances of this class. Similarly, a class-weighted SVM algorithm is applied to deal with imbalanced data in the news dataset. Overall, the experimental results depict that the framework can handle imbalanced data and achieve up to 90.36% in classification performance in terms of F1 measure and 83.02% on average in terms of MCC (Table 3).

4.2 Experimental Results of End-to-End Systems

In this section, we compare the proposed method against WS-Automatic, DP, DALP, and various active learning strategies. The results obtained from the experiments demonstrate that classifiers

Table 3. Asterisk vs. Automatic Weak Supervision Approach, WS-Automatic

Dataset	Asterisk System							Asterisk Improvement over WS-Automatic (%)						
	Heuristics Performance			End-Model Performance			Labeling Accuracy (%)	Heuristics Performance			End-Model Performance			Labeling Accuracy
	Acc	F1	Cov.	Acc	MCC	F1		Acc	F1	Cov.	Acc	MCC	F1	
Higgs	0.81	0.85	0.80	0.85	0.72	0.84	72.14	0.67	1.25	0.99	6.40	13.51	2.28	7.67
Renewal Sales	0.71	0.82	0.78	0.86	0.83	0.90	83.53	3.04	5.40	16.02	-2.55	20.88	50.28	43.08
Rain Prediction	0.73	0.85	0.81	0.85	0.73	0.80	85.25	7.13	5.24	-5.81	3.91	-9.54	0.48	7.91
Travel Insurance	0.78	0.72	0.74	0.78	0.81	0.71	78.14	25.99	32.67	6.15	-17.55	18.94	0.25	20.22
Bank	0.97	0.92	0.87	0.87	0.86	0.82	93.67	13.74	27.41	4.99	6.21	6.19	12.38	6.13
News	0.92	0.91	0.88	0.96	0.92	0.90	96.57	13.67	21.55	14.71	20.91	18.31	73.07	15.16
Credit Card	0.96	0.89	0.91	0.92	0.93	0.95	87.75	10.53	13.98	10.96	18.52	21.08	26.05	12.72
Occupancy Detection	0.90	0.92	0.88	0.95	0.93	0.90	98.91	15.88	24.03	15.37	20.73	16.56	20.20	20.51
Magic	0.82	0.74	0.81	0.90	0.91	0.93	89.73	7.82	34.74	17.43	9.13	25.08	15.38	6.69
MNIST	0.96	0.93	0.86	0.95	0.94	0.92	93.16	8.85	12.05	14.37	5.77	5.75	8.08	8.33

The performance of the generated heuristics (accuracy (Acc), coverage (Cov.), and F1 measure (F1)), the end-model performance (accuracy (Acc), MCC, and F1), and the accuracy of the generated labels (Labeling Accuracy) compared to the ground truth in each dataset.

trained with labels from the proposed system can achieve better performance (F1 measure) than WS-Automatic by 21% on average (Section 4.2.1) and outperform user-defined sources (DP and DALP) by 14.56% on average in F1 measure (Section 4.2.2). The results also show that, when compared to active learning, the proposed method can improve the performance of end models by 6.20% on average in F1 measure (Section 4.2.3).

4.2.1 Asterisk vs. Automatic Generation of Weak Supervision. This section compares Asterisk to the automatic weak supervision approach, WS-Automatic. Since both methods rely on using a small labeled dataset to generate labels for a bigger unlabeled dataset, we provide each method with the same set of labeled points of size M depicted in Table 1. The rest of the data points in each dataset were then used as the unlabeled dataset D_U . Then, each method was used to generate training labels for D_U and train the same classifier with these labels to create a final model. Table 3 shows the results of the proposed method, along with its improvements over WS-Automatic. As shown in the table, the experiments consider the performance of the final heuristics in terms of accuracy (Acc), F1 measure (F1), and the achieved coverage of the generated labels (Cov.). The table also presents the performance of the end models trained using the generated labels in terms of accuracy (Acc), MCC, and F1 measure, along with the accuracy of the generated labels (Labeling Accuracy).

The proposed model does not only rely on the automatic generation of weak sources; it also uses the data-driven learner to enhance the accuracy of the generated heuristics. Therefore, the generative model learns more accurate heuristics. This can be inferred from the results as the proposed system creates more accurate heuristics than WS-Automatic in all the tasks, with the maximum improvement in the travel insurance dataset with a 26% increase in accuracy. Also, the results show that the learner component helps in resolving situations of labeling abstaining, which results in enhancing the coverage of the training dataset by up to 17.43% in the magic dataset. In some datasets such as the credit card, the proposed method manages to create labels for more than 90% of the points in D_U . Also, in large-scale datasets such as the Higgs and the renewal sales, the proposed method creates heuristics that labeled 80% and 78% of D_U , respectively. In general, the results

Table 4. Improvements of Asterisk over User-Defined Heuristics (DP and DALP)

Dataset	Asterisk Improvement over DP (%)						Asterisk Improvement over DALP (%)					
	Generative Model Performance		Final Model Performance			Labeling Accuracy	Generative Model Performance		Final Model Performance			Labeling Accuracy
	Acc	F1	Acc	MCC	F1		Acc	F1	Acc	MCC	F1	
Higgs	31.52	70.40	60.69	40.22	47.13	36.02	10.32	7.64	16.60	5.16	18.12	18.26
Renewal Sales	-1.25	2.95	7.62	7.21	11.89	22.84	-14.34	-8.95	-7.42	-8.34	6.96	-0.67
Rain Prediction	-12.04	13.66	9.17	3.20	24.71	9.49	-9.87	0.29	4.32	-7.25	5.49	4.53
Travel Insurance	2.78	-11.37	6.82	20.72	22.53	11.79	-9.17	-13.06	-10.91	6.42	2.85	8.53
Bank	43.68	36.38	19.30	16.24	6.55	49.63	19.85	9.43	0.10	1.43	1.28	22.13
News	4.05	24.30	6.81	4.21	7.57	88.74	15.01	8.39	2.27	1.57	0.40	62.35
Credit Card	37.98	18.85	42.20	35.31	29.42	68.37	17.24	10.43	14.11	9.68	9.48	36.82
Occupancy Detection	20.51	15.52	17.32	12.62	7.20	57.86	6.33	6.19	10.50	1.03	2.45	21.57
Magic	18.96	2.07	24.37	30.44	36.67	16.58	6.60	-6.21	1.60	11.35	31.29	9.42
MNIST	17.05	25.51	16.09	13.67	9.76	81.83	3.20	3.33	9.41	4.58	9.37	57.90

The performance measures reported by the generative models (accuracy (Acc) and F1 measure (F1)), the performance measures reported by the end models (accuracy (Acc), MCC, and F1 measure (F1)), and the accuracy of the generated labels (Labeling Accuracy) compared to the ground truth.

demonstrate how, for many tasks, using the component of data-driven learner helps in enhancing the quality and the coverage of the generated heuristics.

As for the classification performance, the end models in Asterisk perform better in most of the tasks. Although for some tasks, such as the renewal sales and the travel insurance datasets, WS-Automatic achieves higher accuracy than Asterisk, the accuracy metric could be misleading here due to the class imbalance in these datasets. Alternatively, the results show that Asterisk enhances the MCC values by 21% and 19% in the renewal sales and travel insurance tasks, respectively. Also, the proposed method maintains better F1 measure throughout all datasets. Since training the models using accurate data improves their capabilities to generalize to unseen observations, this proves that the proposed method could enhance the quality of the training labels.

Finally, as to the labeling accuracy, the results show that Asterisk generates more accurate labels than WS-Automatic. In some datasets such as the news and the occupancy detection datasets, the proposed method achieves labeling accuracy more than 90%. Asterisk also improves the labeling accuracy when compared to WS-Automatic with the highest accuracy reported in the renewal sales dataset with a boost of 43.08%. Overall, the results demonstrate that, since the proposed method uses the learner component to provide true labels for a portion of the dataset, it manages to output more accurate labels and hence improves the performance of the final models.

4.2.2 Asterisk vs. User-Defined Heuristics. We compare the proposed method to two labeling methods that rely on user-defined heuristics: DP [6] and DALP [38]. During the experiments, the two methods, along with Asterisk, were used to generate labels and perform the classification tasks using the 10 datasets. Table 4 shows the improvement of the proposed method over DP and DALP. First, since all of the three methods use generative models to produce probabilistic labels, the table shows the improvement of the proposed approach with regard to the performance metrics learned by the generative model (accuracy (Acc) and F1 measure (F1)) over the two methods. The table also shows the improvements made by Asterisk for the performance of the end model (accuracy (Acc), MCC, and F1 measure (F1)) along with the labeling accuracy.

The results show that, in most of the tasks, the proposed approach outperforms the other two methods in the generative model performance. For example, in the Higgs, the credit card, and the

occupancy detection datasets, the proposed approach surpasses the other methods by significant margins. As for the accuracy learned by the generative model in the Higgs dataset, Asterisk improves the performance by 31.52% when compared to DP and 10.32% when compared to DALP. It also achieves the highest F1 measure among the three methods in these datasets. Generally, the results show that the proposed method sustains better results for the generative model in tasks where designing weak supervision sources is challenging. For example, for datasets like the news and MNIST dataset, writing the labeling functions is hard due to the large number of features that must be considered to write accurate heuristics. Also, for some other datasets such as the magic and the Higgs datasets, a high level of domain experience is needed to design the labeling functions. In these situations, automatic creation of the heuristics can be beneficial to obtain a high-quality set of heuristics.

On the other hand, in datasets where domain experience is available (e.g., renewal sales) or the learning task is easy enough to facilitate designing the weak supervision sources (e.g., rain prediction), the proposed method is outperformed by the other approaches. For example, in the renewal sales dataset, DALP achieves the highest accuracy and F1 measure among the three approaches. Also, in the rain prediction dataset, DP surpasses both Asterisk and DALP in terms of accuracy by 13.69% and 2.47%, respectively. Nevertheless, although the proposed method worked with less accurate supervision sources, it enhanced the overall labeling accuracy in both renewal sales and the rain prediction tasks when compared to DP by 22.84% and 9.49%, respectively. It also outperformed DALP in the rain prediction task by 4.53% in labeling accuracy.

Moreover, when considering the end-model performance, Asterisk outperforms DP in all the problems, with the biggest improvement in the credit card dataset with 40% in MCC. Also, when compared to DALP, the proposed method improves the performance of the final model in most of the tasks. Although DALP outperforms the proposed method in the renewal sales and the rain prediction datasets with 9.10% and 7.82% in MCC values, respectively, the proposed method maintains its superiority in the rest of the tasks, with the highest enhancement in the magic dataset with an 11.35% increase in MCC values. Moreover, the results show that except for the renewal sales dataset, the proposed method achieves the highest labeling accuracy in all the problems. It attains an average improvement in the labeling accuracy of 44.31% when compared to DP and 24.08% when compared to DALP. All in all, the results show that the proposed method can be a suitable solution to achieve a high level of labeling accuracy and classification performance, especially when designing supervision sources becomes expensive.

4.2.3 Asterisk vs. Active Learning Query Strategies. In this section, we report the results obtained when comparing the proposed method with four sampling techniques. As mentioned before, the labeling budget B in the proposed method is determined in each task based on the labeling cost of the active learning process. Therefore, to set up the experiments, we first applied the four query strategies (UNC, DWUNC, QBC, and RAND) to the 10 datasets. Figure 4 shows the classification accuracy and the MCC values of the end models achieved by the four sampling strategies. As the figure shows, uncertainty sampling maintains the highest levels of accuracy through all the tasks. Therefore, we report the results of UNC against the proposed method in Table 5. The table shows the performance of the end models and the number of labeled points consumed by UNC to achieve the reported accuracy level (AL cost). Since we use the value of AL cost as the labeling budget B , the table also reports the value of L_C , which represents the size of the unlabeled pool D_{LC} in the data-driven learner component. As mentioned before, D_{LC} represents the points with low-confidence labels, and its size affects the size of D_{AL} . Therefore, the value of L_C is reported to demonstrate the cost of manual labeling in the proposed method.

Fig. 4. Performance of end models in active learning experiments for (a) classification accuracy and (b) MCC.

The results show that Asterisk achieves a higher level of accuracy when compared to active learning in almost all of the datasets. Although UNC outperforms the proposed method in the bank and the magic datasets by 4.81% and 2.01% in accuracy, respectively, for large-scale datasets such as the Higgs, the renewal sales, and the rain prediction, Asterisk manages to surpass UNC by 21.24%, 0.93%, and 4.68%, respectively. Moreover, Asterisk also manages to enhance MCC values in all the datasets when compared to UNC, with an average of 17.28%. As for the F1 measure, Asterisk also achieves higher values than UNC in all the problems, with 6.20% enhancement on average.

Table 5. Asterisk vs. Active Learning

Dataset	The Asterisk System				Active Learning (UNC)					
	End-Model Performance				End-Model Performance			Lift % (F1)	AL Cost	Lower % (Labeling Cost)
	Acc	MCC	F1	L _C	Acc	MCC	F1			
Higgs	0.88	0.71	0.86	941,857	0.72	0.65	0.75	14.01	1,198,850	21.44
Renewal Sales	0.96	0.93	0.89	81,710	0.95	0.84	0.82	8.60	125,988	35.14
Rain Prediction	0.97	0.83	0.93	10,645	0.92	0.79	0.88	5.35	15,004	29.05
Travel Insurance	0.98	0.83	0.91	4,639	0.94	0.80	0.90	0.97	6,704	30.80
Bank	0.89	0.87	0.85	2,867	0.93	0.66	0.84	0.99	3,364	14.77
News	0.98	0.97	0.96	7,193	0.93	0.81	0.91	5.16	13,818	47.94
Credit Card	0.92	0.93	0.95	6,120	0.91	0.67	0.91	4.10	12,958	52.77
Occupancy Detection	0.99	0.98	0.93	7,521	0.93	0.74	0.86	8.00	11,855	36.56
Magic	0.94	0.91	0.93	1,739	0.96	0.82	0.83	12.01	2,141	18.78
MNIST	0.95	0.94	0.92	2,173	0.92	0.84	0.89	2.77	3,472	37.41

The performance measures reported by the end models (accuracy (Acc), MCC, and F1 measure (F1)), the labeling cost of the Asterisk system (L_C), and the number of labeled instances consumed by active learning (AL Cost).

Also, as shown in the table, active learning consumed more labels than the proposed method in all the tasks. The values of L_C shown in the table demonstrate that the size of D_{LC} remained less than the value of B through all the experiments. Hence, the data-driven learner in the proposed method stopped when there were no more points with low-confidence labels to resolve in D_{LC} instead of exceeding the labeling budget B. As a result, Asterisk reduces the labeling cost in all the problems, with the highest reduction of 53% in the credit card task. In short, the superiority of the proposed method over the AL process can be traced to two main reasons. First, the data-driven learner component in the proposed method starts with a larger seed since it employs both the data points with high confidence D_{HC} and D_L to form the seed. As a result, this enhances the initial accuracy of the end model and reduces the labeling cost. Second, the size of the unlabeled pool D_{LC} is much smaller than the size of the unlabeled pool used in the baseline active learning strategies, as D_{LC} only represents the points with low-confidence labels rather than the entire instance space. As a result, this helps reduce the labeling cost since it makes the data-driven learner converge faster.

4.3 Experimental Results of Micro-Benchmarking

In this part, we assess the effect of the individual components of the proposed method, more specifically, the heuristics generator component and the data-driven learner component. The section shows that disabling either component can deteriorate the classification accuracy by up to 10.07% and decline the labeling accuracy by up to 61.26%. To create the Asterisk-Manual modified version, we disable the automatic creation of the weak heuristics. Instead, we use the same labeling functions used for the approaches of user-defined heuristics (Table 2). Alternatively, in Asterisk-AL, the data-driven learner component is replaced by uncertainty sampling to choose the most informative points in D_{LC} for which true labels are provided. The results obtained using both versions to generate labels for the 10 tasks are illustrated in Table 6. The table shows, for each modified version, the evaluation metrics for the end model along with the labeling accuracy.

The results show that, with regard to Asterisk-Manual, disabling the heuristics generator component negatively affects the performance of the end models in most of the tasks, with the highest reduction in the credit card dataset, with a 10.07% decrease in accuracy. However, in the datasets where user-defined labeling functions are more accurate than the generated heuristics, Asterisk-Manual improves the performance of end models. For example, in the renewal sales and the rain

Table 6. Performance of Asterisk-Manual and Asterisk-AL

Dataset	Asterisk-Manual				Asterisk-AL			
	End-Model Performance			Labeling Accuracy (%)	End-Model Performance			Labeling Accuracy (%)
	Acc	MCC	F1		Acc	MCC	F1	
Higgs	0.78	0.69	0.76	72.08	0.85	0.69	0.80	61.34
Renewal Sales	0.96	0.93	0.89	83.19	0.79	0.77	0.87	83.10
Rain	0.83	0.80	0.77	84.15	0.83	0.69	0.77	81.55
Prediction								
Travel	0.89	0.79	0.71	75.26	0.71	0.76	0.69	72.14
Insurance								
Bank	0.91	0.85	0.83	81.35	0.88	0.87	0.84	76.71
News	0.95	0.93	0.94	78.15	0.95	0.93	0.92	56.60
Credit Card	0.83	0.90	0.91	41.23	0.87	0.89	0.91	34.31
Occupancy	0.88	0.94	0.90	72.26	0.91	0.94	0.90	68.30
Detection								
Magic	0.91	0.86	0.73	86.39	0.91	0.86	0.84	82.13
MNIST	0.91	0.91	0.86	67.45	0.89	0.92	0.88	59.04

Performance reported by the end models (accuracy (Acc), MCC, and F1 measure (F1)) and the accuracy of the generated labels (Labeling Accuracy) compared to the ground truth.

prediction datasets, Asterisk-Manual enhances the classification accuracy by 11.76% and 9.36%, respectively, in MCC values. However, Asterisk-Manual achieves less labeling accuracy than the proposed method in all the problems with a maximum reduction of 53.02% in the credit card dataset. Overall, the results empirically posit that the heuristics generator component enhances the overall classification accuracy and the accuracy of the generated labels.

Alternatively, disabling the data learner component decreases the classification performance in all datasets. As the results show, Marque-AL achieved less accuracy in all datasets, with an average of a 5.10% decrease in accuracy, a 3.94% decrease in MCC, and a 4.54% decrease in F1 measure. Moreover, since the active learning part does not incorporate the outcome of the generative model in deciding on the points for which true labels should be provided, this limits the capability of the proposed method to enhance the accuracy of the generated labels. This can be derived from the results, as Asterisk-AL achieved less labeling accuracy than Asterisk in all the problems, with the highest decrease in the credit card dataset. Finally, the results empirically show the importance of the data learner component in enhancing the labeling accuracy and achieving better classification performance.

5 RELATED WORK

The scarcity of labeled training data has been an abiding problem for machine learning developers and data scientists, which has motivated researchers to explore different labeling techniques. Therefore, in this section, we provide an overview of the methods that aim at automating the process of generating training labels.

Generating Noisy Labels. Previous research [56–58] utilized weak supervision sources to provide high-level supervision in the form of noisy labels for massive datasets. For example, the framework proposed in [56] frames the process of aggregating different weak supervision sources as a matrix completion problem for multitask learning. Another work [57] focuses on the multiple instance learning paradigm and proposes a system that casts weak labels as an optimization scheme

to identify the most discriminative instances. Also, researchers in [58] introduce an approach to supervise machine learning models with weak supervision sources by specifying constraints that hold over the output space. Nevertheless, the work in [56] is specific to multitask weak supervision settings where diverse labeling sources have different granularities related to subtasks of a problem. Alternatively, our settings are different since we have a set of weak supervision sources solving the same task, and hence, they abstain, overlap, and conflict. Also, unlike our proposed method, other research [57, 58] focuses on applying weak supervision with specific models. For example, the work in [57] aims at improving the predictive accuracy of Latent SVM for image and text classification tasks. Likewise, the authors in [58] try to enhance the capability of neural network models to handle weakly labeled datasets.

Furthermore, other research [8, 9, 59] has applied weak supervision to generate massive labeled datasets. However, unlike Asterisk, most of these approaches are only applicable to specific domains. For example, Gurjar et al. [8] introduce an approach to retrain the high performance of convolutional neural networks with weak supervision for tasks of handwriting recognition. Similarly, Chaidaroon et al. [9] apply an unsupervised method to extract weak signals from training data and leverage these signals for text hashing. Cao et al. [59] provide an end-to-end solution to the pattern classification problem in medical imaging.

Combining Noisy Signals. Moreover, there is ample research [2, 6, 60] that focuses on using generative models to aggregate weak supervision sources without the use of labeled data. The success of these approaches heavily relies on the quality of the labeling functions the users encode [22], which can be problematic, especially when designing such sources requires a high level of domain experience [13, 22]. Moreover, although the concept of automating the weak supervision sources has been studied in the literature [13, 22], none of these approaches [2, 6, 13, 22, 60] employ any domain experience when denoising the weak supervision sources, which makes it challenging to estimate the coverage and the accuracy of the generated labels.

Optimizing Annotation Cost and Labeling Quality. Similar to weak supervision, other techniques [32, 37, 61, 62] have been proposed to provide solutions to the increasing demand for large-scale, high-quality labeled data. For example, research in [37] aims to formalize the user strategies for selecting data points in the active learning process. The study concludes that user-centered strategies can be beneficial in the early phases of the labeling process to resolve the cold-start problem in active learning. Although their findings [37] are consistent with our proposed system, Asterisk tries to mitigate the bootstrap problem by leveraging weak supervision sources in the beginning. The authors in [61] propose a new active learning method that learns associations from deep neural networks to enhance the batch mode in AL. Also, as for meta-learning, other research [62] introduces a model that uses meta-learning to learn active learning strategies. However, in contrast to Asterisk, most active learning algorithms [32, 61, 62] have been validated on small and medium-size datasets. For example, the technique in [62] was validated on two datasets with a maximum size of 5,000 training samples and 1,000 test samples. Also, the proposed technique in [61] was tested using two datasets varying in size, with a maximum of 70,000 records.

Combining Weak Supervision and Active Learning. Other research [63,–65] provides various labeling solutions that combine active learning with weak supervision [63, 64] and transfer learning [65]. While research [63] explores using active learning and weak supervision as an integrated solution to debug machine learning models, the authors in [64] combine active learning with weakly labeled data to reduce the annotation cost. Nevertheless, none of these techniques have tried to employ domain experience to denoise the weak sources. On the other hand, Zhou et al. [65] employs a pretrained convolutional neural network and gradually fine-tunes it using active learning. Although the method aims at reducing the annotation cost, unlike the proposed

method, it is specific to the application of convolutional neural networks and is only evaluated within the domain of biomedical imaging.

6 CONCLUSIONS

The article presents Asterisk, a framework for generating high-quality labeled datasets at scale. The technique employs an iterative process to automatically generate high-accuracy heuristics to assign initial labels. Then, it applies a data-driven active learning process to further enhance the quality of the generated heuristics. The process learns the active learning strategy while considering the modeled accuracies of the produced heuristics and the noise in the generated labels. The framework applies the learned strategy to economically engage the user and enhance the quality of the generated labels. We evaluate the proposed framework by comparing its performance with other weak supervision techniques such as data programming and automated weak supervision, along with active learning strategies. The empirical results show that the proposed framework can significantly enhance the learned accuracy of the generated heuristics by up 44%, while producing high coverage labels for up to 91% of the unlabeled dataset. Also, compare to the weak supervision techniques, the results show that the framework improves the quality of the generated labels by 28% on average. As well, the framework can reduce the annotation effort by up to 53% when compared to the baseline active learning strategies.

REFERENCES

- [1] W. Zhao, G. Guan, L. Chen, X. He, D. Cai, B. Wang, and Q. Wang. 2018. Weakly-supervised deep embedding for product review sentiment analysis. *IEEE Trans. Knowl. Data Eng.* 30, 1 (2018), 185–197.
- [2] A. J. Ratner, C. M. De Sa, S. Wu, D. Selsam, and C. Ré. 2016. Data programming: Creating large training sets, quickly. *Advances in Neural Information Processing Systems*, pp. 3567–3575.
- [3] V. S. Sheng, J. Zhang, B. Gu, and X. Wu. 2019. Majority voting and pairing with multiple noisy labeling. *IEEE Trans. Knowl. Data Eng.* 31, 7 (2019), 1355–1368.
- [4] P. Cheng, X. Lian, X. Jian, and L. Chen. 2019. FROG: A fast and reliable crowdsourcing framework. *IEEE Trans. Knowl. Data Eng.* 31, 5 (2019), 894–908.
- [5] C. De Sa, A. Ratner, C. Ré, J. Shin, F. Wang, S. Wu, and C. Zhang. 2016. DeepDive: Declarative knowledge base construction. *SIGMOD Rec.* 45, 1 (2016), 60–67.
- [6] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré. 2017. Snorkel: rapid training data creation with weak supervision. *Proc. VLDB Endow.* 11, 3 (2017), 269–282.
- [7] Y. Fu, X. Zhu, and B. Li. 2013. A survey on instance selection for active learning. *Knowl. Inf. Syst.* 35, 2 (2013), 249–283.
- [8] N. Gurjar, S. Sudholt, and G. A. Fink. 2018. Learning deep representations for word spotting under weak supervision. *International Workshop on Document Analysis Systems*, pp. 7–12.
- [9] S. Chaidaroon, T. Ebesu, and Y. Fang. 2018. Deep semantic text hashing with weak supervision. *ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1109–1112.
- [10] A. H. Akbarnejad and M. S. Baghshah. 2019. An efficient semi-supervised multi-label classifier capable of handling missing labels. *IEEE Trans. Knowl. Data Eng.* 31 (2019), 229–242.
- [11] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. 2016. Improved techniques for training GANs. *Advances in Neural Information Processing Systems*, pp. 2234–2242.
- [12] S. H. Bach, B. He, A. Ratner, and C. Ré. 2017. Learning the structure of generative models without labeled data. *Proc. 34th International Conference on Machine Learning*, pp. 273–282.
- [13] P. Varma and C. Ré. 2018. Snuba: Automating weak supervision to label training data. *Proc. VLDB Endowment* pp. 223–236.
- [14] E.-C. Huang, H.-K. Pao, and Y.-J. Lee. 2017. Big active learning. *IEEE International Conference on Big Data*, pp. 94–101.
- [15] Z.-H. Zhou. 2017. A brief introduction to weakly supervised learning. *Natl. Sci. Rev.* 5, 1 (2017), 44–53.
- [16] M.-F. Balcan, S. Hanneke, and J. W. Vaughan. 2010. The true sample complexity of active learning. *Machine Learn.* 80, 2–3 (2010), 111–139.
- [17] B. Settles. 2009. Active Learning Literature Survey.
- [18] N. Dalvi, A. Dasgupta, R. Kumar, and V. Rastogi. 2013. Aggregating crowdsourced binary ratings. *Proc. International Conference on World Wide Web*, pp. 285–294.

- [19] M. Joglekar, H. Garcia-Molina, and A. Parameswaran. 2015. Comprehensive and reliable crowd assessment algorithms. *IEEE International Conference on Data Engineering*, pp. 195–206.
- [20] P. Varma, B. He, D. Iter, P. Xu, R. Yu, C. De Sa, and C. Ré. 2016. Socratic learning: Augmenting generative models to incorporate latent subsets in training data. *ArXiv161008123 Cs Stat*, 2016.
- [21] P. Varma et al. 2016. Socratic learning: Augmenting generative models to incorporate latent subsets in training data. *ArXiv161008123 Cs Stat*, 2016.
- [22] N. Das, S. Chaba, S. Gandhi, D. H. Chau, and X. Chu. 2019. GOGGLES: Automatic training data generation with affinity coding. *ArXiv190304552 Cs*, 2019.
- [23] J. Zhu, H. Wang, B. K. Tsou, and M. Ma. 2010. Active learning with sampling by uncertainty and density for data annotations. *IEEE Trans. Audio Speech Lang. Process.* 18, 6 (2010), 1323–1331.
- [24] R. B. C. Prudencio and T. B. Ludermit. 2008. Active meta-learning with uncertainty sampling and outlier detection. *IEEE International Joint Conference on Neural Networks*, pp. 346–351.
- [25] K. Konyushkova, R. Sznitman, and P. Fua. 2015. Introducing geometry in active learning for image segmentation. *ArXiv150804955 Cs*, 2015.
- [26] Y. Yang, Z. Ma, F. Nie, X. Chang, and A. G. Hauptmann. 2015. Multi-class active learning by uncertainty sampling with diversity maximization. *Int. J. Comput. Vis.* 113, 2 (2015), 113–127.
- [27] M. Liu, W. Buntine, and G. Haffari. 2018. Learning how to actively learn: A deep imitation learning approach. *Proc. Annual Meeting of the Association for Computational Linguistics*. 1 (2018), 1874–1883.
- [28] M. E. Ramirez-Loaiza, M. Sharma, G. Kumar, and M. Bilgic. 2017. Active learning: An empirical study of common baselines. *Data Min. Knowl. Discov.* 31, 2 (2017), 287–313.
- [29] M. Fang, Y. Li, and T. Cohn. 2017. Learning how to active Learn: A deep reinforcement learning approach. *ArXiv170802383 Cs*, Aug. 2017.
- [30] K. Konyushkova, R. Sznitman, and P. Fua. 2017. Learning active learning from data. *Advances in Neural Information Processing Systems*.
- [31] H. Chu and H. Lin. 2016. Can active learning experience be transferred? *IEEE International Conference on Data Mining*, pp. 841–846.
- [32] K. Pang, M. Dong, Y. Wu, and T. Hospedales. 2018. Meta-learning transferable active learning policies by deep reinforcement learning. *ArXiv Prepr. ArXiv180604798*, 2018.
- [33] A. Niculescu-Mizil and R. Caruana. 2005. Predicting good probabilities with supervised learning. *Proc. International Conference on Machine Learning*, pp. 625–632.
- [34] O. Sagi and L. Rokach. 2018. Ensemble learning: A survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* 8, 4 (2018), 1–18.
- [35] B. Desharnais, F. Camirand-Lemyre, P. Mireault, and C. D. Skinner. 2015. Determination of confidence intervals in non-normal data: Application of the bootstrap to cocaine concentration in femoral blood. *J. Anal. Toxicol.* 39, 2 (2015), 113–117.
- [36] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. 2014. Supervised hashing for image retrieval via image representation learning. *AAAI Conference on Artificial Intelligence*, pp. 2156–2162.
- [37] J. Bernard, M. Zeppelzauer, M. Lehmann, M. Müller, and M. Sedlmair. 2018. Towards user-centered active learning algorithms. *Comput. Graph. Forum.* 37, 3 (2018), 121–132.
- [38] M. Nashaat, A. Ghosh, J. Miller, S. Quader, C. Marston, and J. F. Puget. 2018. Hybridization of active learning and data programming for labeling large industrial datasets. *IEEE International Conference on Big Data*, pp. 46–55.
- [39] M. Nashaat, A. Ghosh, J. Miller, S. Quader, and C. Marston. 2019. M-Lean: An end-to-end development framework for predictive models in B2B scenarios. *Inf. Softw. Technol.* 113 (2019), 131–145.
- [40] S. Moro, P. Cortez, and P. Rita. 2014. A data-driven approach to predict the success of bank telemarketing. *Decis. Support Syst.* 62 (2014), 22–23.
- [41] I.-C. Yeh and C. Lien. 2009. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Syst. Appl.* 36, 2 (2009), 2473–2480.
- [42] P. Baldi, P. Sadowski, D. Whiteson. 2014. Searching for exotic particles in high-energy physics with deep learning. *Nat. Comm.* 5, 4308 (2014).
- [43] L. M. Candanedo and V. Feldheim. 2016. Accurate occupancy detection of an office room from light, temperature, humidity and CO 2 measurements using statistical learning models. 2016. *Energy Build.* 112 (2018), 28–39.
- [44] R. K. Bock et al. 2004. Methods for multidimensional event classification: A case study using images from a Cherenkov gamma-ray telescope. *Nucl. Instrum. Methods Phys. Res. Sect. Accel. Spectrometers Detect. Assoc. Equip.* 516, 2 (2004), 511–528.
- [45] K. Fernandes, P. Vinagre, and P. Cortez. 2015. A proactive intelligent decision support system for predicting the popularity of online news. *Conference on Artificial Intelligence*, pp. 535–546.

- [46] H. Xiao, K. Rasul, and R. Vollgraf. 2017. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *ArXiv170807747 Cs Stat*, 2017.
- [47] P. Varma et al. 2017. Inferring generative model structure with static analysis. *ArXiv170902477 Cs Stat*, 2017.
- [48] G. Beatty, E. Kochis, and M. Bloodgood. 2019. The use of unlabeled data versus labeled data for stopping active learning for text classification. *IEEE International Conference on Semantic Computing*, pp. 287–294.
- [49] M. Bloodgood and K. Vijay-Shanker. 2009. A method for stopping active learning based on stabilizing predictions and the need for user-adjustable stopping. *Proc. 13th Conference on Computational Natural Language Learning*, pp. 39–47.
- [50] M. Nashaat, A. Ghosh, J. Miller, and S. Quader. WeSAL: Applying active supervision to find high-quality labels at industrial scale. *Hawaii International Conference on System Sciences*, submitted for publication.
- [51] A. C. Tan and D. Gilbert. 2003. An empirical comparison of supervised machine learning techniques in bioinformatics. *Proc. Conference on Bioinformatics*. 19 (2003), 219–222.
- [52] D. M. Powers. 2011. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. *J. Mach. Learn. Technol.* 2 (2011), 37–63.
- [53] T. Chen and C. Guestrin. 2016. XGBoost: A scalable tree boosting system. *Proc. ACM SIGKDD*, pp. 785–794.
- [54] I. Teinemaa, M. Dumas, F. M. Maggi, and C. Di Francescomarino. 2016. Predictive business process monitoring with structured and unstructured data. *International Conference on Business Process Management*, pp. 401–417.
- [55] J. Kremer, K. Steenstrup Pedersen, and C. Igel. 2014. Active learning with support vector machines. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* 4, 4 (2014), 313–326.
- [56] A. Ratner, B. Hancock, J. Dunnmon, R. Goldman, and C. Ré. 2018. Snorkel MeTaL: Weak supervision for multi-task learning. *Proc. 2nd Workshop on Data Management for End-to-End Machine Learning*.
- [57] T. Durand, N. Thome, and M. Cord. 2018. SyMIL: MinMax latent SVM for weakly labeled data. *IEEE Trans. Neural Netw. Learn. Syst.* 29, 12 (2018), 6099–6112.
- [58] R. Stewart and S. Ermon. 2017. Label-free supervision of neural networks with physics and domain knowledge. *AAAI Conference on Artificial Intelligence*, pp. 2576–2582.
- [59] L. Cao et al. 2019. Smile: A system to support machine learning on EEG data at scale. *Proc. VLDB Endow.* 12, 12 (2019), 2230–2241.
- [60] S. Wu et al. 2018. Fondue: Knowledge base construction from richly formatted data. *Proc. International Conference on Management of Data*, pp. 1301–1316.
- [61] Y. Li, Y. Wang, D. Yu, Y. Ning, P. Hu, and R. Zhao. 2020. ASCENT: Active supervision for semi-supervised learning. *IEEE Trans. Knowl. Data Eng.* 32, 5 (2020), 868–882.
- [62] P. Bachman, A. Sordoni, and A. Trischler. 2017. Learning algorithms for active learning. *Proc. International Conference on Machine Learning*. 70 (2017), 301–310.
- [63] D. Kang, D. Raghavan, P. Bailis, and M. Zaharia. 2018. Model assertions for debugging machine learning. *NeurIPS MLSys Workshop*.
- [64] M. Carbonneau, E. Granger, and G. Gagnon. 2019. Bag-level aggregation for multiple-instance active learning in instance classification problems. *IEEE Trans. Neural Netw. Learn. Syst.* 30, 5 (2019), 1441–1451.
- [65] Z. Zhou, J. Y. Shin, S. R. Gurudu, M. B. Gotway, and J. Liang. 2018. AFT*: Integrating active learning and transfer learning to reduce annotation efforts. *ArXiv180200912 Cs Stat*, 2018.

Received July 2019; revised December 2019; accepted February 2020