Understanding Programmatic Weak Supervision via Source-aware Influence Function

Jieyu Zhang\textsuperscript{1}; Haonan Wang\textsuperscript{2}; Cheng-Yu Hsieh\textsuperscript{1}, Alexander Ratner\textsuperscript{1,3}

\textsuperscript{1}University of Washington \textsuperscript{2}University of Illinois Urbana-Champaign \textsuperscript{3}Snorkel AI, Inc.

\{jieyuz2, cydhsieh, ajratner\}@cs.washington.edu

\{haonan3\}@illinois.edu

Abstract

Programmatic Weak Supervision (PWS) aggregates the source votes of multiple weak supervision sources into probabilistic training labels, which are in turn used to train an end model. With its increasing popularity, it is critical to have some tool for users to understand the influence of each component (e.g., the source vote or training data) in the pipeline and interpret the end model behavior. To achieve this, we build on Influence Function (IF) and propose source-aware IF\textsuperscript{2}, which leverages the generation process of the probabilistic labels to decompose the end model’s training objective and then calculate the influence associated with each (data, source, class) tuple. These primitive influence score can then be used to estimate the influence of individual component of PWS, such as source vote, supervision source, and training data. On datasets of diverse domains, we demonstrate multiple use cases: (1) interpreting incorrect predictions from multiple angles that reveals insights for debugging the PWS pipeline, (2) identifying mislabeling of sources with a gain of 9%-37% over baselines, and (3) improving the end model’s generalization performance by removing harmful components in the training objective (13%-24% better than ordinary IF).

1 Introduction

One of the major bottlenecks for deploying modern machine learning models is the need for a substantial amounts of human-labeled training data, often annotated over long periods of time and at great expense. As machine learning models become increasingly powerful but also data hungry, new “data-centric” AI development workflows and systems have emerged, wherein the labeling and development of this training data is positioned and supported as the central development activity. One recent and increasingly popular type of data-centric AI development uses Programmatic Weak Supervision (PWS), wherein users focus on developing a diversity of noisy, programmatic supervision sources \cite{zhang2022snorkel, wang2022auto3, hsieh2022programmatic} to programmatically annotate training data in an efficient way. Specifically, these weak supervision sources, e.g., heuristics, knowledge bases, and pre-trained models, are often abstracted as labeling functions (LFs) \cite{wang2022auto3}, which is a user-defined program that provides potentially noisy labels for some subset of the data. So far, different modeling techniques have been developed to aggregate the noisy votes of LFs to produce training labels (often referred to as a label model) \cite{zhang2022snorkel, wang2022auto3, hsieh2022programmatic, ratner2018snorkel}. Finally, these training labels are in turn used to train an end model for the downstream classification tasks. In this study, we focus on this two-stage PWS pipeline \cite{ratner2018snorkel}.

With the increasing popularity of PWS in various applications \cite{zhang2022snorkel, wang2022auto3, hsieh2022programmatic, wang2022auto2}, it is of great importance to provide practitioners with a tool that helps them understand the behavior of a trained end model.

\textsuperscript{2}These authors contributed equally to this work.

\textsuperscript{3}The implementation is available at the WRENCH benchmark: https://github.com/JieyuZ2/wrench

as an effect of each upstream component (e.g., source vote, training data, LFs, etc.) involved in the model’s training process. First, by understanding what are the deciding factors that lead to a model’s specific prediction, users are able to verify whether such influence is desirable in terms of critical aspects such as model safety and fairness [11, 35, 16]. Secondly, since developing LFs is still inevitably a partly manual process in real-world applications and may involve human expert in an iterative process [5, 20, 51], it is beneficial to offer users with feedback of how individual component (like each LF) influences the end model performance on the downstream task. Finally, the rendered understanding of (in)efficacy of the PWS pipeline can be naturally exploited to automatically improve the end model performance.

Towards the goal of developing tools to help understand the model behavior, Influence Function (IF) [22, 15, 8] has been recently applied to interpret the model prediction as influence of the training data for large-scale applications. However, most of existing IF studies are unaware of the generation process of training labels like PWS and only estimate the influence of the training data [22, 3]. Notably, the group IF study [23] has attempted to apply IF on PWS, but it simply estimates the influence of each LF as summation of data influence and is limited to a specified PWS without learnable label model. In this work, we aim to develop a general framework that helps estimate the influence of each PWS’s component on the end model and is aware of (yet computationally agnostic to) the choice of label model. To achieve this, we leverage the knowledge of how the probabilistic labels are synthesized from sources and accordingly decompose the training loss of each data into multiple terms, each of which is associated with what we called (i, j, c)-effect, i.e., the effect of j-th LF on c-th dimension (corresponding to class c) of the i-th data’s probabilistic label. With such a training loss decomposition, we propose two means of calculating our source-aware IF, namely, reweighting and weight-moving, based on different perturbations on the training respectively. Each estimated source-aware influence score can be interpreted as the influence of removing the (i, j, c)-effect from the pipeline. By simply aggregating over specific dimension(s), it can be utilized to estimate the influence of different components of PWS including (1) each source vote, (2) each LF and (3) each training data, since the influence score is additive [23].

In experiments, we exploit the source-aware IF in a diversity of 13 classification datasets including tabular, text, and image data, and present multiple applications. First, we show that source-aware IF can help to understand the behavior of the end model in terms of the PWS components; in particular, when interpreting the same incorrect prediction made by two PWS pipelines with different label models, data-level IF might lead to the same influential training data, whereas source-aware IF is capable of revealing the most responsible LF or source vote. Second, we use source-aware IF as a tool to identify mislabeling of each LF and show that it outperforms baselines with a significant margin of 9%-37%. Finally, we demonstrate that the training loss decomposition and source-aware IF enable fine-grained training loss perturbation and consequently lead to better test loss improvement (13%-24% improvement over ordinary IF [22] and group IF [23]).

To summarize, this work makes the following contributions. First, we propose source-aware IF tailored for PWS that incorporates our knowledge of the generation process of training labels. Second, we develop two means of calculating the source-aware IF based on different training loss perturbations. Third, we demonstrate a variety of use cases of the source-aware IF and conduct extensive quantitative and qualitative experiments on real datasets in different domains to verify the efficacy of source-aware IF on understanding and improving PWS pipelines.

2 Preliminary

We denote scalars and generic items as lowercase letters, vectors as lowercase bold letters, and matrices as bold uppercase letters. We use superscripts to index generic items, e.g., the i-th component of a vector, or the vector output of a function f(·). All derivation details are deferred to the appendix.

Programmatic Weak Supervision. We target at a C-way classification problem and have a training dataset of size N, i.e., \( D = \{ x^{(i)}, y^{(i)} \}_{i \in [N]} \), where \( x^{(i)} \in \mathcal{X} = \mathbb{R}^{d} \) and \( y^{(i)} \in \mathcal{Y} = \{1, \ldots, C\} \). The ground truth \( y^{(i)} \)'s are not observed. We have M labeling functions (LFs) \( \{ \lambda^{(j)}(\cdot) \}_{j \in [M]} \). Each \( \lambda^{(j)}(\cdot) \in \{-1\} \cup \mathcal{Y}^{(j)} \), where \( \mathcal{Y}^{(j)} \subseteq \mathcal{Y} \). That is, each \( \lambda^{(j)}(\cdot) \) either abstains (-1) or outputs a specific label \( y \in \mathcal{Y}^{(j)} \) on a given data point. We define the label matrix as \( L_{ij} = \lambda^{(j)}(x^{(i)}) \),

\[ x^{(i)} = \text{Programmatic Weak Supervision}. \]
We aim for a general framework for explaining and debugging the PWS pipeline through Influence
H.
The change of loss w.r.t. one sample is often interpreted as its effect on model predictions over the

Then, as shown in previous work \[24\], the changes of the model parameters due to the introduction of the weight

section, we propose the source-aware IF tailored for PWS and describe its applications.

\begin{align}
\theta^* = \arg\min_{\theta} \hat{L}(\hat{D}; \theta) = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y^{(i)}, f_\theta(x^{(i)})) = \arg\min_{\theta} -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} \hat{y}^{(i)}_c \ell(c, f_\theta(x^{(i)})),
\end{align}

where \(\theta^*\) is the minimizer of the loss \(\hat{L}(\hat{D}; \theta)\) over the training set \(\hat{D}\).

**Influence Function.** The Influence Function (IF) estimate how the minimizer \(\theta^*\) of the empirical
loss \(\ell\) would change if we were to reweight the \(i\)-th training example \(z^{(i)} = (x^{(i)}, y^{(i)})\) by \(\epsilon_i\). The key idea is to make a first-order Taylor approximation of the changes in \(\theta^*\) around \(\epsilon_i = 0\). Specifically, if the \(i\)-th training sample is upweighted by a small \(\epsilon_i\), the perturbed risk minimizer \(\theta^*_\epsilon\) becomes:

\begin{align}
\theta^*_\epsilon \triangleq \arg\min_{\theta \in \Theta} \frac{1}{N} \sum_{i'=1}^{N} \ell\left(y^{(i')}, f_\theta(x^{(i')})\right) + \epsilon_i \ell\left(y^{(i)}, f_\theta(x^{(i)})\right).
\end{align}

The changes of the model parameters due to the introduction of the weight \(\epsilon_i\) are:

\begin{align}
\theta^*_\epsilon - \theta^* \approx \frac{d\theta^*_\epsilon}{d\epsilon_i} \bigg|_{\epsilon_i=0} = -H_{\theta^*}^{-1} \nabla \theta \ell\left(y^{(i)}, f_{\theta^*}(x^{(i)})\right) \epsilon_i,
\end{align}

where \(H_{\theta^*} = \frac{1}{N} \sum_{i'} \nabla_{\theta} \ell\left(y^{(i')}, f_{\theta^*}(x^{(i')})\right)\) is the Hessian of the objective at \(\theta^*\) and assumed to be positive definite \[22\]. Similarly, the change of loss with respect to a single sample \(z' = (x', y')\) is

\begin{align}
\phi_i(z') &= \frac{d}{d\epsilon_i} \left(\ell(y', f_{\theta^*}(x')) - \ell(y, f_{\theta^*}(x))\right) \bigg|_{\epsilon_i=0} \\
&= -\frac{d}{d\epsilon_i} \ell(y', f_{\theta^*}(x')) \bigg|_{\epsilon_i=0} = H_{\theta^*}^{-1} \nabla \theta \ell(y^{(i)}, f_{\theta^*}(x^{(i)})).
\end{align}

Then, as shown in previous work \[24\], the change of loss over a holdout set, e.g., the validation set
\(D_v\), is simply the sum of the changes over every sample in the set:

\begin{align}
\phi_i(D_v) = \mathcal{L}(D_v; \theta^*_{\epsilon_i}) - \mathcal{L}(D_v; \theta^*) = \frac{1}{|D_v|} \sum_{z' \in D_v} \phi_i(z').
\end{align}

The change of loss w.r.t. one sample is often interpreted as its effect on model predictions over the

training set \[22\], while the change of loss over a holdout set can be leveraged to improve the test loss by discarding/downweighting the training examples with negative impact \[38, 42, 24\]. Considering the extraordinary influence that might be caused by mislabeled examples, the relative influence function (RelatIF) \[3\] has been proposed to measure the influence of a sample relative to its global effects. The RelatIF can be defined as

\[\phi_i(z') = \frac{\phi_i(z')}{\sqrt{\phi_i(z^{(i)})}}\]

**3 Methodology**

Given an unlabeled training set \(\{x^{(i)}\}_{i \in |X|}\) and \(M\) labeling functions (LFs) \(\{\lambda^{(j)}(\cdot)\}_{j \in |M|}\), we study the scenario where a Programmatic Weak Supervision (PWS) pipeline is employed to synthesize
probabilistic labels \(\{\hat{y}^{(i)}\}_{i \in |X|}\) for training an end model \(f_\theta(x)\) for a \(C\)-way classification problem. We aim for a general framework for explaining and debugging the PWS pipeline through Influence Function (IF) \[22\], which estimates the influence of each training data on test loss/prediction for a better understanding of model behavior and is able to improve/debug the training data accordingly. In the PWS pipeline, we have full knowledge of how training labels are synthesized from different sources, i.e., LFs, which opens the opportunity to trace the influence in a source-aware way. In this section, we propose the source-aware IF tailored for PWS and describe its applications.
We proceed with the noise-aware loss as
\[
\ell_i^{(i)}(\hat{y}_i), \mathcal{L}_0(x^{(i)}_c) = - \frac{1}{C} \sum_{c=1}^{C} \frac{\sigma(\sum_{j=1}^{M} W_{j,i,L_{ij,c}})}{\sum_{k=1}^{C} \sigma(\sum_{j=1}^{M} W_{j,i,L_{ij,k}})} \log (\mathcal{L}_0(x^{(i)}_c)_c).
\]

We proceed with the \(\sigma(\cdot)\) being the identity function because it enables the loss decomposition introduced below. We will later discuss other choices of \(\sigma(\cdot)\). By simple rearrangement, we could decompose the loss over the training set into the \emph{summation} of all \(N \times M \times C\) terms:
\[
\hat{\mathcal{L}}(\mathcal{D}; \theta) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{c=1}^{C} \hat{\ell}_{i,j,c}(\theta)_c \quad \hat{\ell}_{i,j,c}(\theta)_c = - \frac{W_{j,i,L_{ij,c}}}{\sum_{k=1}^{C} \sum_{j=1}^{M} W_{j,i,L_{ij,k}}} \log (\mathcal{L}_0(x^{(i)}_c)_c).
\]

Such a decomposition is source-aware in the sense that it preserves how the source votes are aggregated via the label model, and each \(\hat{\ell}_{i,j,c}(\theta)\) indeed corresponds to the \((i, j, c)\)-effect.

### 3.2 Source-aware Influence Function

As a consequence of training loss decomposition, we are now able to compute the influence score for each term \(\hat{\ell}_{i,j,c}(\theta)\), which is the influence of removing \(\hat{\ell}_{i,j,c}(\theta)\) or, in other words, the \((i, j, c)\)-effect from training. We provide two means of calculating the influence score: \textbf{reweighting} and \textbf{weight-moving}, which correspond to two different ways of perturbing the training loss. Specifically, given
an (unseen) instance $z' = (x', y')$ and the cross-entropy loss $\ell(y', f_\theta(x'))$, we are interested in the influence score defined by the change of loss $\ell(y', f_\theta(x'))$ caused by the differences in the model parameters $\theta$ after perturbing the training loss $\mathcal{L}(\hat{D}; \theta)$. We first give the formulation of an ordinary IF for a training data $z^{(i)} = (x^{(i)}, y^{(i)})$ in the context of PWS, which is similar to Eq. 4:

$$\phi_i(z) = -\nabla_\theta \ell(y^{(i)}, f_\theta(x^{(i)}))^\top H_\theta^{-1} \nabla_\theta \ell(y^{(i)}, f_\theta(x^{(i)})).$$  \hspace{1cm} (9)

Here, $H_\theta^{-1} = \frac{1}{N} \sum_{i=1}^N \nabla_\theta^2 \ell(y^{(i)}, f_\theta(x^{(i)}))$ and $\theta^*$ is the minimizer of the training loss before the perturbation. We reuse the notation of $H_\theta^*$ and $\theta^*$ in this section as their meanings are unchanged.

**Reweighting.** To study the influence of removing the $(i, j, c)$-effect, we reweight $\hat{L}_{i,j,c}(\theta)$ such that

$$\hat{L}_{\hat{i},j,c}(\hat{D}; \theta) = \frac{1}{N} \sum_{i'=1}^N \sum_{j'=1}^M \sum_{c'=1}^C \hat{L}_{i',j',c'} + \epsilon_{i,j,c} \hat{L}_{i,j,c}(\theta).$$  \hspace{1cm} (10)

Then, the influence score, i.e., the difference of loss on $z' = (x', y')$ caused by reweighting the loss term $\hat{L}_{i,j,c}(\theta)$, is,

$$\phi_i(z') = -\nabla_\theta \ell(y', f_\theta(x'))^\top H_\theta^{-1} \nabla_\theta \hat{L}_{i,j,c}(\theta^*).$$  \hspace{1cm} (11)

In contrast to ordinary IF that reweights the loss w.r.t. individual training data, the proposed one is finer-grained since it reweights the decomposed loss terms. By reweighting, the resultant probabilistic training label is no longer sum-to-one and the solution (Eq. 11) is only for identity $\sigma(\cdot)$ function.

**Weight-moving.** We also explore an alternative way of perturbing the training loss called weight-moving. Different from reweighting, its computation is agnostic to the $\sigma(\cdot)$ function and the probabilistic training label is still sum-to-one. Specifically, we remove the $(i, j, c)$-effect in the probabilistic training label $\hat{y}^{(i)}$ and renormalize it, leading to a new probabilistic training label:

$$\forall c \in [C], \hspace{1cm} \hat{y}^{(i)}_{-j, c'} = \frac{\sigma(\sum_{j=1}^M \mathbb{1}\{c \neq c' \land j \neq j'\} \cdot W_{j', L_{i,c}})}{\sum_{j=1}^M \sigma(\sum_{j=1}^M \mathbb{1}\{k \neq c' \land j \neq j'\} \cdot W_{j, L_{i,j}})}.$$  \hspace{1cm} (12)

Then, we perturb the training loss by moving the weight of the original loss on the $i$-th data $\hat{L}(\hat{y}^{(i)}, f_\theta(x^{(i)}))$ to a new loss term $\hat{L}_{\hat{i},j,c}(\hat{D}; \theta)$:

$$\hat{L}_{\hat{w}, i,j,c}(\hat{D}; \theta) = \frac{1}{N} \sum_{i'=1}^N \hat{L}(\hat{y}^{(i)}_{-j, c'}, f_\theta(x^{(i)})) + \epsilon_{i,j,c} \cdot \hat{L}(\hat{y}^{(i)}, f_\theta(x^{(i)})) - \epsilon_{i,j,c} \cdot \hat{L}(\hat{y}^{(i)}_{-j, c'}, f_\theta(x^{(i)})).$$  \hspace{1cm} (13)

When $\epsilon_{i,j,c} = -\frac{1}{N}$, the loss term $\hat{L}(\hat{y}^{(i)}, f_\theta(x^{(i)}))$ is indeed replaced by the new loss $\hat{L}(\hat{y}^{(i)}_{-j, c'}, f_\theta(x^{(i)}))$. We again provide the formulation of corresponding influence score:

$$\phi_{i,j,c}(z') = -\nabla_\theta \ell(y', f_\theta(x'))^\top H_\theta^{-1} \nabla_\theta \hat{L}(\hat{y}^{(i)}_{-j, c'}, f_\theta(x^{(i)})).$$  \hspace{1cm} (14)

**Discussion.** Note that the change of loss over a holodout set $D_s$ can be similarly computed and we denote it as $\phi_{i,j,c}(D_s)$ and $\phi_{i,j,c}(D_s)$. We omit the input variable and the superscript and use $\hat{\phi}_{i,j,c}$ to represent the proposed source-aware IF in general. The $\hat{\phi}_{i,j,c}$ corresponds to the influence of a specific perturbation on the training loss: setting the label model parameter $W_{j, L_{i,j}}$ to zero for the $i$-th data so that the $(i, j, c)$-effect is removed from training. Then, the difference between reweighting and weight-moving reduces to whether renormalization on the probabilistic labels is performed after the perturbation.

**Extension to RelatIF** [3]. We additionally demonstrate how to compute the RelatIF [3] in our source-aware IF setting. Specifically, the self-influence is $\phi_{i,j,c}^{rel} = -\nabla_\theta \hat{L}(\hat{y}^{(i)}, f_\theta^*(x^{(i)}))^\top H_\theta^{-1} \nabla_\theta \hat{L}(\hat{y}^{(i)}, f_\theta^*(x^{(i)}))$. Then, the source-aware RelatIF for both $\hat{\phi}_{i,j,c}$ and $\hat{\phi}_{i,j,c}$ are $\sqrt{\phi_{i,j,c}^{rel}}$ and $\sqrt{\phi_{i,j,c}^{rel}}$, respectively.

**3.3 Use cases**

**Case 1: estimating influence of different components.** One major benefit of our training loss decomposition and the identity $\sigma(\cdot)$ function is that we could readily reuse the source-aware IF $\hat{\phi}_{i,j,c}$
We study the effect of replacing original label model with the approximated one on the end model. We list the influence of different components of a PWS pipeline as well as how to calculate them with negative impact. Specifically, we have the following theorem for reweighting (a similar result to Theorem 1):

\[ \text{Case 2: improving test loss.} \quad \text{Another application of IF is to improve the test loss by identifying and then discarding/downweighting the harmful training data \cite{24, 42, 38}. Similarly, the source-aware IF } \phi_{i,j,c} \text{ can be employed to improve the test loss by discarding/downweighting the loss term } \ell_{i,j,c}(\theta) \text{ with negative impact. Specifically, we have the following theorem for reweighting (a similar result for weight-moving can be found in Appendix E.2):} \]

\[ \text{Theorem 1. Discarding or downweighting the loss terms in } S_- = \{ \ell_{i,j,c}() | i \in [N], j \in [M], c \in [C], \phi_{i,j,c}^w(D_t) > 0 \} \text{ from training could lead to a model with lower loss over a holdout set } D_1:} \]

\[ \mathcal{L}(D_1; \theta_{S_-}^*) - \mathcal{L}(D_1; \theta^*) \approx -\frac{1}{N} \sum_{\ell_{i,j,c}() \in S_-} \phi_{i,j,c}^w(D_t) \leq 0 \]

where \( \theta_{S_-}^* \) is the optimal model parameters obtained after the perturbation.

In practice, we use the validation set \( D_c \) and \( S_-(\alpha) = \{ \ell_{i,j,c}() | i \in [N], j \in [M], c \in [C], \phi_{i,j,c}^w(D_v) > \alpha \} \) to tolerate the estimation error of IF where the hyperparameter \( \alpha \) controls the proportion of items to be perturbed \cite{24}. Note that this method of improving test loss relies on the training loss decomposition and \( \sigma() \) being the identity function.

### 3.4 Beyond the identity function

The aforementioned use cases rely on the additivity of influence score and the training loss decomposition, which are both consequences of \( \sigma() \) being the identity function. When \( \sigma() \) is the exponential function, we could approximate the label model \( g_W() \) by a new label model \( g_{W^*}() \) with parameter \( W^* \) of the same shape but using identity function instead. Obviously, we want the new label model \( g_{W^*}() \) to reproduce the same probabilistic labels as those generated by \( g_W() \). This motivates us to solve the following optimization problem for \( W^* \):

\[ W^* = \arg \min_{W} \sum_{i=1}^{N} \sum_{c=1}^{C} \left( \frac{\exp(\sum_{j=1}^{M} W_{j,L_{i,j,c}})}{\sum_{j=1}^{M} \exp(\sum_{j=1}^{M} W_{j,L_{i,j,c}})} - \frac{\sum_{j=1}^{M} W_{j,L_{i,j,c}}}{\sum_{j=1}^{M} \sum_{k=1}^{M} W_{j,L_{i,j,k}}} \right)^2. \]  

(15)

Then, we can replace the label model \( g_W() \) with \( g_{W^*}() \) everywhere. As a consequence, we are now able to unify various types of label models in a single framework and enjoy the convenience and advantages brought by the linearity of the identity function. Note that for the original label model \( g_W() \) with \( \sigma() \) being the exponential function, we can still derive the influence \( \phi_{i,j,c} \) (see appendix). We study the effect of replacing original label model with the approximated one on the end model performance in the experiment section.

<table>
<thead>
<tr>
<th>Component of PWS pipeline</th>
<th>Symbol</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x(i) ): the ( i )-th data</td>
<td>( \phi_{x(i)} )</td>
<td>( \sum_{j=1}^{M} \sum_{c=1}^{C} \phi_{i,j,c} )</td>
</tr>
<tr>
<td>( \Lambda(j) ): the ( j )-th LF</td>
<td>( \phi_{\Lambda(j)} )</td>
<td>( \sum_{i=1}^{N} \sum_{c=1}^{C} \phi_{i,j,c} )</td>
</tr>
<tr>
<td>( L_{i,j} ): the ( j )-th LF’s output on the ( i )-th data</td>
<td>( \phi_{L_{i,j}} )</td>
<td>( \sum_{i=1}^{N} \phi_{i,j,c} )</td>
</tr>
<tr>
<td>( W_{j,k,c} ): a individual parameter of label model indexed by ( (j,k,c) )</td>
<td>( \phi_{W_{j,k,c}} )</td>
<td>( \sum_{i=1}^{N} \phi_{i,j,c} { L_{ij} = k } )</td>
</tr>
</tbody>
</table>
4 Experiment

4.1 Setup

Datasets. We include the following four classification datasets in WRENCH [50], a collection of benchmarking datasets for PWS: Census, Youtube, Yelp, and IMDB. Note that the Census dataset is a tabular dataset while the others are textual datasets. We use the labeling functions (LFs) released by WRENCH [50]. We also include the following tabular datasets: Mushroom [12], Spambase [12], and PhishingWebsites (PW) [29], for which we follow the instruction in the WRENCH [50] codebase to generate LFs from a decision tree learned on the labeled data. Finally, we follow [28] to derive LFs for a multiclass image classification task using the DomainNet [30] dataset, which contains 345 classes of images in 6 different domains: real, sketch, quickdraw, painting, infograph and clipart. We construct a classification task for each domain with the 5 classes containing the largest number of instances. We then train classifiers using the selected classes within the remaining five domains as LFs. We name the 6 resultant datasets DN-real, DN-sketch, DN-quickdraw, DN-painting, DN-infograph, and DN-clipart for each domain, respectively. Following [50], we use a pre-trained BERT model [10] to extract features for textual data and use ResNet-18 [17] pre-trained on ImageNet for image data.

Model Training and Evaluation. For label model, we focus on three commonly-used choices: Majority Voting (MV), Dawid-Skene model (DS) [9], and Snorkel [31]. The $\sigma(\cdot)$ function of the latter two label models is the exponential function and we use their approximated variants (as mentioned in Section 3.4) for all experiments. Following [23], we use logistic regression for the end model and defer the experiments on using neural networks to the appendix. We train the end model initialized as zero by gradient descent for 10,000 epochs with learning rate being 0.001, and then calculate the influence score on validation set $D_v$ throughout the experiments. When using the test loss as evaluation metric, we use $D_v$ to do model selection and report the test loss of the select model.

For simplicity, we use IF (R-IF) to represent ordinary IF (RelatIF) baseline; RW (R-RW) and WM (R-WM) are the proposed source-aware IF (RelatIF) calculated by the reweighting and weight-moving method respectively. We also use the term SIF (R-SIF) to represent the source-aware IF (RelatIF) when we do not care whether it is calculated by reweighting or weight-moving. Additional experiments and details are in Appendix G.

4.2 Experimental results

Understand model behavior from multiple angles. First and foremost, source-aware IF can be used to understand the end model behavior from multiple angles, while ordinary IF can only identify influential training data. Here, we take the DN-real dataset as an example and use reweighting-based source-aware IF (RW) to diagnose misclassified test data. For a single misclassified test data of the end model, we present the most “responsible” (1) training data, (2) LF, and (3) labeling (the vote of a LF on a training data), where being the most “responsible” means having the highest influence score on that data and likely being the primary cause of the misclassification. We train the end model initialized as zero by gradient descent for 10,000 epochs with learning rate being 0.001, and then calculate the influence score on validation set $D_v$ throughout the experiments. When using the test loss as evaluation metric, we use $D_v$ to do model selection and report the test loss of the select model.

For simplicity, we use IF (R-IF) to represent ordinary IF (RelatIF) baseline; RW (R-RW) and WM (R-WM) are the proposed source-aware IF (RelatIF) calculated by the reweighting and weight-moving method respectively. We also use the term SIF (R-SIF) to represent the source-aware IF (RelatIF) when we do not care whether it is calculated by reweighting or weight-moving. Additional experiments and details are in Appendix G.

Identify mislabeling of LFs. One unique advantage of source-aware IF over ordinary IF is that it allows us to trace the influence of each LF’s labeling on a specific training data ($\phi_{L_{ij}} = \sum_{c=1}^{C} \phi_{i,j,c}$, the 3rd row in Table 1), which is helpful in understanding the expertise of each individual LF and thus provides clues for practitioners to debug LFs, while ordinary IF can only estimate the influence of training data as a whole. We use $\phi_{L_{ij}}$ as a scoring function of how likely the $j$-th LF mislabels
Table 2: The most "responsible" data, LF, and labeling, identified by source-aware IF, for the misclassification of end models (trained with difference label models) on the same test image. We can see that source-aware IF can reveal the cause of misclassification from multiple angles.

<table>
<thead>
<tr>
<th>Label Model</th>
<th>Misclassified Test Data</th>
<th>Resp. Training Data</th>
<th>Resp. LF</th>
<th>Resp. Labeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV</td>
<td>golf club (0.55)</td>
<td>dog</td>
<td>golf club (1.0)</td>
<td>quickdraw</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>quickdraw</td>
<td></td>
</tr>
<tr>
<td>DS</td>
<td>golf club (0.44)</td>
<td>dog</td>
<td>golf club (0.71)</td>
<td>painting</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>painting</td>
<td></td>
</tr>
<tr>
<td>Snorkel</td>
<td>golf club (0.66)</td>
<td>dog</td>
<td>golf club (0.94)</td>
<td>clipart</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>clipart</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Performance comparison results on identifying mislabeling of LFs. We report the average precision (AP) score averaged over LFs for each dataset. The larger the AP is, the better the method identifies mislabeling of LFs.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>KNN</th>
<th>LM</th>
<th>EM</th>
<th>R-W</th>
<th>R-WM</th>
<th>WM</th>
<th>R-WM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LM</td>
<td>EM</td>
<td>R-W</td>
<td>R-WM</td>
<td>WM</td>
<td>R-WM</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

-i-th data, i.e., \( L_{ij} \neq y^{(i)} \), to identify the mislabeling of individual LF. We thus formulate it as a binary classification task and report the average precision (AP), a standard evaluation metric of a scoring function. We use the discrepancy between j-th LF’s labeling on i-th data \( L_{ij} \) and the probabilistic label output by (1) label model \( LM \), (2) end model \( EM \), and (3) a K-nearest neighbor classifier \( K = 10 \) trained on validation set \( KNN \) as baseline scoring functions. Specifically, we calculated the discrepancy as one minus the dot product of \( L_{ij} \)’s one-hot representation and the probabilistic label; for example, if \( L_{ij} \)’s one-hot representation is \([0, 1, 0]^{\top}\) and the probabilistic label is \([0.5, 0.4, 0.3]^{\top}\), the discrepancy is \(1 - 0.5 \times 1 = 0.6\). Note that \( KNN \) is a more fair and competitive baseline because it directly leverages the validation set and, as our source-aware IF, does not introduce any parametrized model in addition to what PWS has. We present the results in Table 3; we can see the superiority of source-aware IF in this task since it outperformed baselines in most cases and the averaged gains are about 9%–37% for different label models. Intriguingly, when \( LM \) and \( EM \) render unsatisfactory performance, e.g., on \( DomainNet \), which indicates the quality of the end model is not good, the source-aware IF still performed well. This means it is less sensitive to the quality of PWS, which is critical for debugging the pipeline.

**Improve the test loss of the end model.** We compared the proposed source-aware IF as well as its RelatIF variant to ordinary IF/RelatIF in terms of the capability of improving the test loss. For source-aware IF/RelatIF, we discarded loss terms in \( S_-(\alpha) \) and then re-trained the end model (as described in Section 3.3), while for IF/RelatIF, we likewise discarded training data in \( D_-(\alpha) = \left\{ z^{(i)} \big| i \in [N], \phi_i(D_v) > \alpha \right\} \). For these methods, we tuned the hyperparameter \( \alpha \) on validation set for best validation loss. We additionally included the group influence (GIF) [23] that estimates the influence of each LF and is only applicable to \( M \) as in [23]; we discarded k LFs with highest negative impact and tune k on validation set. The results are in Table 4, where \( ERM \) (Empirical Risk Minimization) is the ordinary training without any perturbation. From the results, we conclude that although ordinary IF/RelatIF can already improve the test loss over ordinary training (ERM) by a large margin (>0.1 test loss on average), the source-aware IF/RelatIF can further boost the performance by a similar margin (>0.1 averaged test loss improvement over IF/RelatIF). This shows...
we report the Spearman’s ranking correlation coefficient (whether or not the approximation is well-behaved. To answer with their approximated variants (Section 3.4), it is natural to (Table 5: Spearman’s ranking correlation coefficient (data-level IF and that estimated from source-aware IF. All results pass the significance test (are from 0.71 to 0.99, which further demonstrates the efficacy of source-aware IF.

Estimate ordinary data-level IF via source-aware IF. As described in Section 3.3, we can use the source-aware IF calculated by the reweighting method to estimate the data-level influence score \( \phi_{ij} = \sum_{j=1}^{M} \sum_{c=1}^{C} \phi_{j,c} \), the 1st row in Table 1), which coincides with the influence computed by ordinary IF in theory. Thus, we are curious about how well the source-aware IF can be used to estimate the ordinary data-level IF. Because IF is known to be noisy in the value but good in their ranking [4, 23], we instead study the ranking correlation between \( \phi_{ij} \) and the ordinary IF. In Table 5 we report the Spearman’s ranking correlation coefficient (≤ 1), where larger coefficients indicate better estimation of the ordinary IF based on \( \phi_{ij} \) in terms of the ranking. From the results, we can see that the \( \phi_{ij} \) actually preserves the ranking of ordinary IF quite well as Spearman’s coefficients are from 0.71 to 0.99, which further demonstrates the efficacy of source-aware IF.

Table 5: Spearman’s ranking correlation coefficient (≤ 1) between influence calculated by ordinary data-level IF and that estimated from source-aware IF. All results pass the significance test (\( \rho < 0.05 \)).

Efficacy of label model approximation. Since we bypassed the non-linearity of exponential function in DS and Snorkel with their approximated variants (Section 3.4), it is natural to ask how well the approximated label model can perform and whether or not the approximation is well-haved. To answer these questions, in Figure 2 we show that although the averaged test loss of ordinary training (ERM) and best of IF/R-IF (Best IF/R-IF) are slightly better than their counterpart of approximated label model (Approx. ERM and Approx. Best IF/R-IF), the best averaged test loss obtained by SIF/R-SIF with the approximated label model (Approx. Best SIF/R-SIF) is much lower. Such an observation indicates that with the goal of improving test loss, it is beneficial to choose \( \phi(x) \) to be the identity function so that the training loss can be decomposed and perturbed based on source-aware IF.

Figure 2: Avg. test loss of the end model with/without label model approximation.
5 Related work

Programmatic Weak Supervision. In the Programmatic Weak Supervision (PWS) literature [33, 46], popular choices of labeling functions (LFs) include user-crafted heuristics [32, 39, 45], pretrained models [2, 28], external knowledge bases [18, 26], and crowd-sourced labels [9, 25]. Recently, researchers have developed various label models [33, 31, 14, 48, 44, 37, 47, 47, 49, 43]. In the PWS pipeline, the behavior of the end model is directly dictated by the soft labels, wherein the soft labels themselves are also intermediate products of the label model and the LFs’ votes. Thus, to interpret the trained end model and potentially debug the PWS workflow, it is critical to understand the influence of each PWS component on the end model’s performance, which has yet received adequate research attention.

Influence Function. Among various ways to understand a model’s behavior through the training data points, Influence Function (IF) is an effective technique with a wide range of applications [22, 23, 16, 7]. In PWS, although one can directly apply IF to understand a model’s behavior through the weakly labeled training data, more insights may be unveiled by further investigations of how each PWS component, e.g., LFs and label models, influences the end model. In similar spirit to us, [6] developed multi-stage IF to trace a fine-tuned model’s behavior back to the pretraining data. Perhaps most similarly to our work, [23] conducted initial exploration on using group influence to study how removing an LF from the PWS pipeline would affect the end model’s performance. However, the group influence method is only applicable to a specific simplified setting of PWS.

6 Limitation and social impacts

Limitations. Although we study several popular label models and show how to use approximated label model in case that users need to use a label model that is not included in this work, user may still encounter difficulties in real applications when using complicated label model. Also, this work focuses on the two-stage PWS pipeline, while very recently there are some one-stage methods, to which our framework may not be applicable. In the main body of the paper, we use logistic regression as end model in our experiments, while in appendix we present experimental results of a simple two-layer neural network. But we do not include experiments on complicated deep learning models like Convolutional Neural Network.

Social impacts. This work aims at help people understanding the Programmatic Weak Supervision and has the potential to resolve the bias in the generation process of training label, which might have positive social impact. We do not foresee any form of negative social impact induced by our work.

7 Conclusion

In this work, by leveraging the knowledge of how probabilistic training labels are aggregated from different sources in the PWS pipeline, we have proposed source-aware IF that enables users to estimate the influence of different components (e.g., source vote, training data, LFs, etc.) in the pipeline on the end model’s behavior, and unlocked various practical use cases. Specifically, we have demonstrated how the proposed source-aware IF can be used to (1) unveil the most influential training data point, LF, or a specific vote to a misclassification made by a model, (2) identify mislabeling of LFs on certain data points, and (3) improve the end model’s test performance.

8 Acknowledgement

Thanks the anonymous reviewers for their helpful comments and suggestions. We also thank Haojie Jia for his help of experiments.

References


Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See appendix.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See appendix.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See appendix.
   (b) Did you include complete proofs of all theoretical results? [Yes] See appendix.
3. If you ran experiments...

13
(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplemental materials.

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

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] For experiments with randomness, we report averaged results over 5 runs.

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

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

(a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [Yes]

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes]

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

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

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

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

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

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