Automatic label error detection on text datasets labeled with data programming

Nalbert G. M. Leal[1], **Daniel S. A. Araújo**[1], **Elias J. Menezes Neto**[1]

[1]Universidade Federal do Rio Grande do Norte (UFRN) – Natal – RN – Brazil

nalbertgml@gmail.com, daniel@imd.ufrn.br, elias.jacob@ufrn.br

Abstract. *Supervised machine learning relies on the availability of large volumes of accurately annotated data, a requirement that data programming (DP) alleviates by aggregating weak supervision sources into probabilistic labels. However, DP generated labels remain susceptible to noise, compromising downstream model performance. In this work, we integrate and evaluate four Automatic Error Detection (AED) techniques: Retag, Confident Learning, Source-aware Influence Functions, and Unsupervised Label Functions Correction (ULF); within standard two-stage DP pipelines. Using the WRENCH benchmark on two text-classification tasks (YouTube and SMS spam detection), we optimize each pipeline via Bayesian optimization and collect the metrics Matthews correlation coefficient, accuracy, F1 score, and computational cost. Our experiments show that Influence Functions combined with the Hyper Label Model achieve the most positive trade-off between accuracy improvement and runtime on balanced data, while simpler DP baselines outperform all analyzed AED methods under class imbalance. These findings underscore both the promise and practical limitations of AED in refining weakly supervised workflows, guiding future development of cost-effective label-noise mitigation strategies.*

1. Introduction

In recent years, manual data labeling has played a central role in enabling the performance of supervised machine learning models. As model complexity increased, so did the demand for large volumes of labeled data, making manual annotation prohibitively expensive and time-consuming for many applications [Zhang et al. 2022a]. This limitation favored the development of alternative labeling strategies aimed at reducing annotation costs.

Among these, data programming (DP) [Ratner et al. 2017] stands out by enabling automatic data annotation with minimal human input. However, this automation introduces label noise, as each labeling source—such as heuristics, external knowledge bases, or pre-trained models—may introduce different types of errors. To address these inconsistencies from multiple sources, DP pipelines typically employ a label model to aggregate the noisy outputs and infer the most likely ground truth labels [Fu et al. 2020].

Since the final label quality is determined by the label model's ability to reconcile disagreement among noisy sources, mislabeled data can still persist in the final dataset, potentially degrading the performance of downstream models. Identifying and correcting such label errors is therefore a crucial step toward improving the model's performance [George et al. 2024].

This work investigates recent methods for automatic error detection (AED) in datasets labeled through DP. We also present preliminary experiments evaluating selected AED techniques within the standard DP pipeline. For this evaluation, we utilize the WRENCH benchmark, focusing on text classification datasets due to their prominence in the benchmark and the growing relevance of textual data with the rise of large language models.

The remainder of this paper is organized as follows. In section 2, we define the DP pipeline, its components, the challenge of noisy labels, and the label models used to aggregate them. Section 3 characterizes noisy labels and presents four mislabel-detection methods applied in our experiments. Section 4 details the experimental setup, datasets, label models, hyperparameter tuning, results, and the experiment limitations. In section 5, we briefly discuss works with similar aims, and we conclude in section 6.

We present the following contributions in this work:

1. We conduct an experiment applying selected automatic error detection (AED) methods to assess their potential in enhancing the quality of models generated through the Data Programming (DP) process;
2. We compare DP-specific and general-purpose AED methods, conducting both a performance evaluation and a cost-benefit analysis to identify the most effective approaches for integration in the DP pipeline;
3. We demonstrate that employing AED methods in conjunction with DP can effectively improve the quality of generated labels without significantly increasing dataset annotation costs.

2. Data Programming

The development of more complex machine learning models demands large amounts of labeled examples. One of the challenges faced in new model development is to collect the massive amount of hand-labeled data required. Hand-labeling is a costly and time-consuming task that requires multiple data experts to annotate dataset objects. It can be prohibitive to projects and research that can't handle the costs or can't have multiple experts for data annotation [Zhang et al. 2022a]. Weakly supervised learning (WSL) is a term created to describe any technique that reduces the costs associated with the labeling process [Zhou 2017]. The ideal WSL technique should be scalable and adapt to the problem faced by the model developers. Among the available strategies, data programming emerges as a promising method to label large amounts of data with minimum manual intervention.

Data programming enables rapid dataset annotation through programmatically defined labeling functions (LFs), reducing the reliance on multiple experts by employing automated labeling sources such as knowledge bases, heuristics, feature annotations, and machine learning models [Ratner et al. 2018]. Each LF is modeled as a function $\lambda_i : X \longrightarrow Y \cup \{-1\}$, where X represents the dataset instances and $Y \cup \{-1\}$ is the union of the label set with the abstention label (-1) [Ratner et al. 2017]. The outputs of the LFs are then aggregated by a generative model, referred to as the label model (LM), which consolidates the label agreements and disagreements to produce the final dataset labels used for model training. The model is referred to as the end-model in the literature.

2.1. Label Models

One of the key challenges in data programming is combining the labels generated by different LFs to infer the ground-truth labels for the dataset [Wu et al. 2023]. A common solution to this problem is the use of label models, which efficiently aggregate multiple label sources by estimating the accuracy and correlations among the LFs applied to the dataset [Fu et al. 2020]. The input to a label model is a label matrix L of size $n \times m$, where n represents the number of data points $x \in X$ and m denotes the number of LFs that labeled a subset of X. Each column $L[:, j]$ corresponds to the labels assigned by the j^{th} LF, while each row $L[i, :]$ contains the weak labels associated with the i^{th} data point. Using this matrix, the label model can infer the ground-truth label y_i for each data point x_i [Wu et al. 2023].

Combining labeling functions (LFs) in data programming (DP) can be approached through various methods. One standard approach involves latent variable probabilistic graphical models, aggregating the sources based on agreements and disagreements [Fu et al. 2020]. In DP, label models (LMs) typically assume different distributions to combine LFs. For instance, a common assumption is a distribution $P(y_i|L[i, :], \Theta)$, where Θ represents the parameter vector and y_i the true label of the i^{th} data point [Wu et al. 2023].

In [Ratner et al. 2017], each LF is treated as independent, addressing a binary labeling task where $y \in \{0, 1\}$. The proposed model defines β_j as the probability of the j^{th} LF providing a non-abstention label and α_j as the probability of the label being correct. The joint distribution of the LFs $\Lambda = (\lambda_1, \ldots, \lambda_m)$ is modeled using these probabilities, with the objective of the LM being to learn the α and β parameters.

The Dawid-Skene technique, initially proposed to estimate clinicians' error rates in symptom classification [Dawid and Skene 1979], can be adapted to DP. It estimates two latent variables: the probability p of a clinician correctly annotating a sample and the error rate $\pi_{jl}^{(k)}$, where l represents the identified symptom and j the true symptom. The estimation is performed via the Expectation-Maximization (EM) algorithm, with the E-step estimating the true class and the M-step updating p and $\pi_{jl}^{(k)}$. Although outdated for real-world DP applications, Dawid-Skene serves as a baseline for historical comparisons.

The Flying Squid method, introduced in [Fu et al. 2020], addresses the challenge of efficiently estimating LF accuracies and correlations. Instead of multiple iterations, it reduces parameter estimation to solving a system of equations with closed-form solutions, enabling linear-time estimation. The approach is based on the "triple method," utilizing the label matrix L, the dependency graph G_{dep}, and the prior distribution $P(\bar{y})$.

The Hyper Label Model [Wu et al. 2023] minimizes assumptions about the underlying distributions, only assuming that labels from matrix L are better than random assignments. Instead of defining a prior distribution, it estimates the parameters Θ during the learning phase. The matrix L is represented as a graph, allowing a graph neural network (GNN) to estimate Θ more efficiently, reducing the algorithm's exponential complexity to $O(nm)$. Each label $L[i, j]$ becomes a node connected to other nodes based on LF and label similarities, facilitating consistent inferences regardless of the order of LFs or instances. True labels can be incorporated to fine-tune the GNN.

3. Label Error Detection

Due to the use of multiple label sources in the DP pipeline, the generated labels can be susceptible to noise, potentially leading to the mislabeling of ground truth data. These mislabels can degrade the performance of models trained on such data. This section outlines the various types of noise that can impact labeling, defines what constitutes a mislabeled instance, and presents the techniques employed in the experiment to assess the effectiveness of these methods in detecting and mitigating annotation errors in DP-generated labels.

3.1. Noise Labels Definition

According to [Zhu et al. 2021], a clean dataset D can be defined as $D := (x_i, \hat{y}_i)_{i \in [N]}$, where $[N] := \{1, 2, 3, ..., N\}$. The tuple $(x_i, \hat{y}_i)$, $x_i \in X$ represents labeled data, and $\hat{y}_i \in Y$ is a clean label (the ground truth label) from the set Y. A noise dataset (a set with mislabeled data) is defined as $\tilde{D} := (x_i, \tilde{y}_i)_{i \in [N]}$, where $(x_i, \tilde{y}_i)$ is a data with a label $\tilde{y}_i$ that can be different of the true label, specifically, $\tilde{y}_i \neq \hat{y}_i$.

3.2. Detection Methods

Labeled datasets are crucial for training effective machine learning models. In the presence of noisy labels, annotation error detection (AED) methods become essential for improving dataset quality and model performance. AED techniques vary in complexity; some are simple and quick to implement, while others are more advanced, requiring greater computational resources and domain knowledge. Different AED methods have been developed to either complement weak supervised learning (WSL) techniques or to independently enhance the labeling process.

Retag, proposed in [van Halteren 2000], is a model-based AED technique that retrains a model on a subset of the data and evaluates it on the remaining data to detect labeling inconsistencies. The method uses cross-validation to maintain calibration and systematically compares predicted labels with the originally annotated labels. Any discrepancy is considered a potential mislabeling. Due to its simplicity, Retag has been effectively applied in real-world scenarios where rapid error detection is needed.

Confident Learning is a data-centric AED approach designed to identify mislabeled instances in noisy datasets [Northcutt et al. 2022]. It assumes class-conditional noise and estimates the joint distribution of the observed noisy labels $\tilde{y}$ and the true labels $\hat{y}$. Confident Learning leverages out-of-sample predicted probabilities $\hat{P}k, i$ obtained through cross-validation and a noise transition matrix $Q\tilde{y}, \hat{y}$ to detect inconsistencies.

The technique uses self-confidence and high sparsity as key heuristics. Self-confidence refers to the confidence of the model in assigning a label $\tilde{y}$ to a sample x, while high sparsity indicates the concentration of errors in specific label pairs. For instance, misclassifying a dog as a wolf is more common than confusing a dog with a chair, reflecting realistic error patterns in datasets.

Traditional Influence Functions quantify the impact of adding or removing individual training samples on model performance [George et al. 2024]. In the context of DP pipelines, Source-aware Influence Functions (IF) extend this concept to assess the impact of specific tuples $(data, LF, class)$ on the final labels [Zhang et al. 2022b].

IF calculates influence scores by analyzing the perturbation caused by each LF or data point, using two main approaches: reweighting and weight-moving. This allows for detailed debugging, enabling the identification of unreliable LFs that contribute to label noise. Consequently, the method provides insights into which LFs are beneficial or detrimental, facilitating targeted corrections.

Unsupervised Label Functions Correction (ULF) is a DP-specific AED method introduced by [Sedova and Roth 2023]. It employs k-fold cross-validation to refine LFs, iteratively training a model on a subset of LFs and using the remaining LFs to predict and validate the model's performance.

In each iteration, discrepancies between model predictions and held-out LF outputs indicate potential noise in the LFs. This iterative adjustment reduces noise, prioritizing more reliable LFs and enhancing the robustness of the DP pipeline, particularly in scenarios with variable LF quality.

4. Experiment

The experiment aims to evaluate the effectiveness of AED methods in enhancing the performance of models trained with programmatically generated labels within a data programming (DP) pipeline. Specifically, it examines whether the application of AED methods can improve the end-model's performance and assesses the computational cost associated with their use. Since the final objective of every DP pipeline is to train a model with the programmatically generated labels, the DP pipeline's evaluation relies on metrics obtained from end-models trained on text classification tasks, enabling a direct assessment of AED impact on DP pipeline outcomes.

To ensure standardization and reproducibility, the experiment utilizes the Wrench benchmark [Zhang et al. 2021], a comprehensive platform designed to support consistent evaluation of DP techniques. Wrench includes 22 datasets spanning various domains (e.g., text, tabular, image, biomedical), each equipped with ground truth labels, predefined label functions (LFs), and fixed train/validation/test splits. This setup facilitates systematic experimentation and controlled analysis of LF and label model (LM) behaviors.

Wrench supports two types of DP pipelines: (1) two-stage pipelines, where LFs annotate data and LMs aggregate the outputs to generate labels for training an end model, and (2) one-stage (joint) models that train the LM and the end-model simultaneously. The benchmark standardizes all components—datasets, LFs, LMs, and end-models through a unified interface, enabling consistent application of AED methods and fair comparison across configurations.

In this study, the two-stage pipeline is adopted. AED methods are applied between the LF aggregation with the LM and the end-model training stages. This setup allows for assessing improvements in model performance and measuring any increase in computational time due to AED integration, offering insights into their value and scalability within DP workflows. With the adoption of the Wrench framework, each analyzed pipeline is structured as a tuple comprising four components: dataset, LM, AED method, and end-model.

4.1. Datasets

The selected datasets from Wrench are YouTube [Alberto and Lochter 2015] and SMS [Almeida and Hidalgo 2011], both focused on text classification and include ground-truth labels, LF-generated matrices L, and raw data points. Text classification was chosen due to its broad support in Wrench and its prominence in machine learning research.

YouTube contains public comments for spam classification, with LFs defined in [Team 2019], while SMS also targets spam detection using LFs from [Awasthi et al. 2020]. Table 1 shows class distributions, notably the imbalance in the SMS dataset, where label 0 appears six times more often than label 1 in the training split. This imbalance is expected to affect AED performance.

Table 1. Examples division in the YouTube and SMS datasets

	YouTube			**SMS**		
	Train	Validation	Test	Train	Validation	Test
Number of examples on class 0	801	64	132	3954	436	433
Number of examples on class 1	885	56	118	617	64	67
Total	1686	120	250	4571	500	500

For feature extraction, a pre-trained BERT model provided by Wrench was used, enabling simplified downstream modeling through high-quality embeddings. Consequently, an MLP with three layers (input, hidden, output) was selected as the end model. Wrench treats the model size as a tunable hyperparameter, optimized for best performance.

4.2. Parameters optimization

Bayesian optimization (BO) was employed to enhance model performance in DP pipelines by efficiently tuning hyperparameters. The BO implementation [Nogueira 14] commenced with ten iterations using randomly sampled parameter values to establish a baseline, followed by 20 optimization iterations. Each iteration utilized three-fold cross-validation on the training split, with the mean metric across folds indicating parameter quality. Given that the experiment evaluates DP pipelines as combinations of *(dataset, LM, AED method, end-model)*, BO was re-executed for each component change to account for potential shifts in behavior.

4.3. Label Models Selection

To combine multiple LF data annotations, DP employs a Label Model (LM). Dawid-Skene and Flying Squid were selected due to their prevalence in previous DP experiments, with Dawid-Skene also holding historical significance as a technique developed in the 1970s for crowdsourcing, allowing for the analysis of older label aggregation methods. The Hyper label Model was chosen as a state-of-the-art LM, while Majority Vote, the simplest LM technique, determines the final label by selecting the most frequently annotated label for each data point.

4.4. Pipeline Execution

After 30 iterations of BO, the optimal hyperparameter values were identified and subsequently used to execute the pipeline 15 times. In each run, the metrics Matthews correlation coefficient (MCC), accuracy (ACC), F1 score (F1), and execution time were recorded. Although ACC is a widely used metric in machine learning, it can be misleading in the context of imbalanced datasets, as it may remain high while neglecting the minority class. This issue can arise either due to how the LM combines labels or the inherent nature of the dataset. Therefore, MCC and F1 were selected to provide more robust performance evaluation under such conditions [Geron 2019] [Chicco and Jurman 2020]. Execution time was also measured to assess the computational cost of the pipeline. These metrics were used to compare and analyze the pipelines. Figure 1 presents a diagram of the pipeline experiment. To run the experiment, we used a computer with Linux Ubuntu 24.04, CPU Ryzen 7 7700, GPU RTX 4060 8GB, and 32 GB of RAM DDR5 5600 MHz.

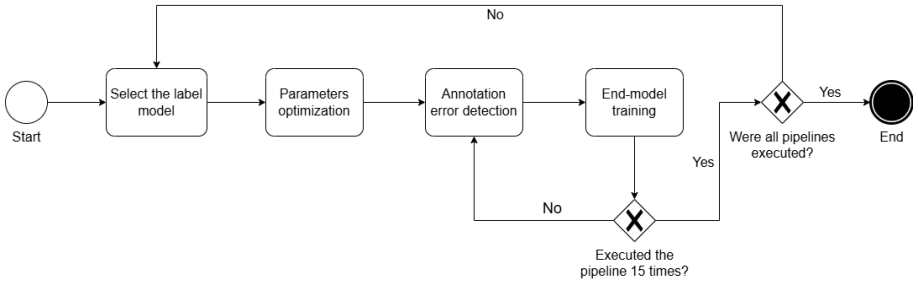

Figure 1. Experiment BPMN diagram

4.5. Results

Based on the results, the investigation aims to evaluate the following hypotheses:

1. Applying AED methods to labels generated by the DP pipeline enhances the performance metrics obtained from the end model in the DP pipeline.
2. The runtime trade-off associated with incorporating AED techniques in the DP pipeline is not prohibitive.

The primary objective of the experiment was to verify hypothesis 1. Hypothesis 2 assesses the practicality of hypothesis 1 in real-world scenarios, determining whether implementing AED techniques in DP pipelines is viable. Specifically, even if hypothesis 1 holds true, excessive computational costs associated with AED (in this case, increased runtime) could render its application in DP pipelines impractical, thereby justifying its avoidance.

The experiment results were separated into tables by the dataset where the metrics were collected. Each line of the tables contains the metrics obtained from the Wrench MLP end-model, trained with a pipeline formed by the dataset, LM, and AED method. The tables 2 and 3 show the metrics collected in the experiment. The column metrics with the lowest values are presented with an underline, the best column metric is in bold, and if a pipeline metric is marked with the italic word ”Error”, it indicates that for some reason the pipeline couldn’t finish. If multiple best (or lowest) values are found (pipelines with the same metric mean), then all of them share the same mark (bold or underline). Under

Table 2. Table with metrics means for 15 executions on YouTube *dataset*

Technique	Label Model	MCC	Acc	F1	Time (seconds)
No AED (baselines)	David-Skene	0.742 (0.731, 0.753)	0.868 (0.862, 0.873)	0.861 (0.854, 0.868)	5.66 (4.94, 6.33)
	Flying Squid	0.762 (0.754, 0.771)	0.881 (0.877, 0.886)	0.873 (0.869, 0.878)	5.03 (4.26, 5.79)
	Hyper Label Model	0.788 (0.778, 0.799)	0.893 (0.888, 0.899)	0.886 (0.880, 0.893)	5.16 (4.26, 6.29)
	Majority Vote	0.783 (0.766, 0.800)	0.890 (0.882, 0.899)	0.881 (0.871, 0.891)	**2.17** (1.78, 2.60)
Confident Learning	David-Skene	0.708 (0.690, 0.724)	0.853 (0.844, 0.861)	0.836 (0.824, 0.848)	12.40 (11.75, 13.02)
	Flying Squid	0.752 (0.743, 0.760)	0.874 (0.870, 0.879)	0.866 (0.861, 0.871)	40.16 (37.42, 43.05)
	Hyper Label Model	0.774 (0.766, 0.783)	0.886 (0.882, 0.891)	0.882 (0.879, 0.887)	40.58 (38.09, 42.91)
	Majority Vote	0.726 (0.711, 0.742)	0.862 (0.855, 0.870)	0.853 (0.845, 0.863)	11.47 (10.99, 11.98)
Influence Functions	David-Skene	0.743 (0.733, 0.753)	0.870 (0.866, 0.876)	0.862 (0.858, 0.868)	17.68 (17.35, 18.02)
	Flying Squid	0.782 (0.729, 0.818)	0.891 (0.863, 0.909)	0.886 (0.860, 0.903)	19.86 (19.17, 20.62)
	Hyper Label Model	0.804 (0.792, 0.816)	0.901 (0.896, 0.907)	0.894 (0.887, 0.901)	19.38 (18.99, 19.80)
	Majority Vote	**0.829** (0.823, 0.836)	**0.914** (0.911, 0.918)	**0.911** (0.908, 0.915)	297.90 (295.14, 300.89)
Retag	David-Skene	0.761 (0.751, 0.771)	0.880 (0.875, 0.885)	0.870 (0.864, 0.877)	65.89 (63.32, 68.38)
	Flying Squid	0.736 (0.725, 0.747)	0.867 (0.861, 0.873)	0.859 (0.854, 0.866)	24.60 (22.30, 26.87)
	Hyper Label Model	0.764 (0.757, 0.772)	0.882 (0.878, 0.886)	0.874 (0.869, 0.879)	21.21 (19.67, 22.61)
	Majority Vote	0.731 (0.720, 0.744)	0.865 (0.859, 0.871)	0.853 (0.844, 0.863)	8.29 (7.54, 9.05)
ULF	ULF LM	0.754 (0.740, 0.770)	0.876 (0.869, 0.885)	0.870 (0.862, 0.879)	212.74 (207.03, 218.50)

each mean is given the confidence interval (with confidence level of 95%) to evaluate the reliability of the results.

The results in table 2 indicate that AED methods can positively affect the performance of DP pipelines without an expensive increase in the computational cost. On the baselines, the Hyper Label Model presents the best MCC, accuracy, and F1 score. While the Majority Vote achieved better runtime and slightly lower metrics. Among the AED methods, Influence Functions consistently improved all the metrics relative to their corresponding baselines. When paired with the Hyper Label Model, that technique increases the MCC from 0.788 to 0.804, at the cost of 19.38, while the cost on the baseline was 5.16 ($\approx$4$\times$ the baseline). However, the combination of the pipeline with the Majority Vote produced the highest MCC (0.829), but with the runtime of nearly five minutes, it makes this combination impractical in real scenarios.

Conversely, Confident Learning degraded all metrics for every LM tested, while also imposing substantial overhead on the pipelines. Retag provided a small MCC uplift only on the pipeline with Dawid-Skene, but at the cost of increasing the runtime from 5.66 to 65.89 seconds, and degrading the metrics on all other configurations. The ULF method, although achieving intermediate predictive gains, required over 200 seconds of processing, offering an unfavorable trade-off compared to simpler baselines. The results suggest that these label–correction heuristics may be ill-suited to the particular noise char-

acteristics of this dataset, or that they're inappropriate to improve the performance of DP pipelines.

Considering the aforementioned points, these findings indicate that Influence Functions paired with the Hyper Label Model emerge as a promising AED method, representing the most advantageous cost-effectiveness between metrics improvement and execution time in our experimental setting. In contrast, where computational efficiency is essential, the Hyper Label Model without AED still yields competitive performance with a small runtime.

Table 3. Table with metrics means for 15 executions on SMS *dataset*

Technique	Label Model	MCC	Acc	F1	Time (seconds)
No AED (baselines)	David-Skene	0.877 (0.871, 0.884)	0.971 (0.970, 0.973)	0.893 (0.888, 0.900)	6.12 (5.84, 6.39)
	Flying Squid	0.402 (0.336, 0.468)	0.888 (0.883, 0.894)	0.362 (0.270, 0.459)	8.56 (7.61, 9.54)
	Hyper Label Model	**0.896** (0.886, 0.905)	**0.976** (0.974, 0.978)	**0.908** (0.899, 0.917)	5.79 (5.64, 5.94)
	Majority Vote	0.886 (0.878, 0.894)	0.974 (0.973, 0.976)	0.899 (0.891, 0.907)	**2.58** (2.33, 2.86)
Confident Learning	David-Skene	0.249 (0.134, 0.367)	0.880 (0.873, 0.889)	0.251 (0.127, 0.383)	18.88 (17.96, 19.79)
	Flying Squid	0.000 (0.000, 0.000)	0.865 (0.866, 0.866)	0.000 (0.000, 0.000)	11.63 (11.58, 11.67)
	Hyper Label Model	0.797 (0.775, 0.820)	0.954 (0.949, 0.959)	0.818 (0.797, 0.840)	8.74 (8.48, 9.04)
	Majority Vote	0.803 (0.782, 0.827)	0.956 (0.952, 0.961)	0.824 (0.803, 0.846)	8.61 (8.30, 9.00)
Influence Functions	David-Skene	0.878 (0.872, 0.885)	0.972 (0.971, 0.974)	0.892 (0.887, 0.899)	21.48 (21.17, 21.78)
	Flying Squid	0.164 (0.075, 0.256)	0.858 (0.854, 0.864)	0.203 (0.091, 0.316)	38.93 (38.47, 39.35)
	Hyper Label Model	0.879 (0.872, 0.887)	0.971 (0.970, 0.974)	0.894 (0.888, 0.902)	448.14 (446.07, 451.70)
	Majority Vote	*Error* ( )	*Error* ( )	*Error* ( )	*Error* ( )
Retag	David-Skene	0.000 (0.000, 0.000)	0.865 (0.866, 0.866)	0.000 (0.000, 0.000)	7.96 (7.90, 8.03)
	Flying Squid	0.000 (0.000, 0.000)	0.865 (0.866, 0.866)	0.000 (0.000, 0.000)	6.89 (6.82, 6.95)
	Hyper Label Model	0.000 (0.000, 0.000)	0.865 (0.866, 0.866)	0.000 (0.000, 0.000)	5.57 (5.54, 5.59)
	Majority Vote	0.000 (0.000, 0.000)	0.865 (0.866, 0.866)	0.000 (0.000, 0.000)	6.24 (6.20, 6.27)
ULF	ULF LM	0.843 (0.827, 0.858)	0.965 (0.962, 0.968)	0.854 (0.837, 0.869)	129.71 (121.94, 138.40)

On Table 3, the results for the SMS dataset reveal that, in unbalanced situations, the absence of AED (only label aggregation by LM strategies) achieves high performance on all metrics. Particularly, the Hyper Label Model presented the highest MCC, accuracy, and F1 score, while requiring, on average, 5.79 seconds of runtime. Similar to the baselines on Table 2, Majority Vote offers close performance with a lower runtime, suggesting that more complex and sophisticated LMs don't necessarily guarantee better performance.

Among the AED methods, none of them provided a significant improvement over the baseline pipelines. Confident Learning, for example, degrades performance across all models, while execution time increased by up to 250%. Influence Functions maintained similar baseline metrics, but with prohibitive runtime (with an average of 448 seconds on the Hyper Label Model). Also, Influence Functions could not complete the pipeline exe-

cution with the Majority Vote. The severe class imbalance led the algorithm to misidentify minority class samples as outliers, causing the algorithm execution to loop indefinitely.

Retag failed in all executions to achieve MCC or F1 score above zero. Also, besides the baseline pipelines, only train the end-model and the Retag pipelines, train at least two models (k-fold cross-validation and the end-model training); the Retag runtime was lower compared to the relative baselines. It indicates that the end-model on Retag didn't train as long as the baselines, stopping the training for no improvement, indicating a possible label degradation.

In conclusion, for the SMS dataset and under the configurations evaluated, the simplest data-programming pipelines—specifically those employing the Hyper Label Model or Majority Vote without any AED—deliver superior label-quality metrics and runtime efficiency. None of the AED methods tested here surpassed these baselines; rather, they introduced either performance deterioration or excessive overhead. Consequently, practitioners should exercise caution when integrating these AED techniques into data-programming workflows for similar text-classification tasks.

4.6. Limitations

This study evaluates only four AED methods and a single end-model architecture (MLP), which may limit the generalization of our findings. Alternative AED techniques or model families (e.g., SVM, k-NN) could yield different detection efficacy. Moreover, experiments are restricted to two spam-classification datasets; tasks with distinct domains or label imbalance may influence AED performance. Finally, our assessment of computational cost is based only on runtime, whereas real-world applications often involve additional factors such as monetary costs, human resources, data availability, and infrastructure.

5. Related Work

The field of annotation error detection (AED) encompasses diverse strategies aimed at identifying mislabeled data, which can degrade model performance and reliability. [Klie et al. 2022] addresses the lack of standardization in AED for natural language processing (NLP) by evaluating 18 methods across nine English-language datasets. These methods are grouped into six categories: variation-based, model-based, training dynamics, vector space proximity, ensembling, and rule-based. The tasks covered include text classification, token labeling, and span labeling. While their analysis is comprehensive, it is limited to English datasets and does not assess computational trade-offs. A key contribution is an open-source library implementing all evaluated methods, enabling reproducibility and systematic comparisons under consistent conditions.

George et al. [George et al. 2024] focus solely on model-based AED techniques and propose a unified framework composed of four modular components: a base model, a model probe (used to score and flag potential mislabels), an optional ensemble mechanism (e.g., bootstrapping, boosting, or cross-validation), and an aggregation step to produce a single trust score per example. This structure supports the development of new AED variants through component-level changes. The authors provide a Python implementation and, unlike prior work that uses synthetic noise, simulate label noise via imperfect labeling rules—closer to scenarios seen in data programming pipelines. Their empirical results show the framework's effectiveness on both text and tabular datasets, demonstrating its practical applicability.

6. Final Remarks

This study investigated the integration of AED methods into DP pipelines to address the label noise generated by DP's inherently noisy label sources. Standardized using the Wrench benchmark, the experiment demonstrated that AED methods have the potential to enhance the performance metrics of models trained with refined DP labels. However, certain AED methods not only failed to improve label quality but also introduced computational overhead, underscoring the importance of carefully selecting AED techniques to optimize DP in practical applications. These findings highlight AED as a viable approach for refining DP pipelines but emphasize the necessity of strategic method selection.

Future work should focus on refining AED methods to increase robustness in scenarios involving imbalanced datasets, as the experiment revealed that such conditions negatively impact the performance of the tested AED techniques. Additionally, there is a need to develop new AED methods that balance computational efficiency and label quality, promoting better scalability and applicability in real-world DP pipelines.

Acknowledgments

The authors acknowledge using ChatGPT (`https://chatgpt.com`) to enhance the text regarding clarity, coherence, and readability at the final stages of preparing this manuscript. The authors confirm they critically checked the provided suggestions for possible errors, inaccuracies, and bias and revised the writing using their own words. This technical content and scientific contributions remain the authors' work.

Disclosure of Interests

The authors have no competing interests to declare that are relevant to the content of this article.

References

Alberto, T. and Lochter, J. (2015). YouTube Spam Collection. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C58885.

Almeida, T. and Hidalgo, J. (2011). SMS Spam Collection. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5CC84.

Awasthi, A., Ghosh, S., Goyal, R., and Sarawagi, S. (2020). Learning from rules generalizing labeled exemplars. In *International Conference on Learning Representations*.

Chicco, D. and Jurman, G. (2020). The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21:1–13.

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

Fu, D. Y., Chen, M. F., Sala, F., Hooper, S. M., Fatahalian, K., and Ré, C. (2020). Fast and three-rious: Speeding up weak supervision with triplet methods.

George, T., Nodet, P., Bondu, A., and Lemaire, V. (2024). Mislabeled examples detection viewed as probing machine learning models: concepts, survey and extensive benchmark. *arXiv preprint arXiv:2410.15772*.

Geron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2nd edition.

Klie, J.-C., Webber, B., and Gurevych, I. (2022). Annotation error detection: Analyzing the past and present for a more coherent future.

Nogueira, F. (2014–). Bayesian Optimization: Open source constrained global optimization tool for Python.

Northcutt, C. G., Jiang, L., and Chuang, I. L. (2022). Confident learning: Estimating uncertainty in dataset labels.

Ratner, A., Hancock, B., Dunnmon, J., Goldman, R., and Ré, C. (2018). Snorkel metal: Weak supervision for multi-task learning. In *Proceedings of the Second Workshop on Data Management for End-To-End Machine Learning*, DEEM'18, New York, NY, USA. Association for Computing Machinery.

Ratner, A., Sa, C. D., Wu, S., Selsam, D., and Ré, C. (2017). Data programming: Creating large training sets, quickly.

Sedova, A. and Roth, B. (2023). Ulf: Unsupervised labeling function correction using cross-validation for weak supervision. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, page 4162–4176. Association for Computational Linguistics.

Team, S. (2019). spam. GitHub.

van Halteren, H. (2000). The detection of inconsistency in manually tagged text. In Abeille, A., Brants, T., and Uszkoreit, H., editors, *Proceedings of the COLING-2000 Workshop on Linguistically Interpreted Corpora*, pages 48–55, Centre Universitaire, Luxembourg. International Committee on Computational Linguistics.

Wu, R., Chen, S.-E., Zhang, J., and Chu, X. (2023). Learning hyper label model for programmatic weak supervision. In *The Eleventh International Conference on Learning Representations*.

Zhang, J., Hsieh, C.-Y., Yu, Y., Zhang, C., and Ratner, A. J. (2022a). A survey on programmatic weak supervision. *ArXiv*, abs/2202.05433.

Zhang, J., Wang, H., Hsieh, C.-Y., and Ratner, A. (2022b). Understanding programmatic weak supervision via source-aware influence function.

Zhang, J., Yu, Y., Li, Y., Wang, Y., Yang, Y., Yang, M., and Ratner, A. (2021). WRENCH: A comprehensive benchmark for weak supervision. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Zhou, Z.-H. (2017). A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53.

Zhu, Z., Dong, Z., Cheng, H., and Liu, Y. (2021). A good representation detects noisy labels. *ArXiv*, abs/2110.06283.