WeShap: Weak Supervision Source Evaluation with Shapley Values

Naiqing Guan
University of Toronto
Toronto, Canada
naiqing.guan@mail.utoronto.ca

Nick Koudas
University of Toronto
Toronto, Canada
koudas@cs.toronto.edu

ABSTRACT

Efficient data annotation stands as a significant bottleneck in training contemporary machine learning models. The Programmatic Weak Supervision (PWS) pipeline presents a solution by utilizing multiple weak supervision sources to automatically label data, thereby expediting the annotation process. Given the varied contributions of these weak supervision sources to the accuracy of PWS, it is imperative to employ a robust and efficient metric for their evaluation. This is crucial not only for understanding the behavior and performance of the PWS pipeline but also for facilitating corrective measures.

In this paper, we introduce WeShap values as an evaluation metric. This metric quantifies the average contribution of weak supervision sources within a proxy PWS pipeline, leveraging the theoretical underpinnings of Shapley values. We demonstrate efficient computation of WeShap values using dynamic programming, achieving quadratic computational complexity relative to the number of weak supervision sources.

Our experiments demonstrate the versatility of WeShap values across various applications, including the identification of beneficial or detrimental labeling functions, refinement of the PWS pipeline, comprehension of the pipeline's behavior, and scrutinizing specific instances of mislabeled data. Although initially derived from a specific proxy PWS pipeline, we empirically demonstrate the generalizability of WeShap values to other PWS pipeline configurations. Our findings indicate a noteworthy average improvement of 5.0 points in downstream model accuracy through the revision of the PWS pipeline compared to previous state-of-the-art methods, underscoring the efficacy of WeShap values in enhancing data quality for training machine learning models.

PVLDB Reference Format:
Naiqing Guan and Nick Koudas. WeShap: Weak Supervision Source Evaluation with Shapley Values. PVLDB, 18(4): 1063 - 1076, 2024.
doi:10.14778/3717755.3717766

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Gnaiqing/WeShap.

Proceedings of the VLDB Endowment, Vol. 18, No. 4 ISSN 2150-8097.
doi:10.14778/3717755.3717766

1 INTRODUCTION

Deploying modern machine learning models in new application scenarios often hinges on the availability of large annotated datasets, posing a significant bottleneck. While obtaining unlabeled data is relatively straightforward, annotating typically demands substantial effort and financial resources. Programmatic weak supervision (PWS) [46, 65] offers a promising avenue for mitigating this manual annotation burden. In the PWS paradigm, instead of annotating individual instances, users concentrate on developing multiple weak supervision sources capable of automatically annotating a portion of the data in a noisy manner. These sources may originate from crowdsourced data [34], human-designed heuristics [46, 63], or pre-trained models [5, 55]. Represented as labeling functions (LFs), these weak supervision sources assign weak labels to some data points while abstaining from others. Given the potential contradictions among weak labels, a label model is trained to denoise and aggregate them. Subsequently, the labels predicted by the label model are utilized to train the downstream machine learning model, also called the end model.

Within the PWS framework, labeling functions (LFs) are the fundamental components for automated annotation, profoundly influencing data quality and downstream machine learning model accuracy. Despite extensive research on the efficient design of LFs, with notable studies [7, 15, 26, 59], scant attention has been devoted to LF quality evaluation methodologies. Conventionally, LF evaluation relies on metrics such as accuracy (evaluated on a withheld labeled dataset), coverage, or confliction with other LFs. For instance, the Snorkel library [46]'s *LFAnalysis* module reports these metrics. However, these metrics gauge different facets of LFs, often yielding conflicting assessments. Notably, while both high coverage and accuracy are desirable, they typically exhibit a negative correlation in practice. Boeckling et al. [7] propose ranking LFs based on $(2\alpha_j - 1) * l_j$, where α_j and l_j denote the accuracy and coverage of LF j, respectively. They additionally present a theorem asserting that under assumptions of LF independence conditional on class labels and uniformly distributed LF errors, this ranking minimizes the expected risk of a certain label model on training data. However, these stringent assumptions are rarely met in practice. Furthermore, along with all aforementioned metrics, their metric overlooks the data distribution in feature space, which can significantly impact downstream model performance.

Figure 1 provides a motivating example to illustrate the limitation of existing LF evaluation metrics. There are three LFs for a binary classification task, where LF 1 and LF 3 have over 90% accuracy and 30% coverage, while LF 2 has only around 50% accuracy and 20% coverage. However, omitting LF 2 during the label model training shifts the downstream model's decision boundary to $x = 0$, leading

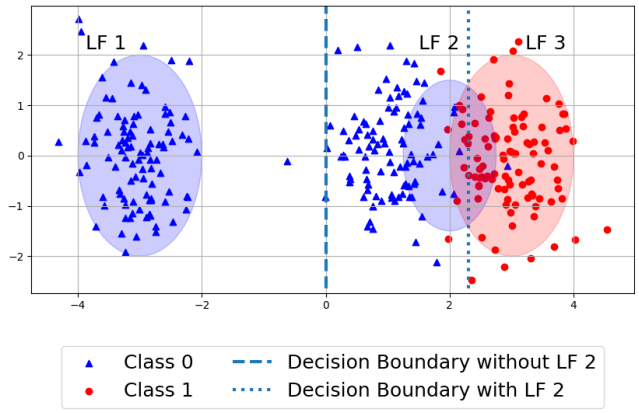


Figure 1: A motivating example for comprehensive LF evaluation. LF 2 is essential for reducing classification errors, although its accuracy is around 0.5.

to substantial classification errors. The key is that although LF 2 has lower accuracy and coverage compared to other labeling functions, it lies near the optimal decision boundary, making a greater contribution to the downstream model accuracy.

The provided example underscores the necessity for more nuanced LF evaluation metrics to assess each LF's contribution on downstream model accuracy directly. Zhang et al.'s work on Source-aware Influence Functions (SIF) [67] pioneered the exploration of comprehensive LF evaluation metrics by learning to evaluate LFs based on their influence. While SIF considers the data distribution in feature space, it falls short of directly measuring an LF's impact on the accuracy of the downstream model.

A well-established approach for allocating contributions in cooperative games is the Shapley Value [53], renowned for its fairness, symmetry, and efficiency properties [11]. Conceptually, if we regard each LF as a player in a cooperative game and utilize downstream model accuracy to gauge utility, then the Shapley value for each LF mirrors its contribution to the downstream model. However, direct computation of the Shapley value entails $O(2^m)$ computations, where m signifies the number of players in the game (equivalent to the number of LFs in our context). Moreover, evaluating a coalition of LFs proves computationally intensive, necessitating training the label model and downstream model from scratch. Consequently, direct computation of the Shapley value becomes impractical. This poses a fundamental challenge for LF evaluation based on Shapley value: *how can we efficiently compute Shapley values to evaluate LFs within a PWS framework?*

In this paper, we address the above challenge by proposing the WeShap value (**We**akly Supervised **Shap**ley Value), which measures the *exact shapley value* of LFs in a specific PWS framework, with majority voting being the label model and K-nearest neighbors (KNN) being the downstream model. We demonstrate that the Shapley value of LFs can be computed efficiently under these specific model choices. The WeShap value can be used to identify helpful or harmful LFs, revise PWS pipelines, and understand model behaviors. Our experiments show that while the WeShap value is computed based on specific model choices, it also attains excellent performance when other models, such as Snorkel [46], BERT [16], or ResNet [24] are utilized in the pipeline.

2 PRELIMINARIES

In this section, we provide background material on setting up the programmatic weak supervision framework and introduce Shapley values and their favorable properties.

2.1 Programmatic Weak Supervision

We consider C-way classification scenarios, where $\mathcal{X}$ represents the feature space, $\mathcal{Y}$ denotes the label space, and $\mathcal{P}$ stands for the underlying data distribution. Within the PWS framework, the user possesses a training dataset $D_{Train} = \{x_i\}_{i=1}^{n}$, where the corresponding labels $\{y_i\}_{i=1}^{n}$ are initially unknown. Optionally, a smaller validation set $D_{Valid} = \{x_i, y_i\}_{i=n+1}^{n+n_v}$ may exist for hyperparameter tuning. Both training and validation datasets adhere to the underlying distribution, i.e., $(x_i, y_i) \sim \mathcal{P}$. The objective is to train a machine learning model f to minimize its expected risk on $\mathcal{P}$.

To attain this objective, the user crafts a set of labeling functions (LFs) $\Lambda = \{\lambda_j\}_{j=1}^{m}$, where each LF furnishes noisy labels (termed weak labels) to a subset of data. We denote by $L_{ij} = \lambda_j(x_i)$ the weak label provided by λ_j for x_i. Here, $L_{ij} \in \{\mathcal{Y} \cup \{\emptyset\}\}$, where $\emptyset$ indicates that λ_j abstains from labeling x_i. Subsequently, the weak labels are employed to train a label model Θ, which gauges the accuracies of each LF and predicts a single probabilistic label $\tilde{y}_i$ per instance. Finally, the training dataset, coupled with the generated labels, is utilized to train the downstream model f.

2.2 Shapley Value

The Shapley value is based on cooperative game theory [17]. Formally, a cooperative game is defined by a pair (I, v), where $I = \{1, ..., m\}$ denotes the set of players and $v : 2^I \to \mathbb{R}$ is the utility function, which assigns a real value $v(S)$ to every coalition $S \subset I$. Furthermore, the utility of an empty coalition is set to 0, i.e., $v(\emptyset) = 0$. Intuitively, the utility function defines how much payoff a set of players can achieve by forming a coalition. One central question in cooperative game theory is how to distribute the total payoff fairly among the players. The Shapley value [53] is a classical solution to this question, which assigns each player their average marginal contribution to the value of the predecessor set over every permutation of the player set. Formally, the Shapley value is defined as

$$\phi_j^{Sh} = \frac{1}{|\Pi(I)|} \sum_{\pi \in \Pi(I)} \left[v(P_j^\pi \cup \{j\}) - v(P_j^\pi) \right] \tag{1}$$

Where $\Pi(I)$ denotes all possible permutations of I and P_j^π denotes the predecessor set of player j in permutation π. Intuitively, suppose the players join the coalition randomly; the Shapley value for player j would be the expectation of their marginal contribution to the payoff. An equivalent formulation is

$$\phi_j^{Sh} = \frac{1}{m} \sum_{S \in I/\{j\}} \frac{1}{\binom{m-1}{|S|}} \left[v(S \cup \{j\}) - v(S) \right] \tag{2}$$

The Shapley value is theoretically appealing as it is the unique solution that satisfies the following desiderata simultaneously:

Efficiency: The payoff of the full player set is completely distributed among all players, i.e., $\sum_{j \in I} \phi_j = v(I)$.

Null Player: If a player contributes nothing to each coalition, then they should receive zero value, i.e., $[\forall S | j \notin S, v(S \cup \{j\}) = v(S)] \Rightarrow \phi_j = 0$.

Symmetry: If two players play equal roles to each coalition, they should receive equal value, i.e., $[\forall S | i, j \notin S, v(S \cup \{i\}) = v(S \cup \{j\}] \Rightarrow \phi_i = \phi_j$.

Additivity: Given two coalition games (I, v) and (I, w) with different utility functions, the value a player receives under a coalition game $(I, v + w)$ is the sum of the values they receive under separate coalition games, i.e., $\phi_j(v + w) = \phi_j(v) + \phi_j(w)$.

In our context, the set of LFs corresponds to the players, while the accuracy of the downstream model on a holdout dataset, utilizing a coalition of LFs to label the training set, serves as the utility function. Under this framework, several desirable properties of the Shapley value emerge:

Efficiency: The accuracy of the downstream model is entirely attributed to all LFs.

Null player: LFs contributing nothing (e.g., abstaining on all data) receive a score of zero.

Symmetry: LFs contributing equally to downstream model accuracy receive identical scores.

Additivity: This property facilitates efficient score computation when the downstream model is employed across multiple applications or datasets.

These properties are crucial for fair LF evaluation within the PWS framework, rendering the Shapley value an appealing solution.

3 OUR METHOD

In this section, we first establish the formalization of the proxy PWS framework, then outline the cooperative game within this proxy framework. Subsequently, we derive the WeShap value, representing the Shapley value of LFs within the specified cooperative game structure. Next, we present an efficient approach for computing the WeShap value using dynamic programming and analyze its computational complexity. Finally, we introduce various application scenarios showcasing the utility of the WeShap value.

3.1 Proxy Framework

Definition 1 (Proxy Framework). *The proxy framework is a programmatic weak supervision framework, where the label model uses majority voting (MV) to aggregate LF outputs, and the downstream model uses the K-nearest neighbors (KNN) algorithm to make predictions.*

For an LF set Λ, let $\Theta^c_\Lambda(x)$ denote the MV model's predicted probability that unlabeled training data x belongs to class c:

$$\Theta^c_\Lambda(x) = \begin{cases} \frac{\sum_{\lambda \in \Lambda} \mathbb{1}(\lambda(x)=c)}{\sum_{\lambda \in \Lambda} \mathbb{1}(\lambda(x) \neq \emptyset)}, & \sum_{\lambda \in \Lambda} \mathbb{1}(\lambda(x) \neq \emptyset) > 0 \\ \frac{1}{C}, & \sum_{\lambda \in \Lambda} \mathbb{1}(\lambda(x) \neq \emptyset) = 0 \end{cases} \tag{3}$$

In other words, if LFs are activated on the instance, the label model uses majority voting to predict its probabilistic label; otherwise, it predicts every class with equal probabilities.

The downstream model in the proxy framework is a KNN classifier, where the user can arbitrarily specify the choice of K, the distance metric, and the weight function. Here, we assume that uniform weights are applied for simplicity. For a validation instance x_{val}, let $\mathcal{N}_K(x_{val})$ contains the K nearest neighbors of x_{val} in the training set. The predicted probability of x_{val} belongs to class c is:

$$f^c_\Lambda(x_{val}) = \frac{\sum_{i \in \mathcal{N}_K(x_{val})} \Theta^c_\Lambda(x_i)}{K} \tag{4}$$

Our framework addresses multiclass scenarios but does not encompass multi-label classification, where multiple non-exclusive labels can be assigned to each instance. Consequently, the probabilities assigned to each class in Equation 4 sum to 1.

The accuracy of the downstream model is measured on a holdout dataset $D_{val} \sim \mathcal{P}$. Note that the accuracy defined in Equation 5 differs from traditional classification accuracy, as it is based on probabilistic predictions rather than categorical ones.

$$Acc_{f,\Lambda}(D_{val}) = \frac{\sum_{(x,y) \in D_{val}} f^y_\Lambda(x)}{|D_{val}|} \tag{5}$$

Definition 2 (Proxy Game). *The proxy game is a cooperative game (I, v) defined under the proxy framework, where $I = \{1, ..., m\}$ denotes the set of LFs, and $v : 2^I \rightarrow \mathbb{R}$ maps an LF coalition to the accuracy gain of the proxy framework (measured by Equation 5) using that LF coalition compared to random prediction (i.e. with accuracy $\frac{1}{C}$).*

We use the accuracy gain instead of the absolute accuracy of the proxy framework to define the utility function v to guarantee $v(\emptyset) = 0$, which is a prerequisite for Shapley value computation.

3.2 WeShap Value

Within the proxy framework, we proceed to derive the formulation of the WeShap value, representing the Shapley values of LFs within the defined cooperative game structure.

First, we consider the marginal utility of an LF λ in classifying a data point (x, y) with the MV label model. Notice that LFs that abstain on (x, y) will not affect the prediction of the MV model, so we only consider LFs providing weak labels. Without loss of generality, we first consider the case where $\lambda(x) = y$. Suppose in a LF permutation, there are a correct LFs and b wrong LFs in the predecessor LF set of λ. The marginal utility of λ on (x, y) is:

$$\psi(a, b) = \begin{cases} \frac{a+1}{a+b+1} - \frac{a}{a+b}, & a + b > 0 \\ 1 - \frac{1}{C}, & a + b = 0 \end{cases} \tag{6}$$

As a result, we can efficiently compute the average marginal contribution of λ by enumerating the number of accurate and inaccurate LFs in the predecessor set of λ over all possible LF permutations.

Theorem 1. *Consider a coalition game $\mathcal{G}$ where a majority voting (MV) model utilizes a set of LFs to label a data point (x, y). The player set $I = \{1, ..., m\}$ denotes the set of LFs, and the utility function v' maps an LF coalition to the accuracy gain of the MV model using the LFs compared to random prediction with accuracy $\frac{1}{C}$. Suppose there are p correct LFs and w incorrect LFs on (x, y), then the Shapley value of a correct LF λ in the coalition game $\mathcal{G}$ is:*

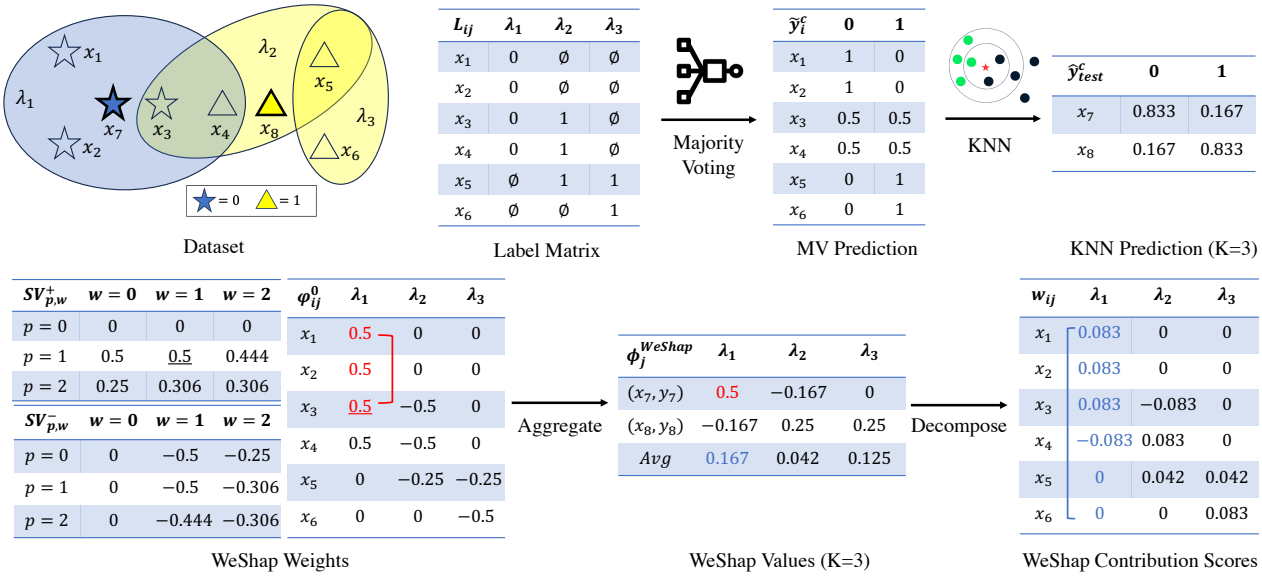


Figure 2: Illustration of the WeShap value computation.

$$SV^{+}_{p,w} = \frac{1}{p+w}\sum_{i=0}^{p-1}\sum_{j=0}^{w}\left[\psi(i,j)\frac{\binom{p-1}{i}\binom{w}{j}}{\binom{p+w-1}{i+j}}\right] \tag{7}$$

Proof. Consider all permutations of LFs that produce weak labels on (x, y). In a permutation with i correct LFs and j incorrect LFs before λ, the marginal utility of λ is $\psi(i, j)$. The number of permutations with i correct LFs and j incorrect LFs before λ is $\binom{p-1}{i}\binom{w}{j}(i+j)!(p+w-1-i-j)!$, as we need to select i correct LFs from the other $p-1$ correct LFs, j incorrect LFs from the w incorrect LFs, and consider the order of LFs in the permutation. Following the definition of Shapley value in Equation 1, the Shapley value of λ in $\mathcal{G}$ is

$$\sum_{i=0}^{p-1}\sum_{j=0}^{w}\left[\frac{\psi(i,j) * \binom{p-1}{i}\binom{w}{j}(i+j)!(p+w-1-i-j)!}{(p+w)!}\right] \tag{8}$$

This equals to $SV^{+}_{p,w}$ expressed in Equation 7. □

As the Shapley value has the efficiency property, we can calculate the Shapley value of an inaccurate LF on (x, y) as

$$SV^{-}_{p,w} = \frac{\frac{p}{p+w} - \frac{1}{C} - p * SV^{+}_{p,w}}{w}, \qquad w > 0 \tag{9}$$

We use φ^{c}_{ij} to denote the Shapley value of λ_j with respect to the MV label model in classifying x_i when the hidden label of x_i is c. The φ^{c}_{ij} value can be computed directly as

$$\varphi^{c}_{ij} = \begin{cases} SV^{+}_{p_i,w_i}, & \lambda_j(x_i) = c \\ 0, & \lambda_j(x_i) = \emptyset \\ SV^{-}_{p_i,w_i}, & otherwise \end{cases} \tag{10}$$

where p_i and w_i denote the number of correct and incorrect LFs in Λ on (x_i, y_i) respectively. We name the φ^{c}_{ij} values **WeShap weights**, which correspond to the average contribution of λ_j in classifying x_i with respect to the MV model, when the hidden label of x_i is c.

Example. *Figure 2 illustrates a running example, where we consider three LFs denoted as λ_1 through λ_3, each contributing to the labeling of data instances. The unlabeled training points, x_1 through x_6, are augmented with labeled validation points, x_7 and x_8, utilized for LF evaluation.*

In the bottom left of Figure 2, we demonstrate the WeShap weights of the LFs on unlabeled training points. As a concrete example, we calculate the Weshap weight of λ_1 on x_3 when $y_3 = 0$. Notice that 1 correct LF (λ_1) and 1 wrong LF (λ_2) are activated on it. Following Equation 7 and 10, the Weshap weight of λ_1 on x_3 is $\frac{\psi(0,0)+\psi(0,1)}{2} = 0.5$. The first part corresponds to the marginal utility of λ_1 in the permutation $[\lambda_1, \lambda_2]$, and the second part corresponds to that in $[\lambda_2, \lambda_1]$. The WeShap weight indicates that λ_1's marginal contribution to the classification of x_3 is 0.5, averaging across all LF permutations under the MV model.

Next, we focus on the downstream KNN model. Given a validation instance (x_{val}, y_{val}), let $\mathcal{N}_K(x_{val})$ denote the K-nearest neighbors of x_{val} in the training dataset. The **WeShap value** of λ_j on (x_{val}, y_{val}) is defined as:

$$\phi_j^{WeShap}(x_{val}, y_{val}) = \frac{1}{K}\sum_{i\in\mathcal{N}_K(x_{val})}\varphi_{ij}^{y_{val}} \tag{11}$$

Similarly, the WeShap value of an LF for a holdout dataset D_{val} is defined as the average WeShap value of the LF with respect to the instances inside the dataset:

$$\phi_j^{WeShap}(D_{val}) = \frac{\sum_{(x,y)\in D_{val}}\phi_j^{WeShap}(x,y)}{|D_{val}|} \tag{12}$$

Theorem 2. *The WeShap value of an LF is equal to its Shapley value in the proxy game using the same set of LFs and holdout dataset.*

Proof. Let's focus on a single validation instance (x_{val}, y_{val}) first. The utility of a coalition of LFs $S \subset \Lambda$ in the proxy game is:

$$\begin{aligned} v(S) &= f_S^{y_{val}}(x_{val}) - \frac{1}{C} && \text{(Definition 2)} \\ &= \frac{1}{K}\left[\sum_{i \in \mathcal{N}_K(x_{val})} \Theta_S^{y_{val}}(x_i)\right] - \frac{1}{C} && \text{(Equation 4)} \\ &= \frac{1}{K}\left[\sum_{i \in \mathcal{N}_K(x_{val})} \left(\Theta_S^{y_{val}}(x_i) - \frac{1}{C}\right)\right] \end{aligned} \tag{13}$$

Next, we define K coalition games $\{\mathcal{G}_i : i \in \mathcal{N}_K(x_{val})\}$. In each game, the LFs are used to classify one point (x_i, y_{val}) using the MV label model. The utility of each coalition game is defined as the label model's predictive accuracy on (x_i, y_{val}) minus $\frac{1}{C}$.

Notice that $\Theta_S^{y_{val}}(x_i) - \frac{1}{C}$ in Equation 13 is the utility of S in game $\mathcal{G}_i$. Following the additivity property of the Shapley value, the Shapley value of an LF λ_j in the proxy game is the average of the Shapley values it receives in $\{\mathcal{G}_i : i \in \mathcal{N}_K(x_{val})\}$. Following Theorem 1, the Shapley value of λ_j receives in $\mathcal{G}_i$ is $\varphi_{ij}^{y_{val}}$. Therefore, the Shapley value of λ_j in the proxy game is $\frac{1}{K}\sum_{i \in \mathcal{N}_K(x_{val})} \varphi_{ij}^{y_{val}}$, which is exactly the WeShap value.

Since we have proved the WeShap value is the Shapley value of a LF in the proxy game when D_{val} contains a single instance, following the additivity property of Shapley value and the definition in Equation 12, we can conclude that the WeShap value is also the Shapley value of a LF when D_{val} contains multiple instances. □

The WeShap value can also be generalized to KNN with non-uniform weights. To apply weighted KNN, we can simply modify Equation 11 as:

$$\phi_j^{WeShap}(x_{val}, y_{val}) = \frac{\sum_{i \in N_K(x_{val})} \left(\omega(x_i, x_{val}) * \varphi_{ij}^{y_{val}}\right)}{\sum_{i \in N_K(x_{val})} \omega(x_i, x_{val})} \tag{14}$$

Where $w(x_i, x_{val})$ specify the weight given to x_i when predicting x_{val}.

Finally, we can decompose the WeShap value in another way, enabling us to inspect each weak label's contribution. We define the **WeShap contribution scores** as:

$$w_{ij} = \frac{\sum_{(x_{val}, y_{val}) \in D_{val} : i \in \mathcal{N}_K(x_{val})} \varphi_{ij}^{y_{val}}}{|D_{val}| * K} \tag{15}$$

The WeShap contribution scores serve as detailed metrics quantifying the impact of weak labels $\lambda_j(x_i)$ on the accuracy of the proxy PWS pipeline. A higher contribution score suggests greater assistance to the pipeline's accuracy.

Example. *In Figure 2, Following Equation 11, the WeShap score of λ_1 on (x_7, y_7) (marked in red) is the average of $\{\varphi_{1,1}^0, \varphi_{2,1}^0, \varphi_{3,1}^0\}$, as $\{x_1, x_2, x_3\}$ are the KNNs of x_7 in the training data and $y_7 = 0$. The result is 0.5, indicating that λ_1 makes an average contribution of 0.5 in the proxy framework for predicting x_7.*

The WeShap Contribution Scores table at the bottom right of Figure 2 is the decomposition of WeShap values. We observe negative scores for w_{32} and w_{41}, indicating detrimental effects on the downstream model's performance. w_{32} is negative because λ_2 misclassify x_3 as class 1 and hurts the prediction of x_7 in D_{val}. Similarly, w_{41} is negative because λ_1 misclassify x_4 as class 0 and hurts the prediction of x_8. Accordingly, we can enhance the PWS pipeline by excluding these weak labels (setting them to $\emptyset$). Following this adjustment, the downstream model achieves a perfect accuracy of 1.0, underscoring the effectiveness of WeShap contribution scores in refining the PWS pipeline.

3.3 Computational Complexity

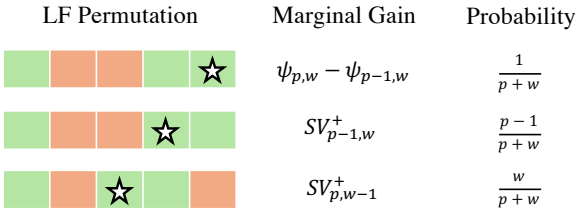


Figure 3: Efficient computation of WeShap scores using dynamic programming. The target LF λ is denoted in star; green and red cells represent LFs making correct and wrong predictions, respectively.

The computational complexity of computing the WeShap value primarily stems from calculating the WeShap weights outlined in Equation 10 and identifying the K-nearest neighbors for each validation instance. Brute-force KNN search entails a complexity of $O(n_{val}dn)$ (where d is the feature dimensionality), but this can be optimized to $O(n_{val}d \log n)$ by employing spatial data structures like KD-Trees [6], with an additional $O(n \log n)$ complexity for constructing the KD-Tree.

Let's focus on computing the WeShap weights, which can be derived from $SV^+_{p,w}$ and $SV^-_{p,w}$ values. Directly computing all $SV^+_{p,w}$ values following Equation 7 requires $O(m^4)$ computations, but this can be optimized further using dynamic programming. As illustrated in Figure 3, we categorize all possible permutations of LFs into three cases:

- The target LF λ is the last in the permutation;
- Some other correct LF is the last in the permutation;
- Some incorrect LF is the last in the permutation.

In the first case, the marginal contribution of λ is $\psi(p-1, w)$; in the second case, the average marginal contribution of λ is $SV^+_{p-1,w}$; and in the last case, the average marginal contribution of λ is $SV^+_{p,w-1}$. This yields the following recursive formula:

$$SV^+_{p,w} = \frac{\psi(p-1, w)}{p+w} + \frac{p-1}{p+w} SV^+_{p-1,w} + \frac{w}{p+w} SV^+_{p,w-1} \tag{16}$$

with the boundary values being

$$SV^+_{p,0} = \frac{C-1}{C*p}, \quad SV^+_{0,w} = 0 \tag{17}$$

Algorithm 1: Dynamic Programming Algorithm for WeShap Value Computation

Input: D_{train}: unlabeled training dataset
D_{val}: labeled holdout dataset
Λ: labeling function set
C: number of candidate classes
K: number of neighbors in the proxy KNN model
δ: distance metric in the proxy KNN model
ω: weight function in the proxy KNN model
Output: ϕ_j^{WeShap}: WeShap values

1 $n, m, n_{val} \leftarrow |D_{train}|, |\Lambda|, |D_{val}|$
2 $SV^+_{*,*}, w_{*,*} \leftarrow 0$
3 **for** $p \leftarrow 1$ *to* m **do**
4 $\quad SV^+_{p,0} \leftarrow \frac{C-1}{C*p}$
5 $\quad$ **for** $w \leftarrow 1$ *to* m **do**
6 $\quad\quad SV^+_{p,w} \leftarrow \frac{\psi(p-1,w)}{p+w} + \frac{p-1}{p+w} SV^+_{p-1,w} + \frac{w}{p+w} SV^+_{p,w-1}$
7 $\quad\quad SV^-_{p,w} \leftarrow \frac{\frac{p}{p+w} - \frac{1}{C} - p*SV^+_{p,w}}{w}$
8 $f \leftarrow KNN(D_{train}, K, \delta)$
9 **for** $(x_{val}, y_{val}) \in D_{val}$ **do**
10 $\quad \mathcal{N}_K(x_{val}) \leftarrow f.KNeighbors(x_{val})$
11 $\quad R \leftarrow n_{val} * \sum_{x_i \in \mathcal{N}_K(X_{val})} \omega(x_i, x_{val})$
12 $\quad$ **for** $x_i \in \mathcal{N}_K(X_{val})$ **do**
13 $\quad\quad p_i \leftarrow \sum_\lambda \mathbb{1}(\lambda(x_i) = y_{val})$
14 $\quad\quad w_i \leftarrow \sum_\lambda \mathbb{1}(\lambda(x_i) \neq \emptyset) - p_i$
15 $\quad\quad$ **for** $\lambda_j \in \Lambda$ **do**
16 $\quad\quad\quad$ **if** $\lambda_j(x_i) = y_{val}$ **then**
17 $\quad\quad\quad\quad w_{ij} \leftarrow w_{ij} + SV^+_{p_i,w_i} * \frac{\omega(x_i,x_{val})}{R}$
18 $\quad\quad\quad$ **else**
19 $\quad\quad\quad\quad$ **if** $\lambda_j(x_i) \neq \emptyset$ **then**
20 $\quad\quad\quad\quad\quad w_{ij} \leftarrow w_{ij} + SV^-_{p_i,w_i} * \frac{\omega(x_i,x_{val})}{R}$
21 **for** $j \leftarrow 1$ *to* m **do**
22 $\quad \phi_j^{WeShap} \leftarrow \sum_i w_{ij}$

Leveraging the recursive formula, we can compute all $SV^+_{p,w}$ values in $O(m^2)$ computations. The $SV^-_{p,w}$ values can be computed in $O(m^2)$ computations using Equation 9. These values are then mapped to the WeShap weights. Algorithm 1 demonstrates the algorithm to compute WeShap values using dynamic programming. The algorithm first utilizes Equation 16 to compute WeShap weights (Line 2-7). Then, it learns a KNN model to get the K nearest neighbors of each validation instance (Line 8) and aggregates WeShap weights to compute the WeShap contribution scores (Line 9-20). Finally, it aggregates WeShap contribution scores to get the WeShap values of each LF (Line 21-22). Assuming data structures like KD-Trees are applied for KNN search, the total computational complexity of Algorithm 1 is $O(n \log n + n_{val} d \log n + K n_{val} m + m^2)$, which generalizes well to large LF sets, improving significantly over the original $O(2^m)$ time complexity for computing exact Shapley values.

3.4 Use Cases

The WeShap value proves valuable in several applications, elucidated in this section.

Identify Helpful/Harmful LFs: Higher WeShap values indicate greater contributions to the proxy pipeline, suggesting helpfulness, while negative values imply potential harm. This aids in filtering out detrimental LFs and optimizing resource allocation. For instance, if LFs stem from multiple supervision sources, users can allocate more resources to the sources yielding the most beneficial LFs.

Enhance Downstream Model Accuracy: We can refine LF outputs by silencing those with WeShap contribution scores below a threshold θ:

$$\tilde{L}_{ij} = \begin{cases} L_{ij} & w_{ij} \geq \theta \\ \emptyset & w_{ij} < \theta \end{cases} \tag{18}$$

The threshold θ is tuned to optimize the PWS pipeline accuracy on the validation set.

Understanding PWS Pipeline Behaviors: WeShap scores identify LFs and training points with the highest or lowest contributions to validation instances, aiding in comprehending pipeline behaviors and diagnosing mispredictions. This is particularly useful when an LF indirectly influences predictions through nearby instances.

Fair Credit Allocation: Leveraging Shapley value properties (Section 2.2), WeShap ensures fair attribution of credits to each LF, valuable when distributing payoffs among multiple contributors.

4 EXPERIMENTS

We have undertaken comprehensive experimentation to assess the efficacy of WeShap values across a spectrum of downstream tasks. These tasks encompass identifying beneficial LFs, enhancing the PWS pipeline's accuracy, and analyzing its behaviors.

4.1 Setup

Datasets. We evaluate our work extensively on 9 datasets, including 5 datasets for text classification (YouTube [2], IMDB [40], Yelp [70], TREC [35], MedAbs [52]), 2 datasets for tabular classification (Census [32], Mushroom (MUSH) [43]), and 2 datasets for image classification (Indoor-Outdoor (IND) [57], and VOC07-Animal (VOC-A) [18] [1]). These datasets have been widely used to evaluate PWS pipelines in prior works [26, 57, 67, 69]. Table 1 outlines the dataset details and LF statistics. To assess LF quality in a scaled context, we generated LFs for the datasets using specific criteria:

For textual datasets, we employed LFs denoted as $\lambda_{k,c}$, which label class c upon detecting a unigram k in the text.

For tabular datasets, we utilized LFs denoted as $\lambda_{e,c}$, where class c is assigned based on the truth value of expression e. We designed LFs for the Mushroom dataset and utilized the LF set introduced by [4] for the Census dataset.

[1]This is a binary classification version of the VOC-2007 dataset detecting whether an animal exist in the picture or not.

Table 1: Dataset summary statistics.

Dataset	Task	Dataset Statistics				LF Statistics				
		#Class	#Train	#Valid	#Test	#LF	Acc	Cov	Overlap	Conflict
YouTube	spam classification	2	1686	120	250	100	0.875	0.038	0.037	0.011
IMDB	sentiment analysis	2	20000	2500	2500	100	0.666	0.026	0.024	0.017
Yelp	sentiment analysis	2	30400	3800	3800	100	0.698	0.027	0.024	0.013
TREC	question classification	6	5033	500	500	100	0.526	0.039	0.037	0.031
MedAbs	disease classification	5	6002	3095	2657	100	0.444	0.025	0.023	0.018
Mushroom	mushroom classification	2	6499	812	813	56	0.793	0.271	0.271	0.216
Census	income classification	2	10083	5561	16281	83	0.787	0.054	0.053	0.015
Indoor-Outdoor	image classification	2	1226	408	410	226	0.921	0.043	0.043	0.012
VOC07-Animal	image classification	2	4008	1003	4952	296	0.950	0.042	0.042	0.012

For image datasets, we employed the Azure Image Tagging API [2] to associate tags with images corresponding to their visual features (e.g., sky, plant). Subsequently, we considered LFs denoted as $\lambda_{t,c}$, which assign class c based on the existence of tag t.

We ensured LF quality by maintaining their accuracy at least 0.1 above random guessing on validation sets.

PWS Pipeline. We follow the standard PWS pipeline: train the label model on unlabeled data using LFs, exclude instances without active LFs, and then train the downstream model on remaining data with label model predictions. We evaluate pipeline performance via downstream model accuracy on the test set.

We assess two label models: majority voting and Snorkel MeTaL [47] implemented in WRENCH [69]. For downstream models, we consider two scenarios:

(1) Feature extraction: Use a frozen pretrained model to extract features, then train an end model on these features.

(2) Fine-tuning: Directly fine-tune the downstream model on weakly labeled data.

For feature extraction, we use dataset-specific feature extractors (Sentence-BERT [49] for YouTube, TREC, MedAbs; Bertweet-sentiment [45] for IMDB, Yelp; ResNet-50 [24] for image datasets). We then train a logistic regression end model. For fine-tuning, we use BERT base [16] for text and ResNet-50 [24] for images. We set $batch\_size = 32, n\_epochs = 5, lr = 5e-5, weight\_decay = 0$ when fine-tuning BERT and $batch\_size = 64, n\_epochs = 50, lr = 1e{-}4, weight\_decay = 1e{-}5$ when fine-tuning the ResNet-50 model with an AdamW optimizer [38] and apply early stopping technique to prevent overfitting. We don't fine-tune on tabular datasets lacking corresponding pretrained models. Each experiment is repeated five times with different seeds, reporting averaged results.

Baselines. We compare the following methods.

Random (RND): Assigns random values to LFs as a baseline.

Accuracy (ACC): Evaluates LFs by validation set accuracy: $V(\lambda_j) = Acc(\lambda_j)$.

Coverage (COV): Evaluates LFs by validation set coverage: $V(\lambda_j) = Cov(\lambda_j)$.

IWS [7]: Combines accuracy and coverage: $V(\lambda_j) = (2 * Acc(\lambda_j) - 1) * Cov(\lambda_j)$. This corresponds to evaluating the LF based on the correct prediction count minus the wrong prediction count on the validation set.

MC-Shap [10]: Approximates Shapley value using Monte Carlo permutation sampling, evaluating gain by end model accuracy on validation set. To balance approximation accuracy and runtime cost, we use $n = 100$ samples in our evaluation.

SIF [67]: Learns fine-grained source-aware influence functions computed on the validation set: $V(\lambda_j) = |\sum_{i,c} \bar{\phi}_{i,j,c}|$, where $\bar{\phi}_{i,j,c}$ is the source-aware influence function of λ_j on x_i with respect to class c.

WeShap: Our proposed method using Shapley values. We optimize K (5-40), distance metric, and weight function for each dataset based on proxy KNN classifier accuracy on the validation set.

4.2 LF Evaluation

In assessing LF quality, we confront the challenge posed by the varying scales of LF evaluation metrics. To standardize this process, we implement the following methodology: initially, we arrange LFs in descending order based on their evaluation metrics, mitigating the influence of scale discrepancies. Subsequently, we adopt an iterative approach where we progressively select the top-p LFs and utilize them to train both the label and downstream models. Our evaluation commences with the top-10 LFs and incrementally expands the LF subset size at intervals of 10 until it encompasses all LFs within the dataset. We present the average accuracy of the downstream model on the test set across these iterations in Table 2. As this evaluation process requires training the downstream model repeatedly, which is time-consuming for the fine-tuning scenario, we only evaluated the feature-extraction scenario for the LF ranking experiments.

Table 2 illustrates the superiority of WeShap values in ranking beneficial LFs. While WeShap does not consistently yield the optimal result, it demonstrates robust performance across diverse datasets. For both label model choices, WeShap achieves the highest average downstream model accuracy. Notably, WeShap exhibits particular efficacy when using Snorkel as the label model — a common PWS configuration in recent studies [26, 67, 69]. In this scenario, WeShap outperforms the second-best metric (MC-Shap) by 1.4 points and the third-best metric (SIF) by 3.6 points on average

[2] https://learn.microsoft.com/en-us/azure/ai-services/computer-vision/concept-tagging-images(last accessed: 01/11/2024)

Table 2: Average downstream model accuracy after ranking LFs based on different metrics.

LM	Metric	Youtube	IMDB	Yelp	MedAbs	TREC	MUSH	Census	IND	VOC-A	AVG
MV	RND	0.825	0.831	0.877	0.497	0.536	0.876	0.782	0.889	0.934	0.783
	ACC	0.813	**0.842**	0.888	0.489	0.583	0.840	0.809	0.884	0.940	0.787
	COV	0.834	0.838	0.849	0.498	0.574	0.834	0.798	0.885	0.937	0.783
	IWS	0.789	0.840	0.856	0.485	**0.605**	**0.900**	**0.818**	0.885	0.946	0.792
	MC-Shap	0.818	0.837	0.858	**0.556**	0.539	0.895	0.813	**0.910**	0.930	0.795
	SIF	**0.854**	0.838	0.872	0.507	0.587	0.885	0.802	0.888	0.951	0.798
	WeShap	0.844	0.836	**0.900**	0.522	0.566	0.893	**0.818**	0.901	**0.952**	**0.803**
Snorkel	RND	0.735	0.778	0.690	0.434	0.475	0.835	0.781	0.879	0.882	0.721
	ACC	0.667	0.796	0.722	0.445	0.504	0.864	0.778	0.877	0.922	0.730
	COV	0.685	0.814	0.621	0.452	0.479	0.829	0.763	0.883	0.883	0.712
	IWS	0.665	**0.821**	0.629	0.441	**0.544**	**0.892**	0.758	0.886	0.893	0.725
	MC-Shap	0.811	0.782	0.743	0.466	0.480	0.886	0.778	0.891	**0.948**	0.754
	SIF	0.727	0.793	0.741	0.440	0.460	0.882	**0.793**	0.883	0.869	0.732
	WeShap	**0.827**	0.810	**0.836**	**0.470**	0.473	0.883	0.775	**0.897**	0.938	**0.768**

while significantly reducing runtime, as demonstrated in Figure 8. The most substantial improvement is observed in the Yelp dataset, where WeShap shows a remarkable 9.3-point increase over the next best metric.

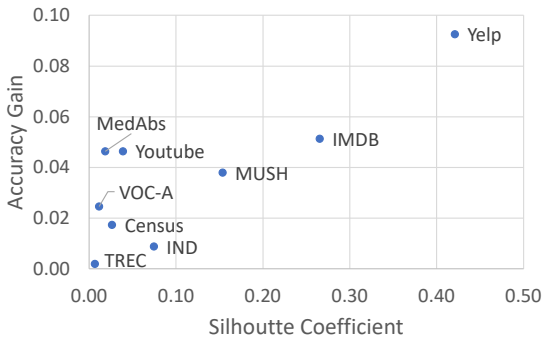


Figure 4: Average downstream model accuracy gain of WeShap compared to random baseline in ranking LFs.

To discern scenarios where WeShap values reliably rank LFs, we analyze the data distribution across the evaluated datasets and the characteristics associated with LFs. Notably, we observe a strong correlation between WeShap's LF ranking performance and the smoothness of the training data.

The smoothness assumption, a cornerstone in semi-supervised learning, posits that points in a high-dimensional space proximal to each other should share similar labels [58]. To quantify data smoothness, we employ the silhouette coefficient [50], where a higher coefficient denotes superior smoothness. Given the varying factors affecting absolute end model accuracy across datasets, we assess WeShap's efficacy by measuring its average downstream model accuracy gain relative to a random baseline.

Figure 4 depicts the silhouette coefficient plotted against the downstream model accuracy gain in the proxy framework. Evidently, a positive correlation ($r = 0.810$) emerges between dataset smoothness and WeShap's accuracy gain. The rationale lies in WeShap's utilization of KNN as the downstream model within the proxy pipeline, wherein the KNN model's performance directly hinges on data smoothness. Consequently, if the dataset exhibits poor smoothness, the KNN model's accuracy diminishes, thereby impairing its efficacy as a proxy model for identifying beneficial LFs. The low smoothness in the TREC dataset likely contributes to WeShap's poor performance, a finding corroborated by the inferior performance of the Sentence-BERT encoder on TREC, as reported in [49]. While the encoder's poor performance affects all evaluated metrics, it has a more pronounced effect on WeShap, which uniquely relies on the smoothness assumption, as previously discussed.

To better illustrate the effect of LF set size on downstream model performance, Figure 5 depicts the downstream model accuracy as we incorporate more LFs in the selected set across three datasets. As the dataset's smoothness level increases, WeShap demonstrates greater performance gains than baseline methods. In the TREC dataset, WeShap's performance is comparable to the random baseline, whereas in the Yelp dataset, it significantly outperforms all other metrics. Notably, WeShap exhibits performance advantages from the early stages in the VOC-Animal and Yelp datasets, with as few as tens of LFs selected. This underscores WeShap's utility in identifying the most beneficial LFs from a large set. Another observation is that when the metric reliably evaluates LF helpfulness (e.g., WeShap on Yelp), the downstream model accuracy resembles an inverted U-shape as the number of LFs increases. This occurs because we first include helpful LFs, followed by harmful ones in the selected LF set. The position of the curve's peak is dataset-dependent. Conversely, if the metric distinguishes between helpful and harmful LFs poorly, the downstream model accuracy tends to change monotonically as more LFs are incorporated into the set.

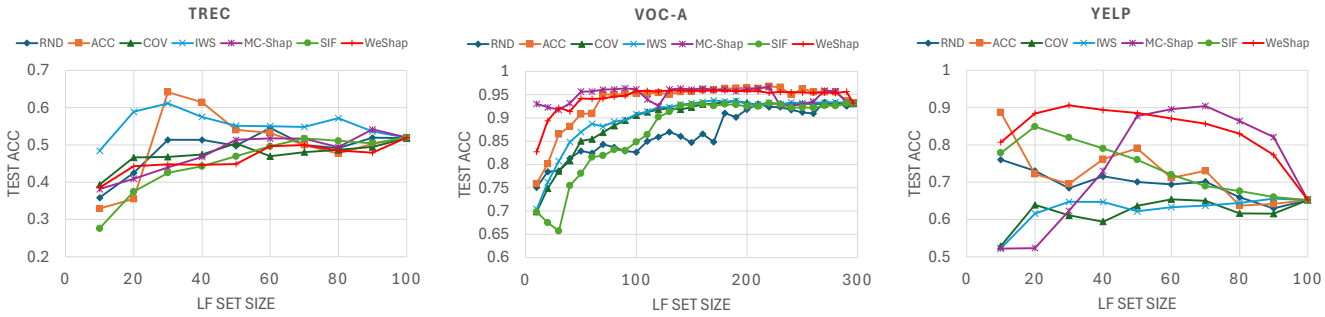


Figure 5: Ranking LFs on selected datasets.

4.3 PWS Pipeline Revision

Subsequently, we assess the impact of WeShap scores on refining the PWS pipeline regarding the downstream model's accuracy post-refinement. We explore two distinct refinement strategies: (1) Pruning: Eliminating detrimental LFs with low evaluation scores, and (2) Fine-Grained Revision: Modifying specific LF outputs or soft labels predicted by the label model.

Both WeShap and SIF support fine-grained revision, thus enabling the evaluation of both options for these methods. We designate the approaches as follows: WeShap-P (or SIF-P) for pruning out harmful LFs and WeShap-F (or SIF-F) for fine-grained revision. As for other metrics, we solely evaluate their efficacy in pruning out harmful LFs, as they lack support for fine-grained revision.

The revision pipeline operates as follows: For LF pruning, we commence by ranking the LFs based on the evaluation metric, subsequently selecting the top-p LFs for training the label model. We tune the threshold p to optimize the PWS pipeline's accuracy on the validation dataset utilizing Optuna [1] with a Tree-structured Parzen Estimator (TPE) Sampler for 20 trials.

For fine-grained revision, we adjust threshold θ for muting LF outputs (as described in Section 3.4) for WeShap-F, and threshold α for perturbing train losses for SIF-F (which serves a similar purpose as θ for WeShap, as elaborated in [67]). We optimize these thresholds to maximize the PWS pipeline's accuracy on the validation dataset using Optuna with a TPE Sampler for 100 trials.

Additionally, we include two baselines for comparison: the *Base* method reflects the downstream model's accuracy without any revision, while the *Golden* method showcases the downstream model's accuracy when trained on golden training labels without weak supervision. These baselines provide insight into the performance of specific revision methods and their disparity compared to using golden labels.

Table 3 presents various revision methods' performance under feature extraction and fine-tuning scenarios. We evaluate the MC-Shap metric exclusively in the feature extraction scenario, as computing MC-Shap in fine-tuning scenarios is prohibitively time-consuming. Note that the MUSH and Census datasets are tabular ones that do not have corresponding pre-trained models for fine-tuning, so we omit the fine-tuning results on these two datasets. Due to space limitations, we report the figures when Snorkel is used as the label model. The performance comparison results are similar when using the MV model.

We first focus on the feature-extraction scenario. In the context of LF pruning, both the WeShap and MC-Shap methods emerge as frontrunners, showcasing an advantage of 1.3 points over alternative pruning strategies. The result indicates the effectiveness of the Shapley value in ranking LFs. When integrating fine-grained supervision, the fine-grained revision methodologies surpass traditional LF pruning techniques. Particularly, WeShap-F distinguishes itself by outshining all other baseline methods in 6 out of 9 datasets, with the MedAbs dataset witnessing the most substantial leap in performance, an 11.7-point increase. On average, across nine datasets, WeShap-F significantly boosts downstream model accuracy by 11.2 points over using original LFs and exceeds the performance of state-of-the-art (SOTA) revision techniques by 5.0 points. For the fine-tuning scenario, we observe similar trends, where WeShap outperforms other baseline methods in 6 out of 7 datasets and achieves competitive results in the remaining one, surpassing the performance of SOTA revision techniques by 3.9 points on average.

Surprisingly, WeShap demonstrates superior performance when revising the PWS pipeline on the TREC dataset despite its mediocre ranking of LFs on the same dataset. We hypothesize that this discrepancy arises because fine-grained revision performance is more influenced by local data smoothness (where neighboring data points share similar labels) than global data smoothness (where data points with similar labels cluster together). To test this hypothesis, we trained a KNN model on TREC using ground-truth labels, achieving an accuracy of 73.2%. This result suggests that TREC exhibits good local smoothness despite its poor global smoothness, as indicated by a low Silhouette score.

4.4 Understanding Pipeline Behaviors

We present a case study on the VOC-Animal dataset to demonstrate how WeShap values can be used to understand and improve the PWS pipeline through human-in-the-loop intervention. The study uses Snorkel as the label model, Resnet-50 as the feature extractor, and logistic regression as the downstream model. Class Y indicates the presence of animals, while N indicates their absence.

Initially, the test set accuracy was 0.920. Table 4 shows LF statistics for the VOC07-Animal dataset, ranked by WeShap values. The top LF, assigning Y when the image is tagged "animal," has a

Table 3: Downstream model accuracy after revising the PWS pipeline. Tabular datasets (MUSH, Census) do not have corresponding pre-trained models for fine-tuning.

Scenario	Metric	YouTube	IMDB	Yelp	MedAbs	TREC	MUSH	Census	IND	VOC-A	AVG
Feature Extraction (LogReg)	Base	0.744	0.811	0.655	0.433	0.487	0.906	0.751	0.886	0.932	0.734
	ACC	0.663	0.845	0.839	0.468	0.613	0.957	0.769	0.909	0.949	0.779
	COV	0.718	0.809	0.638	0.445	0.496	0.900	0.764	0.898	0.786	0.717
	IWS	0.712	0.840	0.757	0.477	0.621	0.862	0.764	0.901	0.808	0.749
	MC-Shap	0.841	0.808	0.911	0.479	0.517	0.927	0.788	0.904	**0.959**	0.793
	SIF-P	0.747	0.807	0.857	0.475	0.526	0.890	0.790	0.880	0.810	0.754
	SIF-F	**0.917**	0.844	0.895	0.430	0.549	0.914	0.781	**0.911**	0.925	0.796
	WeShap-P	0.868	0.827	0.905	0.486	0.496	0.927	0.764	0.905	0.950	0.792
	WeShap-F	0.910	**0.852**	**0.927**	**0.603**	**0.632**	**0.990**	**0.838**	0.909	0.952	**0.846**
	Golden	*0.914*	*0.873*	*0.942*	*0.646*	*0.922*	*1.000*	*0.807*	*0.929*	*0.972*	*0.889*
Finetuning (BERT/ Resnet-50)	Base	0.877	0.789	0.770	0.445	0.495	–	–	0.880	0.855	0.730
	ACC	0.856	0.812	0.830	0.409	0.590	–	–	0.905	0.977	0.768
	COV	0.890	0.803	0.684	0.482	0.466	–	–	0.885	0.778	0.712
	IWS	0.890	0.797	0.635	0.501	0.516	–	–	0.886	0.926	0.736
	SIF-P	0.881	0.806	0.827	0.466	0.454	–	–	0.887	0.715	0.719
	SIF-F	**0.934**	0.857	0.866	0.489	0.517	–	–	**0.914**	0.946	0.789
	WeShap-P	0.849	0.822	0.872	0.473	0.495	–	–	0.877	0.920	0.758
	WeShap-F	0.919	**0.882**	**0.942**	**0.539**	**0.621**	–	–	**0.914**	**0.978**	**0.828**
	Golden	*0.968*	*0.888*	*0.955*	*0.644*	*0.960*	–	–	*0.918*	*0.981*	*0.902*

Table 4: LF statistics for VOC07-Animal (Snorkel-LogReg).

LF	Accuracy	Coverage	Overlap	Conflict	WeShap
animal→Y	0.992	0.223	0.223	0.221	0.024
mammal→Y	0.997	0.154	0.154	0.152	0.015
vehicle→N	0.977	0.342	0.342	0.027	0.012
...	...	...	...	...	...
tree→N	0.626	0.161	0.161	0.068	-0.007
outdoor→N	0.660	0.604	0.604	0.229	-0.022

positive WeShap value of 0.024, indicating it improves the proxy downstream model (KNN) accuracy by an average of 0.024. The second LF, "mammal→Y," also has a positive WeShap value of 0.015. Based on these findings, we added three new animal-related LFs ("cat→Y," "dog→Y," and "bird→Y"), increasing test set accuracy to 0.934. Following that, we observed that the LF "outdoor→N" had a negative WeShap value of -0.022 despite an accuracy of 0.66. Removing this LF improved accuracy to 0.941. Lastly, noting that high WeShap values corresponded to high-accuracy LFs, we increased the LF selection threshold to 0.7, further improving test set accuracy to 0.957. These revisions led to a total accuracy improvement of 0.034, a substantial enhancement given that the downstream model's accuracy using ground truth labels is 0.972.

WeShap values also offer a versatile application in identifying the most influential LFs and training data pertinent to specific test instances. Illustrated in Figure 6, we showcase a subset of images from the validation dataset that the downstream model mispredicts. Subsequently, we compute WeShap scores for these mispredicted images individually.

The influential LFs are identified as those with the lowest WeShap scores, while the influential training images are determined by the images associated with the lowest WeShap contribution scores. Our analysis reveals that the mispredictions predominantly stem from certain LFs associated with the absence of animals. Consequently, users can opt to discard or downweight these LFs to rectify these mispredictions.

4.5 Ablation Studies

We assess the sensitivity of WeShap scores to various configuration choices, including the number of neighbors (K), distance metrics, weight functions, and holdout dataset size. Our default settings are K=10, Euclidean distance, and uniform weight. We evaluate performance using downstream model accuracy after fine-grained PWS pipeline revision (WeShap-F). We test the following configuration choices:

Number of neighbors (K): 5, 10, 20, and 40.
Distance metrics: Cosine, Euclidean, and Manhattan.
Weight functions: Uniform (Equation 11) and inverse distance (Equation 14).
Holdout dataset sizes: 50, 100, 200, 400, and 800 (for datasets with over 800 validation points).

The findings, as depicted in Figure 7, demonstrate the robustness of WeShap scores in enhancing the PWS pipeline, irrespective of variations in K values, distance metrics, and weight functions. Moreover, enlarging the holdout dataset consistently enhances the performance of the PWS pipeline. Specifically, doubling the holdout dataset size correlates with a 1.2-point increase in downstream model accuracy across our experiments. Remarkably, even with

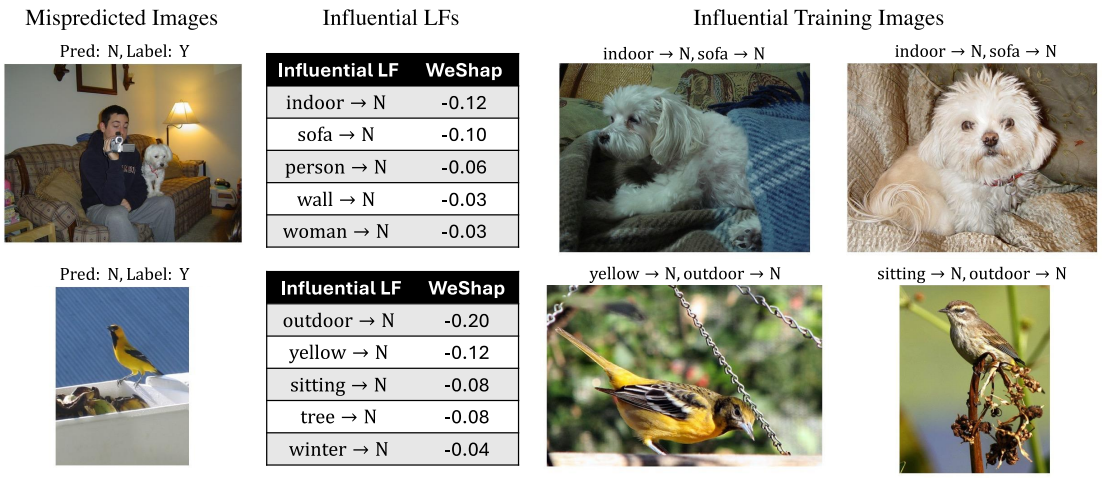
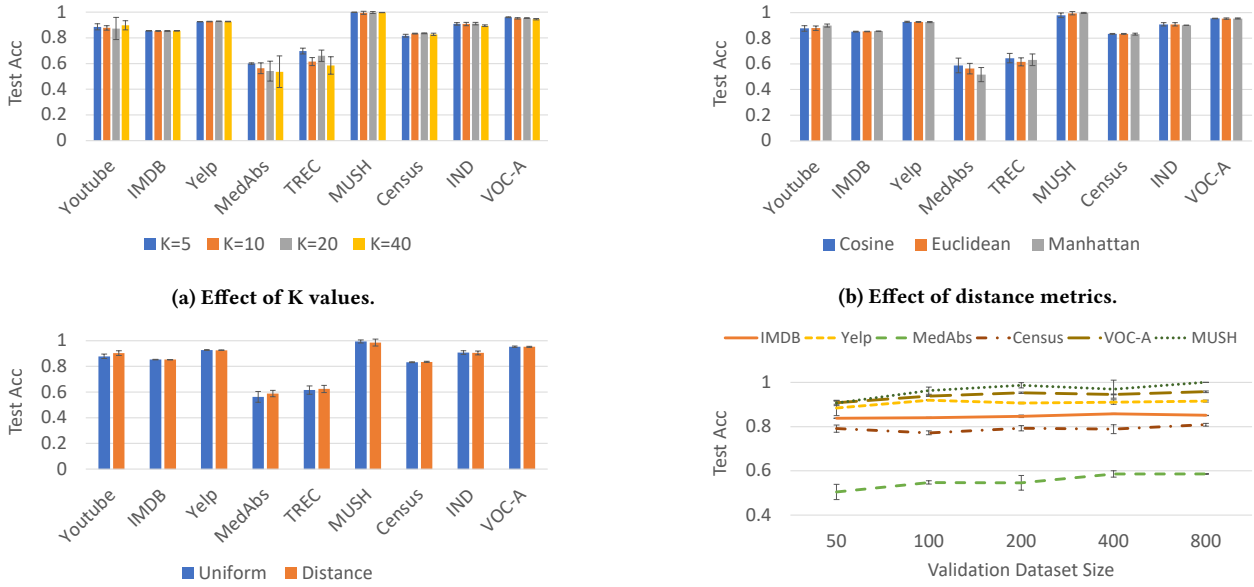


Figure 6: Analyze mispredictions in VOC07-Animal dataset (Snorkel-LogReg). Class "Y" denotes the presence of animals, and "N" denotes the absence of them.



(a) Effect of K values.

(b) Effect of distance metrics.

(c) Effect of weight functions.

(d) Effect of holdout dataset size.

Figure 7: Effect of WeShap configurations for PWS pipeline revision (Snorkel-Logreg).

as few as 50 labeled data points, WeShap scores contribute to a substantial 5.7-point average improvement in downstream model accuracy compared to the original PWS pipeline. These results underscore the continued utility of WeShap scores, particularly in scenarios with constrained labeling budgets.

4.6 Runtime

The experiments were conducted on a 4-core Intel(R) Xeon(R) Gold 5122 CPU (3.60 GHz) with an NVIDIA TITAN Xp GPU (12 GB memory). Multiprocessing and GPU acceleration were used for MC-Shap and SIF score computations, respectively. Figure 8 shows the runtimes for LF evaluation metrics, excluding label model and end model training times. Results are averaged over five runs, with error bars indicating standard deviation.

WeShap scores were computed within seconds across all datasets, while SIF and MC-Shap required significantly longer — often hours to days, even with parallelization. On average, SIF and MC-Shap took 356 and 5,300 times longer than WeShap, respectively. This difference is due to their computational complexity: SIF calculates

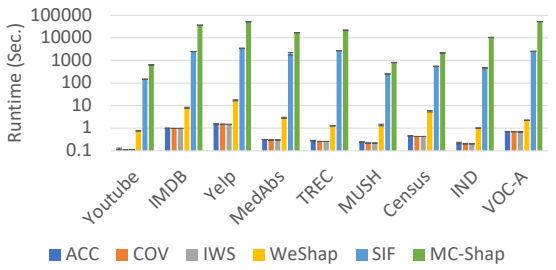


Figure 8: Runtime for different LF evaluation metrics.

Hessian-vector products recursively, while MC-Shap requires training both label and downstream models $m \times n$ times (where m is the LF set cardinality and n is the Monte Carlo sample size). In our evaluation, this resulted in tens of thousands of model training iterations. Consequently, WeShap offers a more efficient and lightweight approach to LF evaluation.

5 DISCUSSIONS

In this section, we compare WeShap with selected baseline methods in our evaluation, highlighting its key advantages and limitations.

Weshap vs SIF. Both methods offer fine-grained influence decomposition but differ in their approach. SIF uses the *(i-j-c) effect* to quantify the weight of the label model for L_{ij} in predicting $P(y_i = c)$. WeShap employs *WeShap weights* φ_{ij}^c to measure the *average contribution* of L_{ij} in predicting $P(y_i = c)$. The key distinction is that while SIF focuses on current label model weights, WeShap considers all possible LF subsets to compute average contributions.

For example, when assessing λ_j's impact on x_i in the MV label model, the (i-j-c) effect is binary (1 when $L_{ij} = c$, 0 otherwise). In contrast, the WeShap weight φ_{ij}^c accounts for other LFs' outputs on x_i, providing a more comprehensive view of LF contribution.

Our evaluation shows that WeShap outperforms SIF in ranking LFs and improving PWS pipeline accuracy while being computationally more efficient. While approaches like FastIF [22] could potentially be applied to accelerate SIF, they are unlikely to bridge the runtime gap with WeShap fully. Optimizing influence score computation remains an open research area, though beyond our current scope. In summary, WeShap's broader reflection of LF contributions and efficiency give it practical advantages over SIF, despite both offering valuable insights into LF influence within the PWS pipeline.

WeShap vs MC-Shap. Both WeShap and MC-Shap are founded on Shapley values. MC-Shap's primary advantage lies in its ability to use the exact label and end models for Shapley value computation, potentially yielding more accurate LF evaluation results compared to proxy models. However, the Monte Carlo sampling process introduces estimation errors that may offset this advantage. As our demonstrations have shown, MC-Shap values are computationally expensive, rendering it impractical to evaluate a large number of samples within a reasonable timeframe. Furthermore, unlike WeShap, MC-Shap does not support fine-grained decomposition or revision, leading to inferior performance in PWS revision tasks.

Limitations. We acknowledge several limitations of WeShap values in this study. Firstly, our theoretical analysis is predicated on a specific proxy PWS setting, as discussed in Section 3.1. Consequently, the WeShap values may not accurately reflect the ground-truth Shapley values of LFs in alternative settings. Extending the theoretical guarantees of WeShap to broader contexts remains an intriguing avenue for future research. Secondly, while WeShap demonstrates robust performance across the evaluated datasets, our analysis reveals that its efficacy is influenced by the underlying dataset's smoothness. Future investigations could explore techniques such as contrastive learning [30] to enhance data smoothness, potentially improving LF evaluation accuracy.

6 RELATED WORK

Programmatic Weak Supervision. In the programmatic weak supervision framework [46, 48, 65], users design labeling functions (LFs) that come from various sources to label large datasets efficiently, such as heuristics [46, 63], pre-trained models [5, 55], external knowledge bases [25, 36], and crowd-sourced labels [34]. Researchers have also explored approaches to automate the LF design process [21, 27, 55, 59] or guide users to develop LFs more efficiently [7, 15, 26]. On the other hand, there is rich literature on learning the models to aggregate LFs and de-noise the weak labels [4, 19, 46, 47, 60, 62, 66, 68]. While the work on programmatic weak supervision is abundant, few works focused on the rigorous evaluation of LFs. Work on LF design [7, 21, 23, 26] usually use simple heuristics like empirical accuracy or coverage to evaluate and prune out LFs, which does not support fine-grained analysis and revision. The most relevant work is WS Explainer (SIF) [67], which leverages the influence function [31] to evaluate the influence of each weak supervision source.

Shapley Values. Shapley value [53], originated from game theory and has been applied in machine learning tasks including feature selection [13, 56, 61], data evaluation [20, 29, 33, 54], deep learning explanation [3, 12, 64] and federated learning [37]. The wide application of the Shapley value is credited to its favorable properties that include fairness, symmetry, and efficiency [11, 51]. However, the high computational complexity of the Shapley value needs to be addressed before applying it in practice. Common approaches to efficiently computing approximate Shapley values include Monte-Carlo permutation sampling [8–10, 41], multilinear extension [42, 44] and linear regression approximation [14, 39]. While computing the exact Shapley value is infeasible in most cases, it is possible in specific settings. Specifically, Jia et al. [28] demonstrated that the Shapley value for data evaluation can be efficiently computed for nearest-neighbor algorithms.

7 CONCLUSIONS

In our study, we propose WeShap values as an innovative method for assessing and refining Programmatic Weak Supervision (PWS) sources. We demonstrate notable computational efficiency and versatility across various datasets and PWS configurations. Our results unveil an average downstream model accuracy enhancement of 5.0 points compared to conventional methods, highlighting the pivotal contribution of WeShap values in the progression of machine learning models.

REFERENCES

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[2] Túlio C Alberto, Johannes V Lochter, and Tiago A Almeida. 2015. Tubespam: Comment spam filtering on youtube. In *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*. IEEE, 138–143.

[3] Marco Ancona, Cengiz Oztireli, and Markus Gross. 2019. Explaining deep neural networks with a polynomial time algorithm for shapley value approximation. In *International Conference on Machine Learning*. PMLR, 272–281.

[4] Abhijeet Awasthi, Sabyasachi Ghosh, Rasna Goyal, and Sunita Sarawagi. 2019. Learning from Rules Generalizing Labeled Exemplars. In *International Conference on Learning Representations*.

[5] Stephen H Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alex Ratner, Braden Hancock, Houman Alborzi, et al. 2019. Snorkel drybell: A case study in deploying weak supervision at industrial scale. In *Proceedings of the 2019 International Conference on Management of Data*. 362–375.

[6] Jon Louis Bentley. 1975. Multidimensional binary search trees used for associative searching. *Commun. ACM* 18, 9 (1975), 509–517.

[7] Benedikt Boecking, Willie Neiswanger, Eric Xing, and Artur Dubrawski. 2021. Interactive Weak Supervision: Learning Useful Heuristics for Data Labeling. In *International Conference on Learning Representations*.

[8] Mark Alexander Burgess and Archie C Chapman. 2021. Approximating the Shapley Value Using Stratified Empirical Bernstein Sampling.. In *IJCAI*. 73–81.

[9] Javier Castro, Daniel Gómez, Elisenda Molina, and Juan Tejada. 2017. Improving polynomial estimation of the Shapley value by stratified random sampling with optimum allocation. *Computers & Operations Research* 82 (2017), 180–188.

[10] Javier Castro, Daniel Gómez, and Juan Tejada. 2009. Polynomial calculation of the Shapley value based on sampling. *Computers & operations research* 36, 5 (2009), 1726–1730.

[11] Georgios Chalkiadakis, Edith Elkind, and Michael Wooldridge. 2022. *Computational aspects of cooperative game theory*. Springer Nature.

[12] Jianbo Chen, Le Song, Martin J. Wainwright, and Michael I. Jordan. 2019. L-Shapley and C-Shapley: Efficient Model Interpretation for Structured Data. In *International Conference on Learning Representations*.

[13] Shay Cohen, Gideon Dror, and Eytan Ruppin. 2007. Feature selection via coalitional game theory. *Neural computation* 19, 7 (2007), 1939–1961.

[14] Ian Covert and Su-In Lee. 2021. Improving kernelshap: Practical shapley value estimation using linear regression. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 3457–3465.

[15] Benjamin Denham, Edmund MK Lai, Roopak Sinha, and M Asif Naeem. 2022. Witan: unsupervised labelling function generation for assisted data programming. *Proceedings of the VLDB Endowment* 15, 11 (2022), 2334–2347.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *North American Chapter of the Association for Computational Linguistics*. https://api.semanticscholar.org/CorpusID:52967399

[17] Edith Elkind and Jörg Rothe. 2016. Cooperative game theory. *Economics and computation: an introduction to algorithmic game theory, computational social choice, and fair division* (2016), 135–193.

[18] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. 2007. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. Last accessed: 01/11/2024.

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

[20] Amirata Ghorbani and James Zou. 2019. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*. PMLR, 2242–2251.

[21] Naiqing Guan, Kaiwen Chen, and Nick Koudas. 2023. Can large language models design accurate label functions? *arXiv preprint arXiv:2311.00739* (2023).

[22] Han Guo, Nazneen Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. 2021. FastIF: Scalable Influence Functions for Efficient Model Interpretation and Debugging. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 10333–10350.

[23] Braden Hancock, Martin Bringmann, Paroma Varma, Percy Liang, Stephanie Wang, and Christopher Ré. 2018. Training classifiers with natural language explanations. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, Vol. 2018. NIH Public Access, 1884.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[25] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*. 541–550.

[26] Cheng-Yu Hsieh, Jieyu Zhang, and Alexander Ratner. 2022. Nemo: Guiding and Contextualizing Weak Supervision for Interactive Data Programming. *Proc. VLDB Endow.* 15, 13 (sep 2022), 4093–4105. https://doi.org/10.14778/3565838.3565859

[27] Tzu-Heng Huang, Catherine Cao, Spencer Schoenberg, Harit Vishwakarma, Nicholas Roberts, and Frederic Sala. 2023. Scriptoriumws: A code generation assistant for weak supervision. In *ICLR Deep Learning for Code Workshop*.

[28] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gurel, Bo Li, Ce Zhang, Costas Spanos, and Dawn Song. 2018. Efficient task specific data valuation for nearest neighbor algorithms. *Proceedings of the VLDB Endowment* 12, 11 (2018), 1610–1623.

[29] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J Spanos. 2019. Towards efficient data valuation based on the shapley value. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 1167–1176.

[30] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *Advances in neural information processing systems* 33 (2020), 18661–18673.

[31] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*. PMLR, 1885–1894.

[32] Ron Kohavi. 1996. Census Income. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5GP7S. Last accessed: 01/11/2024.

[33] Yongchan Kwon, Manuel A Rivas, and James Zou. 2021. Efficient computation and analysis of distributional shapley values. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 793–801.

[34] Ouyu Lan, Xiao Huang, Bill Yuchen Lin, He Jiang, Liyuan Liu, and Xiang Ren. 2020. Learning to Contextually Aggregate Multi-Source Supervision for Sequence Labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2134–2146.

[35] Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

[36] Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. 2020. Bond: Bert-assisted open-domain named entity recognition with distant supervision. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 1054–1064.

[37] Zelei Liu, Yuanyuan Chen, Han Yu, Yang Liu, and Lizhen Cui. 2022. Gtg-shapley: Efficient and accurate participant contribution evaluation in federated learning. *ACM Transactions on Intelligent Systems and Technology (TIST)* 13, 4 (2022), 1–21.

[38] Ilya Loshchilov and Frank Hutter. 2018. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

[39] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems* 30 (2017).

[40] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*. 142–150.

[41] Sasan Maleki, Long Tran-Thanh, Greg Hines, Talal Rahwan, and Alex Rogers. 2013. Bounding the estimation error of sampling-based Shapley value approximation. *arXiv preprint arXiv:1306.4265* (2013).

[42] Rory Mitchell, Joshua Cooper, Eibe Frank, and Geoffrey Holmes. 2022. Sampling permutations for shapley value estimation. *Journal of Machine Learning Research* 23, 43 (2022), 1–46.

[43] Mushroom 1981. Mushroom. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5959T. Last accessed: 01/11/2024.

[44] Ramin Okhrati and Aldo Lipani. 2021. A multilinear sampling algorithm to estimate shapley values. In *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 7992–7999.

[45] Juan Manuel Pérez, Juan Carlos Giudici, and Franco Luque. 2021. pysentimiento: A Python Toolkit for Sentiment Analysis and SocialNLP tasks. arXiv:2106.09462 [cs.CL]

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

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

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

[49] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Conference on Empirical Methods in Natural*

Language Processing. https://api.semanticscholar.org/CorpusID:201646309

[50] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20 (1987), 53–65.

[51] Benedek Rozemberczki, Lauren Watson, Péter Bayer, Hao-Tsung Yang, Olivér Kiss, Sebastian Nilsson, and Rik Sarkar. 2022. The Shapley Value in Machine Learning. In *The 31st International Joint Conference on Artificial Intelligence and the 25th European Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 5572–5579.

[52] Tim Schopf, Daniel Braun, and Florian Matthes. 2022. Evaluating unsupervised text classification: zero-shot and similarity-based approaches. In *Proceedings of the 2022 6th International Conference on Natural Language Processing and Information Retrieval*. 6–15.

[53] Lloyd S Shapley. 1953. A value for n-person games. *Contribution to the Theory of Games* 2 (1953).

[54] Dongsub Shim, Zheda Mai, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. 2021. Online class-incremental continual learning with adversarial shapley value. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 9630–9638.

[55] Ryan Smith, Jason A Fries, Braden Hancock, and Stephen H Bach. 2022. Language models in the loop: Incorporating prompting into weak supervision. *ACM/JMS Journal of Data Science* (2022).

[56] Xin Sun, Yanheng Liu, Jin Li, Jianqi Zhu, Huiling Chen, and Xuejie Liu. 2012. Feature evaluation and selection with cooperative game theory. *Pattern recognition* 45, 8 (2012), 2992–3002.

[57] Wee Hyong Tok, Amit Bahree, and Senja Filipi. 2021. *Practical Weak Supervision*. " O'Reilly Media, Inc.".

[58] Jesper E Van Engelen and Holger H Hoos. 2020. A survey on semi-supervised learning. *Machine learning* 109, 2 (2020), 373–440.

[59] Paroma Varma and Christopher Ré. 2018. Snuba: Automating weak supervision to label training data. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, Vol. 12. NIH Public Access, 223.

[60] Paroma Varma, Frederic Sala, Shiori Sagawa, Jason Fries, Daniel Fu, Saelig Khattar, Ashwini Ramamoorthy, Ke Xiao, Kayvon Fatahalian, James Priest, et al. 2019. Multi-resolution weak supervision for sequential data. *Advances in Neural Information Processing Systems* 32 (2019).

[61] Brian Williamson and Jean Feng. 2020. Efficient nonparametric statistical inference on population feature importance using Shapley values. In *International conference on machine learning*. PMLR, 10282–10291.

[62] Renzhi Wu, Shen-En Chen, Jieyu Zhang, and Xu Chu. 2023. Learning Hyper Label Model for Programmatic Weak Supervision. In *The Eleventh International Conference on Learning Representations*.

[63] Yue Yu, Simiao Zuo, Haoming Jiang, Wendi Ren, Tuo Zhao, and Chao Zhang. 2021. Fine-Tuning Pre-trained Language Model with Weak Supervision: A Contrastive-Regularized Self-Training Approach. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1063–1077.

[64] Hao Zhang, Yichen Xie, Longjie Zheng, Die Zhang, and Quanshi Zhang. 2021. Interpreting multivariate shapley interactions in dnns. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 10877–10886.

[65] Jieyu Zhang, Cheng-Yu Hsieh, Yue Yu, Chao Zhang, and Alexander Ratner. 2022. A survey on programmatic weak supervision. *arXiv preprint arXiv:2202.05433* (2022).

[66] Jieyu Zhang, Bohan Wang, Xiangchen Song, Yujing Wang, Yaming Yang, Jing Bai, and Alexander Ratner. 2022. Creating Training Sets via Weak Indirect Supervision. In *International Conference on Learning Representations*.

[67] Jieyu Zhang, Haonan Wang, Cheng-Yu Hsieh, and Alexander J Ratner. 2022. Understanding programmatic weak supervision via source-aware influence function. *Advances in Neural Information Processing Systems* 35 (2022), 2862–2875.

[68] Jieyu Zhang, Yujing Wang, Yaming Yang, Yang Luo, and Alexander Ratner. 2022. Binary classification with positive labeling sources. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 4672–4676.

[69] Jieyu Zhang, Yue Yu, Yinghao Li, Yujing Wang, Yaming Yang, Mao Yang, and Alexander Ratner. 2021. WRENCH: A Comprehensive Benchmark for Weak Supervision. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[70] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems* 28 (2015).