SAGED: Few-Shot Meta Learning for Tabular Data Error Detection

Mohamed Abdelaal
Software AG, Darmstadt, Germany
Mohamed.Abdelaal@softwareag.com

Tim Ktitarev
Software AG, Darmstadt, Germany
Tim.Ktitarev@softwareag.com

Daniel Städtler
Hochschule Darmstadt, Germany
Daniel.Staedtler@yahoo.com

Harald Schöning
Software AG, Darmstadt, Germany
Harald.Schoening@softwareag.com

ABSTRACT

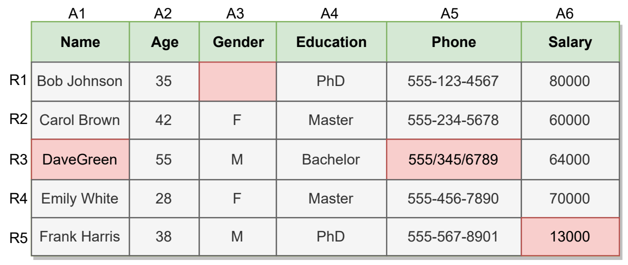
High data quality is paramount for the success of machine learning (ML) applications, as it broadly impacts model performance and decision outcomes. In domains like medical diagnosis and financial systems, inaccuracies or data incompleteness can result in unreliable predictions, posing risks to system users. Unfortunately, real-world data commonly exhibits noise, errors, missing values, and inconsistencies, posing challenges to ML model accuracy. Data cleaning plays a vital role in addressing these issues before model training, yet identifying dirty data instances can be a cumbersome and time-consuming process.

In this context, numerous automated error detection tools for tabular data have been introduced. Nevertheless, these tools suffer from several shortcomings, encompassing the necessity for domain-specific expertise and substantial time requirements. To address these shortcomings, we introduce a novel error detection tool, denoted as SAGED[1][2], which leverage meta-learning principles. Specifically, SAGED exploits an ensemble of pre-trained models derived from historical datasets to facilitate error detection in new data with limited labeled instances. The method consists of two key phases: knowledge extraction and error detection. In the knowledge extraction phase, ML models are trained to distinguish erroneous instances within historical datasets, thereby accumulating valuable insights. In the error detection phase, pre-trained base models, chosen through rigorous matching, generate a comprehensive feature vector based on predictions, facilitating the role of a meta-classifier in pinpointing errors efficiently. As a proof of concept, we conduct a comparative study of SAGED against ten state-of-the-art error detection tools, employing a set of 14 real-world datasets. The findings reveal the superior performance of SAGED in error detection tasks, with limited user intervention. SAGED's promising results demonstrate its effectiveness and efficiency in real-world scenarios.

1 INTRODUCTION

Data Quality Problems: In today's data-driven world, enterprises and organizations across various industries broadly rely on data to drive business growth and gain a competitive edge. Data-intensive industries, e.g., banking, insurance, retail, and telecoms, typically collect diverse types of data, including sensory readings, financial records, and medical reports, to automate business tasks, facilitate better decision-making, understand performance, and satisfy customer requirements [35]. To reap these benefits, businesses leverage analytics and business intelligence tools to extract hidden patterns or effectively predict trends and future events. However, the conclusions drawn from these tools can be misleading when the collected data contains error profiles. Real-world data often contain heterogeneous error profiles that may emerge during data collection or transfer. Some common data quality problems include missing values, duplicates, numerical outliers, inconsistencies, and violation of business and integrity rules (cf. Figure 1). Consequently, ensuring the accuracy and reliability of the collected data becomes an essential prerequisite for effective data-driven applications.

[1]The source code of SAGED, along with the baseline methods, is available at https://github.com/mohamedyd/SAGED
[2]SAGED is an abbreviation for **S**oftware **AG** **E**rror **D**etection



	A1	A2	A3	A4	A5	A6
	Name	Age	Gender	Education	Phone	Salary
R1	Bob Johnson	35		PhD	555-123-4567	80000
R2	Carol Brown	42	F	Master	555-234-5678	60000
R3	DaveGreen	55	M	Bachelor	555/345/6789	64000
R4	Emily White	28	F	Master	555-456-7890	70000
R5	Frank Harris	38	M	PhD	555-567-8901	13000

Figure 1: Example of a dirty dataset which exhibits various types of errors, including a typo (R3, A1), a missing value (R1, A3), an improper formatting (R3, A5), and a numerical outlier (R5, A6).

Challenges: In general, data cleaning tasks involve identifying and correcting errors, handling missing values, removing duplicates, and resolving inconsistencies. Performing such tasks manually can be laborious and time-consuming, especially for large datasets. Hence, numerous commercial and academic tools have been developed recently to facilitate (semi)-automated detection and repair of data errors. Examples of such tools include, RAHA [24], OpenRefine and Trifacta [30], and AutoCure [2]. These tools primarily aim to streamline the data cleaning process, offering time and effort savings while enhancing data quality. However, despite their advantages, these tools still exhibit some limitations. First, the ML-agnostic tools, e.g., NADEEF [10] and HoloClean [32], typically require users to provide cleaning signals, such as functional dependencies rules and integrity constraints, which can be a cumbersome task. Second, the semi-supervised tools, e.g., ED2 [28] and HoloDetect [18], often necessitate extensive execution time and numerous user labels to identify erroneous data instances, leading to poor scalability, which can be a significant drawback while working with large volumes of data or time-sensitive tasks. Third, existing data cleaning tools are mostly context-blind, meaning they overlook context information and historical knowledge that could help make better cleaning decisions [13].

Proposed Method: To address these challenges, we introduce a novel solution called SAGED, which is a meta-learning-based error detection tool specifically designed for tabular data. In general, meta-learning exploits pre-trained models that have been learned in prior tasks to facilitate the learning of new tasks or domains with limited labeled data. Pillared on this concept, SAGED has been designed to harness the insights derived from a set of pre-cleansed historical data. By capitalizing on this reservoir of knowledge, we can significantly reduce the execution time without compromising the accuracy of error detection. It is crucial to acknowledge that our approach is fundamentally motivated by the ubiquitous presence of historically cleaned data exhibiting comparable error profiles in a wide array of real-world applications. By tapping into this rich resource, SAGED leverages the inherent similarities within these datasets to enhance its detection capabilities. This enables our tool to efficiently and effectively pinpoint errors, even in complex and multifaceted tabular data structures commonly encountered in practical scenarios. It is crucial to mention that SAGED's operation is independent of the domain congruity between the historical and input dirty datasets. As shown in Section 5, SAGED effectively functions with as few as two historical datasets.

In practice, SAGED consists of two sequential phases, namely the *knowledge extraction* phase and the *detection* phase. The knowledge extraction phase aims to train a series of ML models (i.e., one binary classifier for each column) to differentiate between erroneous and clean instances within the historical datasets. The detection phase begins by selecting a set of these pre-trained models, which are chosen through a rigorous matching process that aligns the characteristics of the dirty dataset with those of the historical datasets. This matching ensures that the selected pre-trained models possess relevant knowledge to address the specific errors in the input dirty dataset. Once the appropriate models have been selected, SAGED utilizes them to generate high-level feature vectors. Specifically, the feature vectors are the predictions generated by the selected base pre-trained models. These predictions encapsulate valuable insights and patterns learned from the historical datasets, representing an abstract representation of the knowledge extracted from the base pre-trained models. The feature vectors are then utilized to train meta-classifiers to precisely detect errors in each column of the input dirty dataset.

Summary of Contributions: The paper provides the following contributions: (1) We introduce a novel two-stage architectural framework to detect heterogeneous errors in tabular data. This framework encompasses a comprehensive approach that combines the power of meta-learning and semi-supervised learning techniques. By leveraging this combined approach, our framework achieves significant improvements in both the effectiveness and efficiency of error detection, together with enhancing SAGED's ability to handle diverse and complex error patterns present in tabular data. (2) SAGED implements several advanced strategies which collectively contribute to the accuracy and efficiency of error detection. These strategies encompass various stages, including *dataset matching* to ensure that the most relevant and informative insights are leveraged during the error detection process, *tuples selection for labeling* which enhances the utilization of limited labeling resources, and *label augmentation* which increases the labeled tuples to further improve the meta classifiers. (3) We conduct extensive experiments to evaluate SAGED in comparison to a wide collection of baseline detection tools. The evaluation encompasses several metrics, including detection accuracy, efficiency, and the impact of error detection on downstream ML models. The results consistently demonstrate that SAGED significantly reduces detection time while achieving state-of-the-art accuracy. Notably, SAGED exhibits superior performance compared to baseline tools in terms of both accuracy and efficiency. This evaluation provides empirical evidence of the effectiveness and practicality of SAGED. To the best of our knowledge, SAGED is the first error detection tool that combines meta-learning and semi-supervised learning for error detection in structured data.

2 SYSTEM OVERVIEW

Before delving into the architecture of SAGED, we first define the structure of tabular data and dirty instances. Let's denote a tabular dataset as D, which consists of N tuples and M columns. Each tuple, indexed by $i \in 1, 2, \ldots, N$, represents an instance or observation, while each column, indexed by $j \in 1, 2, \ldots, M$, corresponds to a specific attribute or feature. The value in the cell located at the intersection of the i-th tuple and j-th column is denoted as d_{ij}, where d_{ij} represents the observed data point. In this context, a dirty cell refers to a specific cell, denoted as d_{ij}^{dirty}, within a tabular dataset D that deviates from expected data quality standards or exhibits anomalies.

To illustrate the process and capabilities of SAGED, we will walk through a detailed running example. This example utilizes a fictional dataset of employee records to demonstrate how SAGED detects errors in real-world scenarios. Figure 1 depicts an example of historical HR datasets collected from 2018 to 2022, containing records with various known errors. For instance, if cell d_{ij} is undefined, i.e., represented as $\emptyset$ or NULL, it indicates the absence or lack of recorded information, e.g., the cell (R1, A3) in Figure 1. Alternatively, a cell is considered an outlier if its value significantly deviates from the expected range or statistical distribution of the corresponding column, e.g., the cell (R5, A6). Another error type is inconsistency which arises when d_{ij} conflicts with predefined data constraints, violating data type conventions, logical dependencies, or domain-specific rules, e.g., the cell (R3, A5). Finally, cells containing errors due to data entry mistakes, computational errors, sensor inaccuracies, or transmission glitches are also classified as erroneous, e.g., the cell (R3, A1). In the running example, data engineers may manually examine data or employ dataset-specific scripts to detect and correct errors in historical datasets. However, they found this process to be labor-intensive due to the challenges outlined in Section 1. Herein, they decided to exploit SAGED that leverages prior efforts and offers a dataset-agnostic automated solution for identifying errors $H_{\text{dirty}} = \{(i, j) \mid d_{ij}^{\text{dirty}} \in D\}$ in new or similar datasets, e.g., HR data gathered after 2022.

Figure 2 demonstrates the architecture of SAGED, which comprises two sequential phases: an offline knowledge extraction phase and an online detection phase. In the knowledge extraction phase, a set of pre-cleansed datasets $\mathcal{D}_{\text{hist}} = \{D_1, D_2, \ldots, D_W\}$ are used as input for the automatic featurization module. Within each historical dataset, every column $C_{kj} \in D_k$ is accompanied by a set of labels denoted as $L_{C_{kj}} = l_1, l_2, \ldots, l_N$. These labels indicate whether a particular cell $d_{ij} \in C_{kj}$ is classified as dirty or clean, and they are derived from a prior cleaning process. The primary objective of this module is to generate a feature vector $F_{C_{kj}} \in \mathbb{R}^{z_{kj}}$ for each column C_{kj} in the historical datasets $\mathcal{D}_{\text{hist}}$, where z_{kj} is the number of generated features. To this end,

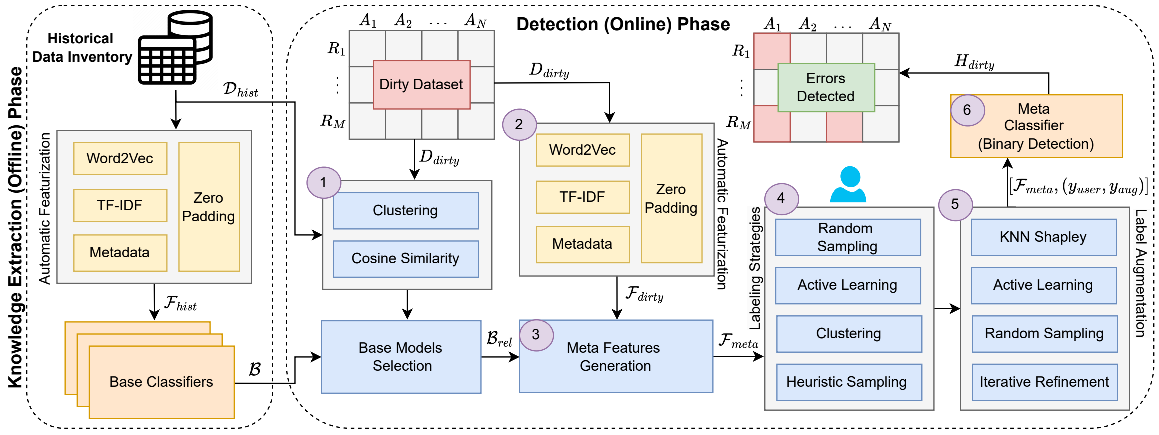


Figure 2: Architecture of SAGED showing several examined strategies for dataset matching and label generation.

we employ the Word2vec embeddings and the TF-IDF weighting scheme, along with extracted metadata about each column. These features serve to differentiate between clean and dirty cells, thereby facilitating subsequent analysis.

For the running example with the hypothetical employee dataset from 2018 to 2022, let's focus on word embeddings for the column *Job Title* and TF-IDF features for the *Email* column. For job titles, Word2vec breaks down each title into individual words, such as "Senior", "Software", and "Engineer" for "Senior Software Engineer". Each word is then represented as a vector in a high-dimensional space, capturing semantic meaning. The resulting vectors are averaged to obtain a single vector representation for the entire job title. For the *Email* column, TF-IDF determines the significance of individual words within email content. Specifically, it calculates the frequency of unique words, adjusted by their frequency across all emails. As a result, domain-specific terms might have higher TF-IDF scores, while common terms, such as "@companydomain.com", will have lower scores due to their high frequency across all emails. Conversely, unique identifiers, like an employee's name, might yield higher TF-IDF scores, emphasizing their importance in the email's context. In this context, instances with erroneous or atypical content are likely to yield elevated TF-IDF scores. For instance, the misspelled domain "companidomain.con" and the use of "(at)" instead of "@" are unique compared to other emails, thus they would have higher TF-IDF scores.

The generated feature vectors may possess varying sizes due to the diverse nature of the input data. However, the presence of feature vectors with different dimensions can introduce challenges when matching with other datasets. To address this problem, we leverage the zero padding technique [17] to standardize the size of all feature vectors. By padding the feature vectors with zeros, we ensure uniform dimensions across the dataset, i.e.,$F^{pad}_{C_{kj}} = [F_{C_{kj}}, 0, 0, \dots, 0] \in \mathbb{R}^z$, where z is the unified size of all feature vectors. For each column in the historical datasets, the padded feature vectors, accompanied by the labels obtained from a previous cleaning process, are utilized to train a binary classifier, e.g., XGBoost or MLP network. Consequently, the outcomes of this phase encompass the list of standardized feature vectors $\mathcal{F}_{hist} = \{F^{pad}_{C_{kj}} \mid k \in [0, W], j \in [0, M]\}$ (for simplicity, we assume that all historical datasets have the same number of columns) and the corresponding base pre-trained classifiers $\mathcal{B} = \{B_1, B_2, \dots, B_{W\times M}\}$.

In the detection phase, SAGED leverages the outcomes of the knowledge extraction phase to identify errors present in dirty datasets. Figure 3 illustrates the detection algorithm, delineating the essential input parameters and the subsequent procedural steps. This phase commences with identifying the most relevant pre-trained base models $\mathcal{B}_{rel} \subseteq \mathcal{B}$ (cf. lines 1-4). To this end, SAGED implements two distinct matching methods, including clustering and cosine similarity (i.e., $\text{sim}(C_t, C_{kj})\ \forall\ C_t \in D_{dirty}, C_{kj} \in \mathcal{D}_{hist}$). Such methods are utilized to determine the compatibility between the pre-trained models and the feature vectors of the dirty dataset. The second step is to extract a feature vector F_{C_t} for each column $C_t \in D_{\text{dirty}}$ (cf. lines 5-10). The feature vectors encompass various elements such as Word2Vec embeddings and TF-IDF scores, as well as metadata. These feature vectors serve as representations capturing the relevant characteristics of the columns. As analogue to the knowledge extraction phase, the generated feature vectors are zero-padded to ensure consistent dimensions, where $\mathcal{F}_{dirty} = \{F_{C_t} \mid C_t \in D_{\text{dirty}}, F_{C_t} \in \mathbb{R}^z\}$ represents the set of zero-padded feature vectors of all columns in the dirty dataset.

The padded feature vectors are then used as input to the selected pre-trained models $\mathcal{B}_{\text{rel}}$ to generate the meta-features $\mathcal{F}_{\text{meta}} = \{F_{C_t} \mid C_t \in D_{\text{dirty}}\}$, which encapsulate higher-level representations derived from the combination of the pre-trained models $\mathcal{B}_{\text{rel}}$ and the padded feature vectors $\mathcal{F}_{\text{dirty}}$ (cf. lines 11-13). To train the meta classifier, which is responsible for error detection in the dirty dataset, a subset of the meta-features needs to be labeled. To accomplish this, SAGED implements various methods for selecting a subset of meta-features that can be labeled by an oracle. These methods include active learning, random sampling, clustering, and heuristic sampling. To augment the number of labeled data without increasing the labeling budget, SAGED explores four additional methods: KNN Shapely, active learning, random sampling, and iterative refinement. These methods enable the expansion of the labeled dataset by incorporating additional labeled instances in a resource-efficient manner. The labeled data, obtained through the aforementioned methods, is then utilized to train the meta classifier, which effectively detects errors within the dirty dataset.

Revisiting the running example, we notice that the 2023 HR dataset's *Age* column is quite similar to the *Age* columns from 2018 and 2020, as shown by their high cosine similarity. Given this similarity, SAGED utilizes the base models associated with these past datasets to generate meta-features $\mathcal{F}^{age,23}_{\text{meta}}$ for the 2023

```
Require: historical data $\mathcal{D}_{\text{hist}}$, dirty dataset $D_{\text{dirty}} = A_1, \ldots, A_M$, base classifiers $\mathcal{B}$, size of feature vector $z$, labeling budget $\gamma_l$, similarity threshold $\theta_{\text{sim}}$
1:  for all column $C_t$ in $D_{\text{dirty}}$ and $C_{kj}$ in $\mathcal{D}_{\text{hist}}$ do          ▷ Section 3.1
2:      Estimate $S_{t,kj} \leftarrow$ similarity($C_t, C_{kj}$)
3:      if $S_{t,kj} \geq \theta_{\text{sim}}$ then
4:          Append $\mathcal{B}_{\text{rel}} \leftarrow \mathcal{B}_{\text{rel}} + B_{kj}$          ▷ Relevant base models
5:  for all column $A_i \in D_{\text{dirty}}$ do                                   ▷ Section 3.2
6:      $F_{A_i} \leftarrow$ [TF_IDF($A_i$), Word2Vec($A_i$), profiler($A_i$)]
7:      if $|F_{A_i}| < z$ then
8:          Compute $Z_i \leftarrow z - |F_{A_i}|$          ▷ # of zeros to be appended
9:          Generate $\overrightarrow{zeros} \leftarrow$ zeros($Z_i$)          ▷ Create a zeros vector
10:         $F^{\text{pad}}_{A_i} \leftarrow$ Concatenate($F_{A_i}, \overrightarrow{zeros}$)          ▷ Zero padding
11:     for all model $m_j$ in $\mathcal{B}_{\text{rel}}$ do
12:         Append $F^{\text{meta}}_{A_i} \leftarrow F^{\text{meta}}_{A_i}$ + predict($F^{\text{pad}}_{A_i}, m_j$)
13:     Append $\mathcal{F}_{\text{meta}} \leftarrow \mathcal{F}_{\text{meta}} + F^{\text{meta}}_{A_i}$          ▷ All meta features
14: Estimate $y_{\text{user}} \leftarrow$ get_labels($\mathcal{F}_{\text{meta}}, \gamma_l$)          ▷ Section 4.1
15: Train $\mathcal{B}_{\text{meta}} \leftarrow$ mlp($\mathcal{F}_{\text{meta}}, y_{\text{user}}$)
16: Find $H_{\text{dirty}} \leftarrow$ predict($\mathcal{F}_{\text{meta}}, \mathcal{B}_{\text{meta}}$)          ▷ Error detection
17: Compute $y_{\text{aug}} \leftarrow$ aug_labels($\mathcal{F}_{\text{meta}}, H_{\text{dirty}}$)          ▷ Section 4.2
18: Update $\bar{\mathcal{B}}_{\text{meta}} \leftarrow$ mlp($\mathcal{F}_{\text{meta}}$, [$y_{\text{user}}, y_{\text{aug}}$])
19: Refine $H_{\text{dirty}} \leftarrow$ predict($\mathcal{F}_{\text{meta}}, \bar{\mathcal{B}}_{\text{meta}}$)
```

Figure 3: Error detection algorithm

Age data. These meta-features are thus expressed as $\mathcal{F}^{age,23}_{\text{meta}} = [B_{age,18}(F_{age,23}), B_{age,20}(F_{age,23})]$. The elements of this array are derived from the outputs of the base models $B_{age,18}$ and $B_{age,20}$, which have been previously trained on the padded features (i.e., metadata, Word2Vec, and TF-IDF) of the *Age* columns from the 2018 and 2020 datasets, respectively. The inputs to these base pre-trained models are the padded feature vectors $F_{age,23}$– corresponding to the *Age* column from the 2023 dataset. In this manner, the meta-learning part of SAGED facilitates the exploitation of prior cleaning efforts.

Following this, the semi-supervised learning part commences with manually labeling a subset of the data to provide training instances for a meta-classifier. Labeling is a prerequisite for semi-supervised learning in all ML-based error detection tools. As shown in Section 5, SAGED achieves superior performance over baseline tools with significantly fewer labels. This classifier is then employed to detect and categorize various errors within the *Age* column. In this example, we utilized historical data from the same domain as the new, dirty dataset. However, our experimental evaluations (cf. Section 5) demonstrate that SAGED remains effective even when the historical and input dirty datasets originate from distinct domains. This cross-domain generalizability highlights the robustness and adaptability of the proposed method in diverse real-world scenarios where domain alignment may not be guaranteed.

3 META LEARNING APPROACH

The offline phase primarily encompasses the extraction of diverse features and the training of base pre-trained models. Given that analogous feature extraction techniques will be applied during the online phase, we have structured this paper to introduce the components in the sequence of their execution within the online phase. In this section, we explain the adopted similarity measures and automated featurization techniques that underpin the implementation of our meta-learning approach.

3.1 Similarity Measures

A pivotal step in our meta-learning approach involves the identification of historical datasets that exhibit a high degree of similarity with the given dirty dataset. Empirical evidence from our experiments indicates that columns with similar characteristics across various datasets typically exhibit comparable error profiles. Accordingly, the identification of similar columns is crucial for SAGED to effectively leverage base pre-trained models that are most pertinent to the characteristics of the given dirty dataset. SAGED implements two distinct techniques for identifying highly similar datasets, namely cosine similarity and clustering. The former method quantifies how similar the data instances in the dirty dataset are to the historical datasets. Before computing the cosine similarity, SAGED ensures normalization for all attributes from both the historical and dirty datasets. If the similarity between column C_t and historical column C_{kj} surpasses a predetermined threshold θ_{sim}, then the associated base model B_{kj} will be incorporated into the set of relevant models $\mathcal{B}^{C_t}_{rel}$ for column C_t.

The second technique entails clustering the columns of the historical datasets $\mathcal{D}_{\text{hist}}$, where each cluster comprises the most closely related columns from these datasets. The core idea behind this technique is to assign each dirty column to the cluster containing the most similar historical columns. Figure 4 demonstrates an example of the clustering technique employed to find the historical attributes mostly similar to the dirty attributes. The historical columns denoting attributes like the age of employees C_{11}, customers C_{23}, or patients C_{33} are grouped in a single cluster (e.g., cluster K in Figure 4). As shown in the figure, each historical column is associated with a pre-trained base model (e.g., model B_{11} is associated with column C_{11}), which has been generated in the knowledge extraction phase. To generate such clusters, SAGED extracts a set of metadata features for each column in the historical datasets (cf. Section 3.2). These extracted features are then employed to train a K-Means algorithm. During the online phase, the trained K-Means inference function is utilized to assign each column $C_t \in D_{\text{dirty}}$ to one of the clusters established during the offline phase. Upon successful assignment to a particular cluster, SAGED retrieves the corresponding base pre-trained models $\mathcal{B}^{C_t}_{\text{rel}}$ that correspond to the historical columns encompassed within the assigned cluster. These retrieved models $\mathcal{B}^{C_t}_{\text{rel}}$ are then utilized to generate the meta-features $\mathcal{F}^{C_t}_{\text{meta}}$, thus facilitating effective adaptation and generalization of the learning process for the dirty dataset.

3.2 Feature Representation

In this section, we elaborate on the various features that must be extracted to enable error detection. SAGED extracts a comprehensive set of representative features, accurately characterizing the underlying distribution of each column in both historical and dirty datasets. To this end, a combination of methods and techniques is employed to automatically extract these features, including metadata profiling, Word2Vec embeddings, and the TF-IDF weighting scheme. To generate these features, a metadata profiler systematically traverses each column within the dataset and performs computations on various parameters. These parameters encompass the frequency of values occurring in a column, the proportion of explicitly specified missing values, the character count of each value, the proportion of alphabetic values in a column, the proportion of numeric values in a column, the

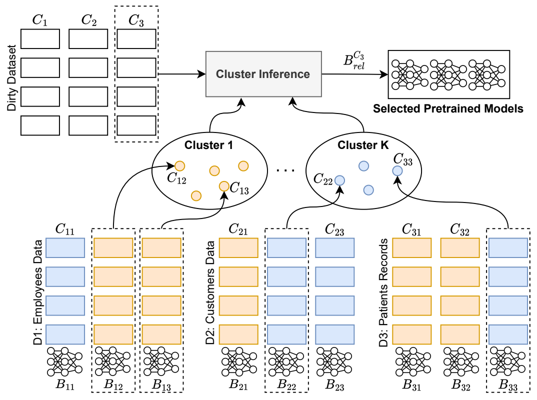


Figure 4: Example of assigning columns of a dirty dataset to the clusters generated in the knowledge extraction phase.

proportion of values in a column with punctuation, and the proportion of unique values in a column.

The extracted metadata provides valuable insights into the underlying structure of the data within each column, enhancing our understanding of its characteristics and distribution. Additionally, it is pertinent to acknowledge that the scope of metadata features can be expanded further by incorporating additional relevant parameters. The second set of features is obtained using Word2Vec embeddings[15], a widely adopted technique in natural language processing and information retrieval. Word2Vec efficiently creates word embeddings, representing words or strings as fixed-size numeric vectors. This process facilitates the extraction of contextual information, semantic and syntactic similarities, and word relationships within documents. To this end, SAGED trains a standard word embedding model, treating each tuple as a document. Within this model, vectors are generated for individual words, capturing their interdependence with their surrounding context. When a Word2Vec vector exhibits a statistically significant dissimilarity from other vectors within the same column, it can serve as informative evidence for detecting potential errors.

Finally, the TF-IDF (Term Frequency-Inverse Document Frequency) weighting scheme [34] is a statistical measure widely employed to assess the significance of words, characters, or n-grams within a corpus of documents. In the context of the error detection problem, the terms “corpus” and “document” are defined as follows: each value in a column represents a document, while the values–encompassing the entire column–form the corpus. In the context of SAGED, the TF-IDF method is thoughtfully implemented at the character level, wherein each value in a column is assigned a vector of TF-IDF values specific to individual characters. The computation of TF-IDF values involves two essential metrics: the frequency of a character’s occurrence within a document (TF), and the inverse document frequency of the character across the entire corpus (IDF). For instance, Equation 1 expresses the TF-IDF value of the character X in tuple i, where $\alpha(X, i)$ denotes the number of character X in tuple i, $\alpha(i)$ is the number of characters in tuple i, and $\beta(X)$ denotes the number of tuples with character X. By harnessing the TF-IDF method at the character level, SAGED broadly captures the relative importance of characters within the context of the dataset. This character-level analysis enables the identification of distinctive character patterns and their significance in the error detection process.

$$\text{TF-IDF}(\text{character } X, \text{tuple } i) = \frac{\alpha(X,i)}{\alpha(i)} \times \log_2\left(\frac{N}{\beta(X)+1}\right) \quad (1)$$

Figure 5 demonstrates the zero padding procedure employed to preserve uniform feature vector dimensions. The number of features generated by the metadata profiler and the Word2Vec model remains constant across all attributes. Conversely, the TF-IDF method is contingent upon the distinct characters present within each column. For instance, consider column C_{11}, which contains the characters 1, 2, and 3; while column C_{12} encompasses the characters 2, 3, 5, and ’s’. Since these two columns contain different character sets, the TF-IDF feature count diverges for each column; specifically, three TF-IDF features for C_{11} and four for C_{12}. To standardize the dimensions of each feature vector, after concatenating metadata, Word2Vec, and TF-IDF features, we introduce zero-padding to fill in the absent characters, assigning them a value of 0. Consequently, the TF-IDF feature vectors for both C_11 and C_{12} encompass five features corresponding to the characters 1, 2, 3, 5, and ’s’. In this case, the feature values for characters 5 and ’s’, in the case of C_{11}, are set to zero. This padding procedure thus facilitates the harmonization of feature vectors, enabling the application of pre-trained base models to the columns within the dirty datasets in the online phase.

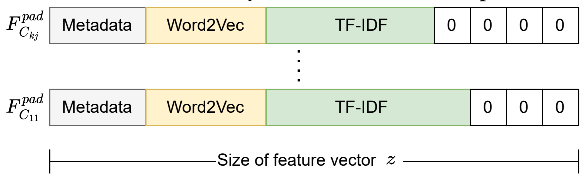


Figure 5: Zero padding of feature vectors.

4 SEMI-SUPERVISED LEARNING

In this section, we present the tuple sampling strategies for labeling and the label augmentation methods required to achieve a semi-supervised detection of tabular data errors within the confines of a constrained labeling budget.

4.1 Tuple Selection for Labeling

As described in Section 2, the detection phase initiates with the selection of a set of base pre-trained models B_{rel} for generating meta features $\mathcal{F}_{meta}$. Subsequently, such meta-features are utilized to train a meta-detection classifier. To enable semi-supervised learning, a subset of the meta-features requires labeling. One of the main objectives of SAGED is to reduce the labeling budget γ_l while preserving the detection accuracy. Hence, we implemented four distinct strategies for selecting tuples that will receive labels from an oracle, including random sampling, heuristic-based sampling, clustering-based sampling, and active learning. The first strategy entails a random selection of tuples for labeling, making it straightforward but not leveraging any external knowledge. Alternatively, the heuristic-based sampling strategy adopts a more informed approach by considering the values within the meta-features. These meta-features encapsulate inferences from pre-trained models, specifically designed to discriminate between dirty and clean data instances. Consequently, the heuristic-based sampling strategy involves a meticulous examination of each tuple within the meta-features, followed by the selection of a subset with the highest occurrence of positive values (i.e., ones). This selection criterion is predicated on the assumption that a greater prevalence of positive values may signify a higher likelihood of the data instance being deemed dirty.

The clustering-based sampling strategy, inspired by [24], primarily aims to address the challenge of limited labeled data by exploiting the cluster assumption [8]. Such an assumption suggests that when two data points belong to the same cluster, they are likely to share the same class label. Therefore, by clustering data cells, we can expect cells within each cluster to have consistent labels, i.e., either dirty or clean. Specifically, the clustering-based sampling strategy employs the hierarchical Agglomerative clustering algorithm to cluster each column of the meta-features. To ensure contextual relevance for users during labeling, the strategy selects entire tuples rather than individual data cells. This allows users to better assess the data cell quality within the context of its associated tuple. To this end, the strategy conducts k clustering iterations, where $k = \gamma_l$, and in each iteration, the cells belonging to every column in the meta-features are clustered. The strategy iteratively increments the number of clusters, which can be generated by the Agglomerative clustering algorithm, and solely one tuple is sampled for labeling in each iteration. To select such a tuple, the cluster labels for each column are obtained, and the number of labels per cluster is counted. A probability distribution p, using the Softmax probability function, is then computed based on the label count. Through the distribution p, we can select a tuple that covers the most number of unlabeled clusters, contributing to a more representative labeling process. This iterative sampling process continues until the allocated labeling budget is fully utilized. It is important to highlight that SAGED avoids label propagation, proposed in [24], since it increases the number of noisy labels, thus adversely impacting the detection accuracy.

Finally, the active learning strategy, inspired from [28], aims to improve the performance of meta classifiers, i.e., detection classifiers generated for each column $C_t \in D_{\text{dirty}}$, by iteratively selecting and labeling the most informative data cells. To this end, it directs labeling efforts toward uncertain regions of the feature space. Specifically, the strategy comprises two main components: column selection and data cell selection. The column selection component determines which column should be labeled next by analyzing the predictions made by the existing meta-classifiers for each column in the dirty dataset. It identifies the column that would benefit the most from acquiring new labels. Classification models typically provide a probability score for each prediction, indicating the certainty of the prediction. A higher certainty implies higher convergence. The strategy calculates the average certainty of all cells within each column and selects the column with the lowest average certainty. Once a column is selected, the data cell selection component randomly samples a set of data cells within the chosen column. Afterward, an oracle manually verifies these selected cells. The newly labeled data cells are incorporated into the training set for the selected column's meta-classifier, which is then retrained with the augmented dataset. The active learning process continues in a loop until the entire labeling budget is utilized.

4.2 Label Augmentation

Label augmentation is an approach that aims to increase the quantity of labeled data while adhering to a constrained labeling budget. Figure 6 shows that this approach employs the predictions generated by initial meta-classifiers that have been trained using user-provided labels. Notably, our experimental findings indicate that these classifiers typically exhibit high precision, instilling confidence in their predictions and warranting their reuse to augment the training data in subsequent iterations. Within the context of SAGED, we implemented four distinct methods to sample the predictions, each contributing to the expansion of the labeled dataset, namely random sampling, iterative refinement, active learning, and KNN Shapely. The first method involves the stochastic selection of a subset of data cells along with their corresponding labels from the obtained predictions. Although less targeted than other methods, random sampling can still contribute to the expansion of the labeled dataset.

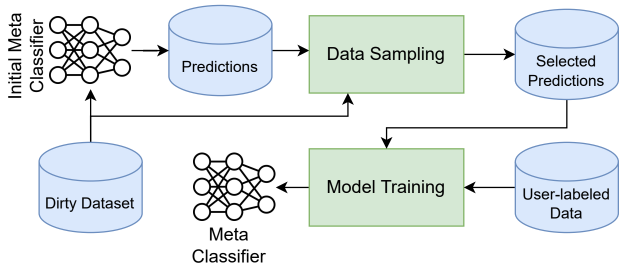


Figure 6: Augmenting user labels with model predictions.

To mitigate potential noise introduced by random sampling, the iterative refinement method adopts a more focused approach by exclusively sampling positively labeled instances, i.e., the dirty data cells. The active learning method strategically selects instances that pose challenges to the model's current knowledge, enabling the model to refine its understanding and adapt accordingly. Our implementation of the active learning methods has been introduced in Section 4.1. Finally, the KNN Shapley method [20] assigns relevance scores to individual data cells, highlighting their importance in the model's decision-making process. By prioritizing the data cells with higher Shapley values, the KNN Shapely method effectively enriches the labeled dataset with significant data cells. To this end, the KNN Shapley method combines the K-Nearest Neighbors (KNN) algorithm with Shapley values from cooperative game theory to prioritize the most informative data cells. In our implementation, we opted to select the top 20% most important tuples based on a predefined importance threshold. However, in cases where all tuples within a specific column possess equal importance, we decided to skip such columns, as they do not contribute to the diversification and enrichment of the selected subset.

5 PERFORMANCE EVALUATION

In this section, we present an extensive evaluation of SAGED in comparison to a set of baseline methods. Through a series of carefully designed experiments, we aim to address the following key questions: (1) Which similarity measure method contributes more significantly to enhancing the overall performance of SAGED? (2) Which labeling strategy proves to be the most effective and efficient in generating the meta-features? (3) What is the impact of label augmentation, and which augmentation method is best suited for error detection? (4) What is the influence of the number of historical datasets on the overall performance of SAGED? (5) How does SAGED compare to ML-based baselines in terms of the required labeling budget? (6) How does SAGED compare to the baselines concerning detection accuracy and efficiency? (7) Finally, how does SAGED compare to the baselines when integrated into an ML pipeline, specifically in terms of predictive performance? By addressing these questions, we endeavor to comprehensively demonstrate the strengths and capabilities of SAGED and shed light on its effectiveness and potential advantages over the baseline methods in the context of error detection. We first describe the setup of our evaluations, before discussing the results and the lessons learned throughout this study.

5.1 Experimental Setup

We conducted a comprehensive evaluation of SAGED, employing a diverse set of 14 real-world datasets that encompassed varying data sizes and exhibited distinct error rates. Table 1 provides a summary of the key characteristics of each dataset. It is worth noting that such datasets are commonly encountered in the domain of data cleaning and have been extensively used in related literature. Among these datasets, we specifically reserved two, namely Adult and Movies, for the historical inventory (except for the similarity experiment which explores the impact of increasing the number of historical datasets). The remaining datasets were utilized for evaluating the performance of SAGED in comparison to the baseline tools. Notably, several datasets were included in the evaluation due to their relevance to ML tasks, e.g., regression or classification. Examples of such datasets are Beers, Smart Factory, and Nasa. For scalability analysis, two large datasets, namely Soccer and Tax, were exclusively used. These datasets enabled us to assess the performance of SAGED under conditions of increased data size, a crucial aspect in real-world applications.

Table 1: Datasets used in the evaluation where the error types are rules violation (RV), formatting issues (FI), outliers (OT), missing values (MV), and typos (TP)

Data Set	Rows	Columns	Error Types	Error Rate
Adult [21]	45223	15	RV, OT	0.09
Movies [22]	7390	17	MV, FI	0.06
Beers [19]	2410	11	MV, RV, TP	0.16
Bikes [12]	17378	16	OT, RV	0.1
Hospital [24]	1000	20	TP, RV, FI	0.03
Rayyan [29]	1000	11	MV, TP,RV	0.09
Flights [24]	2376	7	MV, TP, RV	0.3
Restaurants [28]	28788	16	OT, MV	0.15
Soccer [26]	200000	10	MV, OT, RV	0.27
Tax [6]	200000	15	TP, FI, RV	0.04
Breast Cancer [11]	700	12	MV, TP, OT	0.4
Smart Factory [7]	23645	19	MV, OT	0.83
Nasa [37]	1504	6	MV, OT, TP	0.13
Soil Moisture [33]	679	129	MV, OT	0.3

We present a comprehensive comparison of SAGED against a diverse set of baseline tools, encompassing various methodologies. The evaluated baseline tools include ML-based approaches, such as RAHA [24] and ED2 [28]; rule-based techniques, such as HoloClean [32] and NADEEF [10]; knowledge base-powered tools, including KATARA [9]; ensemble methods, such as dBoost [25] and min-K [3]; and outlier detectors, such as FAHES [31], standard deviation (SD), isolation forest (IF), and inter-quartile range (IQR) [40]. The chosen baseline tools represent a comprehensive range of state-of-the-art solutions in the field of error detection and data cleaning. Each tool brings a distinct approach to the problem, incorporating various techniques and algorithms. Further information about these tools and their specific methodologies can be found in Section 6.

As evaluation metrics, we employ precision, recall, F1 score, and runtime to assess the effectiveness and efficiency of our proposed approach. In this context, precision (P) quantifies the proportion of relevant instances (i.e., actual erroneous cells) among the detected instances, represented as $P = \frac{t_p}{t_p+f_p}$, where t_p and f_p denote true positives and false positives, respectively. On the other hand, recall (R) is defined as the fraction of erroneous instances that are correctly detected, given by $R = \frac{t_p}{t_p+f_n}$, where f_n represents false negatives. The F1 score is computed as the harmonic mean of precision and recall, given by $F1 = 2 \cdot \frac{P \cdot R}{P+R}$. The F1 score provides a balanced measure that takes into account both precision and recall, making it suitable for evaluating the overall performance of the error detection process. Additionally, we consider the runtime, which represents the time elapsed while traversing an entire dataset to identify the erroneous cells. This metric quantifies the efficiency of our approach in terms of computational time. All experiments have been repeated ten times, where the means of the ten runs are reported. We run all the experiments on an Ubuntu 20.04 LTS machine with 16 2.60 GHz cores and 64 GB memory.

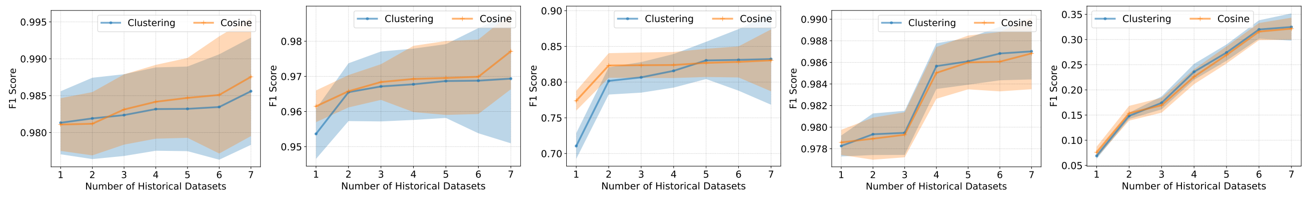
5.2 Results

Before delving into the comparative analysis between SAGED and the baseline tools, it is crucial to assess the influence of the various components of SAGED, including similarity measures, labeling strategies, and label augmentation methods [3].

Similarity. Figure 7 depicts a comparison of the performance of SAGED when leveraging cosine similarity and clustering techniques to identify relevant base pre-trained models. For instance, Figure 7a illustrates the F1 score of SAGED when applied to the Beers dataset, considering varying numbers of historical datasets. As depicted in the figure, both clustering and cosine similarity contribute significantly to SAGED's performance, with comparable effectiveness (i.e., both have an average F1 score of 98.3%). Similar findings have been observed for other datasets, as depicted in Figures 7b-7e. Furthermore, the presented figures indicate that increasing the quantity of historical datasets has a beneficial effect on the accuracy of error detection. However, it is important to note that this impact exhibits variations among the diverse datasets. For instance, when considering Flights and Soil Moisture datasets, a steep curve is observed, wherein augmenting the number of historical datasets from one to seven results in performance improvements of 18% and 371%, respectively. Conversely, the remaining datasets, such as Beers, Movies, and Smart Factory, display a more gradual curve in response to the increase in the number of historical datasets. Regarding the detection time, The results show that both clustering and cosine similarity exhibit similar impacts on the performance of SAGED. Additionally, augmenting the number of historical datasets results in a linear increase in the detection time of SAGED. For example, augmenting the number of historical datasets from one to seven results in a 47% rise in the detection time of SAGED for the Beers dataset. This is attributed to the utilization of more meta-features for training the meta classifier.

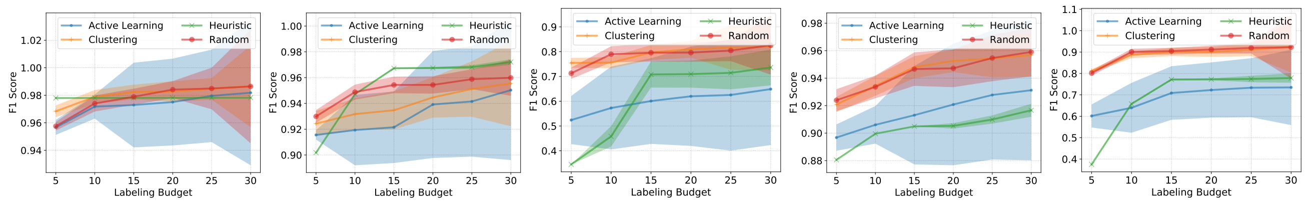
Labeling Strategy. Figures 8-9 present a comprehensive evaluation of four labeling strategies concerning the detection accuracy and detection time of SAGED. For instance, Figure 8a delineates the F1 score of SAGED while identifying errors in the Beers dataset, considering different labeling budgets. The figure highlights that both clustering and random sampling slightly outperform other strategies (both achieve an average F1 score of 98%), with active learning exhibiting higher variance. Similar findings are observed across other datasets, as depicted in Figures 8b-8e, except for the Breast Cancer dataset (Figure 8b), where heuristic-based sampling demonstrates superior performance. For the Flights, Hospital, and Rayyan datasets, random sampling and clustering exhibit superior performance compared to heuristic and active learning (e.g., random sampling outperforms active learning by 24% in the case of the Flights dataset).

[3]Due to space constraints, only a subset of the results is presented here. Comprehensive findings will be uploaded to the project's online repository.



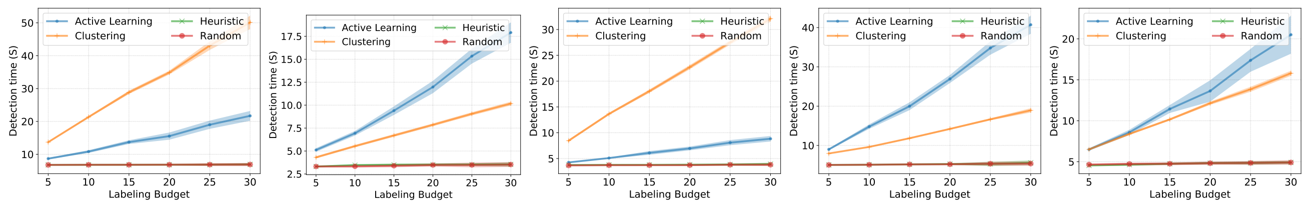
(a) Beers (b) Movies (c) Flights (d) Smart Factory (e) Soil Moisture

Figure 7: Comparison of different similarity measures in terms of detection accuracy over a different number of historical datasets used in the offline phase.



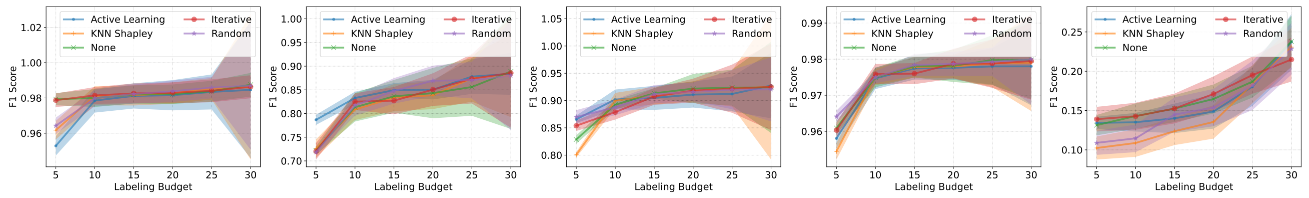
(a) Beers (b) Breast Cancer (c) Flights (d) Hospital (e) Rayyan

Figure 8: Comparison of different labeling strategies in terms of detection accuracy.



(a) Beers (b) Breast Cancer (c) Flights (d) Hospital (e) Rayyan

Figure 9: Comparison of different labeling strategies in terms of detection time.



(a) Beers (b) Bikes (c) Rayyan (d) Smart Factory (e) Soil Moisture

Figure 10: Comparison of different label augmentation methods in terms of detection accuracy.

The superiority of random sampling, observed across most datasets, can be attributed to its ability to provide a diverse set of samples drawn from the meta-features. Additionally, random sampling ensures that all regions of the meta-features have an equal chance of inclusion in the training set, thereby mitigating potential biases that might arise from selective sampling. In terms of the detection time, Figure 9 reveals that, for various datasets, both random sampling and heuristic-based sampling exhibit notably reduced time requirements compared to active learning and clustering (e.g., random sampling requires less time than active learning by on average 54% for the Beers dataset). Random sampling's computational efficiency stems from its straightforward implementation, involving random selection of samples without complex decision-making or querying. Conversely, as the labeling budget increases, the detection time exhibits linear growth when employing active learning and clustering strategies.

Label Augmentation. Figure 10 shows the evaluation of SAGED while employing diverse label augmentation methods under varying labeling budgets. In these experiments, we augmented the user-labeled features with 20% of the predictions generated by the meta-classifier for each column. As a baseline, we also included the case of no-label augmentation. Surprisingly, the results indicate no substantial difference between the various augmentation methods. In several instances, SAGED demonstrates robust performance even without label augmentation, as observed with the Rayyan and Smart Factory datasets. Moreover, it is noteworthy that the KNN-Shapley method did not perform well in most cases, which suggests its limited effectiveness in this context. Drawing upon the outcomes of these experiments and the aforementioned ones, we have made an informed decision to configure SAGED for the subsequent set of experiments. Specifically, we will employ clustering as the similarity measure method, adopt random sampling as the labeling strategy, opt for no label augmentation, and use the Adult and Movies datasets in the historical inventory during the evaluation process. These settings were chosen based on their demonstrated effectiveness and efficiency in previous evaluations, making them well-suited for further analysis.

Labeling Budget. In this series of experiments, we conduct a comparative analysis between SAGED and ML-based baseline tools, focusing on the impact of the required labeling budget. Figures 11-12 present the accuracy and efficiency evaluations of the compared tools across four datasets, while detecting errors. For the Beers dataset (cf. Figure 11a), SAGED consistently outperforms RAHA and ED2 even with small labeling budgets (on average by 19.5% and 28% compared to ED2 and RAHA, respectively). However, for the Bikes, Flights, and Smart Factory datasets, the performance of ED2 becomes comparable to SAGED when the labeling budget is relatively large. Regarding efficiency, as shown in Figure 12, both SAGED and RAHA exhibit minimal time requirements compared to ED2, which displays a linear increase in detection time with a larger labeling budget (SAGED saves 91% of time needed by ED2 for the Beers dataset). Notably, SAGED consistently demonstrates promising results across different datasets, even with limited labeling budgets, highlighting its potential for efficient error detection tasks.

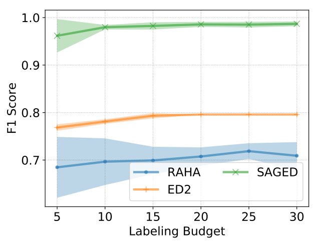
Error Detection. In this series of experiments, we conducted a comprehensive comparison of SAGED against all baseline tools across diverse datasets. Table 2 provides a concise summary of the obtained results, showcasing the detection precision, recall, F1 score, and detection time for each tool. To facilitate the comparison, we have emphasized the best value in each column using bold font and underlined the second-best value. For the detection time column, we have considered the best values in light of the accuracy of the corresponding baseline tools, ensuring that the time value is considered only if the accuracy of the tool exceeds 50%. Throughout these experiments, we maintained a fixed labeling budget of 20 user labels for both SAGED and the ML-based tools. Notably, some simpler baseline tools, such as SD, IF, and IQR, did not detect any errors in certain datasets like Beers and Rayyan. The results highlight SAGED's superior performance, achieving the highest accuracy while also demonstrating the shortest time requirement for most datasets. ED2, on the other hand, exhibits competitive F1 scores, but its extensive time requirements hinder its practicality. For example, in the Hospital dataset, ED2 attains an average F1 score of 99% but requires approximately 209 seconds. In contrast, SAGED achieves an average F1 score of 0.98 in merely five seconds. These findings underscore SAGED's efficiency and effectiveness in error detection tasks.

Robustness. In this series of experiments, we delve into the examination of the robustness of SAGED in comparison to several baseline tools. Robustness, in this context, refers to the ability of the error detection tool to maintain favorable performance across various error rates or outlier degrees (i.e., the magnitude of deviation from the mean). To assess this robustness, we analyze the F1 score and detection time of SAGED and the baseline tools across different datasets, considering diverse error rates. Figure 13 presents the results of these analyses for various datasets. For instance, Figure 13a showcases the accuracy of SAGED and a set of baseline tools while detecting errors in the Hospital dataset, with the error rate ranging from 10% to 50%. Notably, SAGED consistently outperforms the baseline tools (on average by 19% and 21.6% compared to ED2 and RAHA, respectively) across different error rate values. Furthermore, Figure 13b reveals an intriguing observation, wherein the amount of error has no discernible influence on the detection time of SAGED. This finding underscores the efficiency of SAGED, as it consistently requires significantly less time than these baseline tools (e.g., on average by 97%, 96.3%, and 96.6% compared to ED2, KATARA, and dBoost). For the Nasa dataset (cf. Figure 13c), SAGED and ED2 display comparable performance, particularly at an error rate of 20%. However, SAGED continues to outperform ED2, KATARA, and dBoost in terms of time requirements, further highlighting its efficiency in the context of robust error detection.

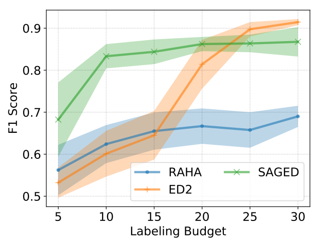
In addition to assessing the error rate, we conducted an evaluation of SAGED and the baseline tools' capability to detect outliers with varying degrees. Figure 14 presents the F1 score and detection time of the compared tools, considering different outlier degrees. For both the Hospital and Nasa datasets (cf. Figures 14a and 14c), SAGED consistently outperforms the other baseline tools, regardless of the outlier degree (SAGED has an average F1 score of 98.6% and 98.3% for the Hospital and Nasa dataset, respectively). Although tools specifically designed for detecting outliers, such as SD and IQR, exhibit improved performance with increasing outlier degrees, they still significantly trail behind the ML-based error detectors, including SAGED, ED2, and RAHA. In terms of the detection time, Figures 14b and 14d illustrate SAGED's remarkable efficiency compared to other ML-based baseline tools. SAGED consistently demonstrates significantly shorter detection times (on average by 93% and 94.6% compared to dBoost and KATARA, respectively), reinforcing its practicality for real-world applications. Overall, the results demonstrate the robustness and efficiency of SAGED across different datasets, error rates, and outlier degrees, substantiating its suitability for error detection tasks in various real-world scenarios.

Scalability. In this series of experiments, we investigate the performance of SAGED and the baseline tools concerning error detection in datasets with large volumes of data. Figure 15 provides insights into the detection time of SAGED and the baseline tools across various data fractions. For instance, considering the Restaurants dataset (cf. Figure 15a), all detectors effectively process all data fractions. However, SAGED exhibits notable superiority over ED2, dBoost, and KATARA, requiring significantly less time (on average by 92%, 97.4%, and 94%, respectively). In terms of accuracy, SAGED achieves an impressive average F1 score of 98.5%, while other tools fail to capture errors effectively. Moving to the Soccer dataset (cf. Figure 15b), some baseline tools, such as RAHA, dBoost, and KATARA, are terminated at a data fraction of 80% due to their limited ability to handle large volumes of data. In contrast, SAGED and ED2 successfully process data fractions, with SAGED requiring substantially less time than ED2 (on average by 98%). Furthermore, SAGED attains an average F1 score of 93% compared to a mere 19% for ED2. Similar commendable results have been observed for the Flights and Tax datasets, as depicted in Figures 15c and 15d. These findings demonstrate SAGED's robustness and efficiency in dealing with large-scale datasets, making it a promising tool for effective error detection across various real-world applications.

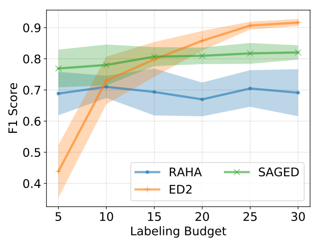
Modeling Accuracy. To comprehensively evaluate the performance of SAGED and baseline tools in ML pipelines, we implemented a neural network using Keras capable of handling regression, binary, and multi-class classification tasks. Leveraging a Bayesian-based informed search technique called Optuna [5], SAGED effectively tunes crucial hyperparameters such as the learning rate, number of hidden layers, and units per layer. Throughout the experiments, we fixed the number of training epochs to 500 to maintain consistency across evaluations. Radar plots, as depicted in Figure 16, showcase the modeling accuracy



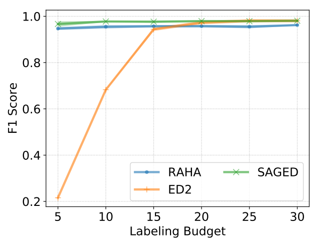
(a) Beers



(b) Bikes

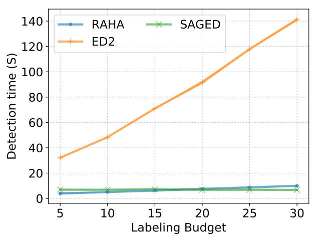


(c) Flights

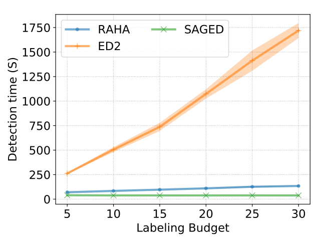


(d) Smart Factory

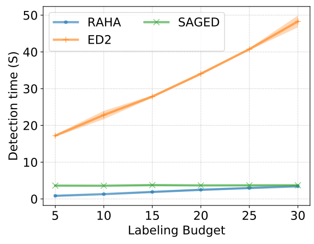
Figure 11: Impact of labeling budget on the detection accuracy of SAGED and the baseline methods.



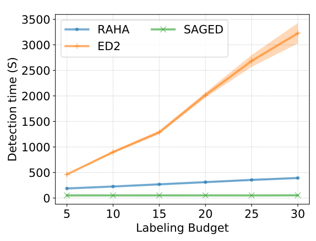
(a) Beers



(b) Bikes



(c) Flights

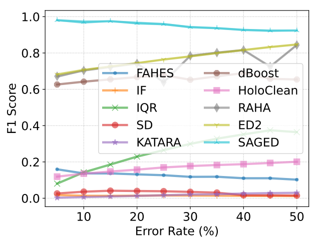


(d) Smart Factory

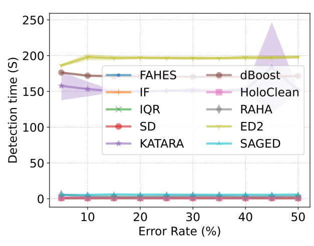
Figure 12: Impact of labeling budget on the detection time of SAGED and the baseline methods.

Table 2: Comparison of SAGED and baseline tools in terms of detection accuracy and runtime

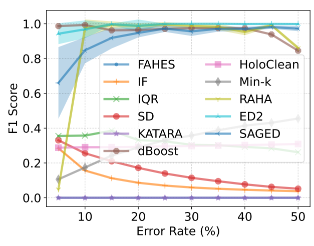
	Beers				Breast Cancer				Flights				Hospital				Nasa				Rayyan			
	P	R	F1	T	P	R	F1	T	P	R	F1	T	P	R	F1	T	P	R	F1	T	P	R	F1	T
FAHES	0.03	0.00	0.00	1.78	0.11	0.04	0.06	0.57	0.35	0.01	0.03	1.23	0.06	0.03	0.04	1.59	**1.00**	0.05	0.09	0.50	0.00	0.00	0.00	0.45
IF	0.00	0.00	0.00	2.00	0.93	0.01	0.01	1.16	0.00	0.00	0.00	0.62	0.00	0.00	0.00	0.74	0.00	0.00	0.00	0.58	0.00	0.00	0.00	0.95
IQR	0.00	0.00	0.00	1.21	0.34	0.03	0.05	0.35	0.00	0.00	0.00	0.65	0.00	0.00	0.00	0.74	0.00	0.00	0.00	0.40	0.00	0.00	0.00	0.49
SD	0.00	0.00	0.00	1.29	1.00	0.01	0.03	0.34	0.00	0.00	0.00	0.63	0.00	0.00	0.00	0.75	0.00	0.00	0.00	0.40	0.00	0.00	0.00	0.49
KATARA	0.02	0.05	0.03	141.76	0.35	0.78	0.48	103.35	0.00	0.00	0.00	72.47	0.08	0.08	0.08	156.01	**1.00**	0.13	0.23	67.44	0.03	0.09	0.04	100.66
HoloClean	0.07	0.05	0.06	4.33	1.00	0.07	0.14	0.71	0.56	0.48	0.52	**1.13**	0.02	0.01	0.01	1.31	**1.00**	0.00	0.01	0.55	NaN	NaN	NaN	NaN
dBoost	NaN	NaN	NaN	NaN	0.54	**0.99**	0.70	69.24	0.81	0.57	0.67	100.99	0.80	0.12	0.21	168.09	0.98	0.82	0.89	64.63	0.15	0.84	0.25	93.50
Min-k	0.74	0.67	0.70	247.36	0.86	0.20	0.27	263.62	**0.88**	0.70	0.78	140.38	0.48	0.02	0.03	540.16	**1.00**	0.46	0.63	165.79	0.71	0.84	0.77	290.4
RAHA	0.76	0.67	0.71	8.11	0.84	0.10	0.18	1.41	0.78	0.66	0.69	2.60	0.27	0.02	0.04	3.73	**1.00**	0.33	0.49	2.36	0.78	0.71	0.74	**0.53**
ED2	**1.00**	0.66	0.80	92.20	0.84	0.10	0.18	86.10	0.86	**0.86**	**0.86**	33.89	**0.99**	0.07	0.13	209.21	**1.00**	0.34	0.51	29.75	0.87	0.29	0.44	95.43
SAGED	0.99	**0.98**	**0.98**	**6.88**	**0.99**	0.92	**0.95**	**3.73**	0.83	0.78	0.80	3.84	0.98	**0.91**	**0.95**	**5.11**	**1.00**	**1.00**	**1.00**	**3.36**	**0.95**	**0.90**	**0.92**	4.77



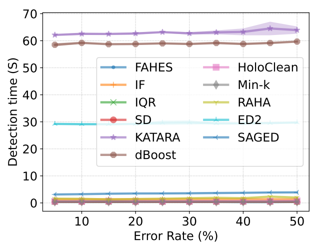
(a) Hospital-Accuracy



(b) Hospital-Time

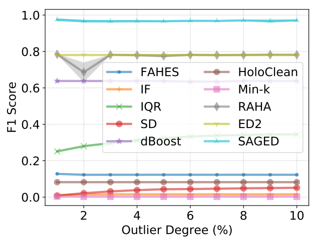


(c) Nasa-Accuracy

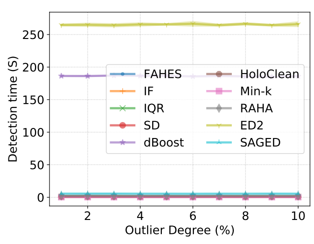


(d) Nasa-Time

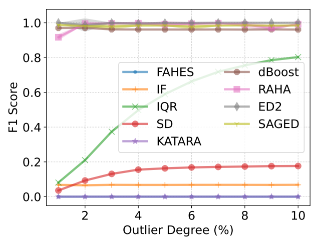
Figure 13: Comparison of SAGED with baseline methods with different error rates.



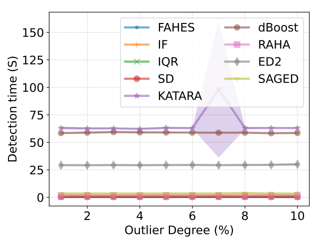
(a) Hospital-Accuracy



(b) Hospital-Time

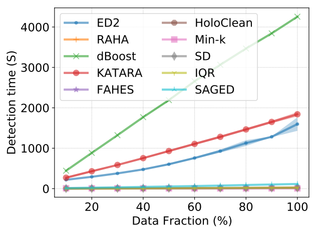


(c) Nasa-Accuracy

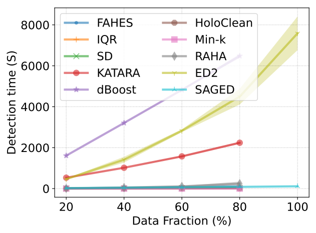


(d) Nasa-Time

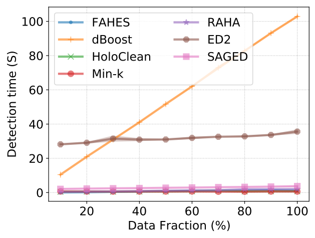
Figure 14: Comparison of SAGED with baseline methods with different outlier degrees.



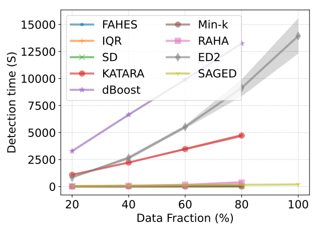
(a) Restaurants



(b) Soccer



(c) Flights



(d) Tax

Figure 15: Scalability analysis

of the neural network when trained on ground truth (indicated by the green area) and repaired data using SAGED (SG2), alongside a set of baseline tools. For repairing the detected errors, we employed an ML-based imputation method that utilizes a decision tree model for numerical attribute repair and missForest for categorical attribute repair. Figure 16a provides insights into the modeling accuracy, represented by F1 scores, of the neural network trained on various versions of the Beers dataset. Remarkably, SAGED achieves performance comparable to the ground truth, attaining an average F1 score of 76.4%, whereas the model trained on the original ground truth data achieves an average F1 score of 79%. Similar commendable results have been observed for the Nasa dataset (regression task) and the Smart Factory dataset (classification task), as illustrated in Figures 16b and 16c. These results substantiate the efficacy of SAGED in effectively detecting errors and improving the overall modeling accuracy of neural networks in various ML pipeline tasks.

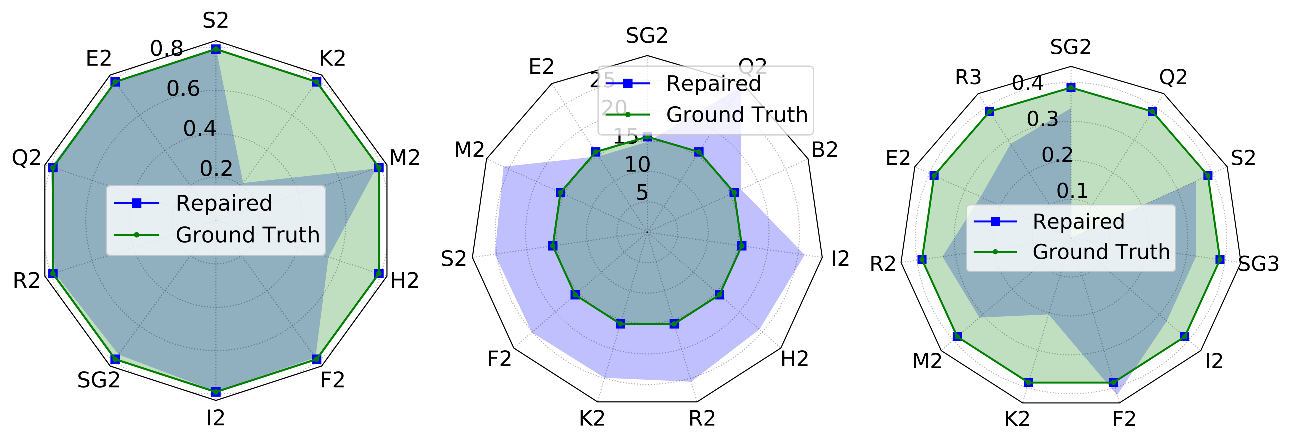
Discussion. In this section, we highlight the main findings and considerations derived from the comprehensive evaluation of SAGED. An ablation study has been carried out to determine the most effective methods and algorithms that can enhance the performance of SAGED. In terms of similarity measures, both cosine similarity and clustering techniques contribute equally to SAGED's performance. Moreover, both techniques have the same time requirements. Notably, increasing the number of historical datasets leads to substantial accuracy improvements, ranging from 18% to a remarkable 371%, depending on the dataset. As the number of historical datasets grows, the likelihood of encompassing a wider range of error patterns and scenarios increases. This diversity in error profiles allows the error detection model to learn from a more extensive spectrum of potential data errors. In the realm of labeling strategies, random sampling and clustering prove highly effective and demonstrate computational efficiency. Random sampling's ability to provide a diverse set of samples and mitigate potential biases proves advantageous. Furthermore, it requires significantly less time, surpassing active learning and clustering strategies.

Aside from the ablation study, the results showed that SAGED consistently outperforms baseline methods across various dimensions. In the context of labeling budgets, SAGED consistently outperforms RAHA and ED2 even with small budgets, showcasing its effectiveness and efficiency. For instance, SAGED saves 91% of the time needed by ED2 for the Beers dataset while achieving an average F1 score of 0.98. The efficiency of SAGED stems from leveraging the random sampling strategy for tuples selection, which proved to be sufficient given the comprehensive meta-features. Such features incorporate not only the characteristics of the dirty data but also encapsulate knowledge of error patterns extracted from historical datasets. Moreover, SAGED's robustness is strikingly evident as it outperforms baseline tools across different datasets, error rates, and outlier degrees. It maintains its efficiency, consistently requiring significantly less time than its counterparts, even as error rates and outlier degrees vary. Finally, in terms of scalability, SAGED shines, achieving impressive F1 scores and significantly shorter detection times than other tools in large-scale datasets. These findings collectively demonstrate SAGED's strong potential as a reliable and efficient error detection tool for a wide range of data-driven tasks.

While SAGED offers notable advantages and performance, it exhibits certain considerations: (1) It relies on the availability of pre-cleaned historical data with similar error profiles. Nevertheless, in real-world scenarios, obtaining such historical data might not always be problematic. (2) SAGED extracts insights and patterns from base models trained on historical data. The generalization of these patterns to new, unseen errors might be limited, potentially leading to reduced accuracy in scenarios with different error patterns. In this case, it is highly recommended to continuously update the inventory of historical datasets to keep track of new error patterns. (3) SAGED's reliance on predictions generated by base pre-trained models and meta-classifiers may result in a lack of interpretability, making it challenging to explain why specific errors are detected or missed. However, employing post-hoc interpretability techniques, e.g., feature importance analysis or model visualization, can provide insights into the decision-making process. (4) Finally, the use of historical data raises concerns about data privacy and security, especially if the historical data contains sensitive information. Adequate precautions and data anonymization measures would be necessary to address these concerns.

6 RELATED WORK

In this section, we provide an overview of state-of-the-art error detection tools from academia and industry. We cover tools based on LLMs, rule-based tools, ML-based tools, and ensemble methods [1]. Several recent proposals have explored the utilization of Large Language Models (LLMs) in data management tasks, including data cleaning and data integration. For instance, Narayan et al. [27] applied a prompting approach to address error detection and demonstrated that augmenting prompts with examples, known as a few-shot approach, allows LLM models to outperform ML-based error detection tools. Additionally, Vos et al. [39] introduced prefix-tuning as an alternative to fine-tuning when employing LLMs for data-wrangling tasks. Another LLM-based data cleaning method, RetClean [4], has been developed to enhance the output of ChatGPT with knowledge from a user-provided data lake. Despite the potential benefits of LLMs in data management tasks, their adoption is still in its infancy and poses significant challenges due to their domain specificity, data privacy concerns, and resource-intensive nature.



(a) Beers (F1 Score) **(b) Nasa (RMSE)** **(c) Smart Factory (F1 Score)**

Figure 16: Modeling accuracy of a neural network trained on the ground truth and repaired data. The blue region represents the accuracy when using the repaired versions as input to the neural network, while the green region represents the accuracy when using the ground truth version of the dataset. The abbreviations represent SAGED as SG2 and the baseline tools: E2 for ED2, K2 for KATARA, R2 for RAHA, and B2 for dBoost.

Aside from LLMs, the rule-based detection methods, e.g., HoloClean [32] and NADEEF [10], tackle rule violation errors by enforcing functional dependencies and integrity constraints. Pattern enforcement and transformation methods, such as OpenRefine [16], identify syntactic and semantic patterns in the data to detect inconsistencies effectively. Quantitative error detection algorithms, including dBoost [25], leverage statistical techniques like histograms, Gaussian, and multivariate Gaussian mixtures to detect outliers that deviate from the statistical distribution of the dataset. Record linkage and de-duplication methods, such as Data Tamr system [36], focus on entity consolidation when multiple samples correspond to the same entity. It is noteworthy that the mentioned methods do not employ machine learning for error detection or data repair.

To address the heterogeneous dirtiness profiles encountered in real-world datasets, holistic ML-based error detection methods have been developed, treating error detection as a classification task. For example, RAHA [24] employs simple error detection methods to generate feature vectors, followed by training detection classifiers. Similarly, metadata-driven error detection [38] extracts metadata to guide the training process. To enhance detection performance, various methods, such as ED2 [28], Picket [23], and HoloDetect [18], model attribute-level, tuple-level, and dataset-level features describing the underlying distribution of a dataset. These methods adopt different labeling strategies: RAHA clusters samples by similarity and acquires labels per cluster, propagating them within each cluster. ED2 utilizes active learning to acquire labels for samples the model is uncertain about. HoloDetect reduces the labeling budget by generating synthetic erroneous samples based on learned error patterns. Picket relies on self-supervision to avoid the need for user labels.

The state-of-the-art error detection tools face several challenges that limit their effectiveness and adaptability: (1) the reliance on expert knowledge to provide cleaning signals, such as functional dependencies, creates a barrier to entry and depends heavily on the expertise of the data specialists. The accuracy of these methods is contingent on the thoroughness and quality of the input configurations. (2) simpler error detection algorithms often miss a wide range of errors due to their focus on specific types, leading to a low overall recall rate. (3) ML-based approaches typically do not incorporate contextual or historical data, which could otherwise enhance the precision of the error detection process. Finally, ML-based methods can suffer from long execution times, rendering them less feasible for use with larger datasets, as seen with tools like Picket and RAHA, which can be especially slow when generating cleaning strategies. SAGED distinguishes itself from existing tools by forgoing the need for an array of detection methods and the associated pre-configurations. Instead, it capitalizes on the statistical and integrity characteristics intrinsic to the data to craft feature vectors that encapsulate the data's statistical distribution. This strategy simplifies the preparatory process and mitigates the need for tuning configurations. Moreover, SAGED employs meta-learning to draw upon the insights gained from cleaning historical datasets within the same or analogous projects. This knowledge transfer facilitates a more comprehensive identification of error patterns in new datasets, improving SAGED's error detection capability even when operating under constrained labeling resources.

7 CONCLUSION & OUTLOOK

In this study, we present SAGED, a meta-learning-based error detection tool designed for tabular data. SAGED leverages historical datasets to gain valuable insights into error profiles in the given datasets. The tool employs two phases, knowledge extraction, and detection, to achieve this goal. Extensive evaluation against various baseline methods demonstrates SAGED's superior performance in terms of detection accuracy and time efficiency. Moreover, SAGED exhibits scalability with large datasets and can handle diverse error rates and outlier degrees. Future work involves extending SAGED by integrating a context modeling tool, such as RTClean [14], to update the models when the surrounding context changes. Additionally, innovative error repair approaches like AutoCure [2] and BARAN [24] will be incorporated to enhance the detected error repair capabilities.

ACKNOWLEDGMENTS

This work was supported (in part) by the Federal Ministry of Education and Research through grants 02L19C155, 01IS21021A (ITEA project number 20219).

REFERENCES

[1] Mohamed Abdelaal, Christian Hammacher, and Harald Schoening. 2023. REIN: A Comprehensive Benchmark Framework for Data Cleaning Methods in ML Pipelines. In *26th International Conference on Extending Database Technology (EDBT)*. https://arxiv.org/abs/2302.04702

[2] Mohamed Abdelaal, Rashmi Koparde, and Harald Schoening. 2023. AutoCure: Automated Tabular Data Curation Technique for ML Pipelines. In *Proceedings of the Sixth International Workshop on Exploiting Artificial Intelligence Techniques for Data Management in conjunction with SIGMOD 2023*. 1–11.

[3] Ziawasch Abedjan, Xu Chu, Dong Deng, Raul Castro Fernandez, Ihab F Ilyas, Mourad Ouzzani, Paolo Papotti, Michael Stonebraker, and Nan Tang. 2016. Detecting data errors: Where are we and what needs to be done? *Proceedings of the VLDB Endowment* 9, 12 (2016), 993–1004.

[4] Mohammad Shahmeer Ahmad, Zan Ahmad Naeem, Mohamed Eltabakh, Mourad Ouzzani, and Nan Tang. 2023. RetClean: Retrieval-Based Data Cleaning Using Foundation Models and Data Lakes. *arXiv preprint arXiv:2303.16909* (2023).

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

[6] Patricia C Arocena, Boris Glavic, Giansalvatore Mecca, Renée J Miller, Paolo Papotti, and Donatello Santoro. 2015. Messing up with BART: error generation for evaluating data-cleaning algorithms. *Proceedings of the VLDB Endowment* 9, 2 (2015), 36–47.

[7] Oliver Birgelen, Alexander; Niggemann. 2018. Smart Factory: High Storage System Data for Energy Optimization. https://www.kaggle.com/inIT-OWL/high-storage-system-data-for-energy-optimization accessed on January 2022.

[8] Olivier Chapelle, Jason Weston, and Bernhard Schölkopf. 2002. Cluster kernels for semi-supervised learning. *Advances in neural information processing systems* 15 (2002).

[9] Xu Chu, John Morcos, Ihab F Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. 2015. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 1247–1261.

[10] Michele Dallachiesa, Amr Ebaid, Ahmed Eldawy, Ahmed Elmagarmid, Ihab F Ilyas, Mourad Ouzzani, and Nan Tang. 2013. NADEEF: a commodity data cleaning system. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. 541–552.

[11] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[12] Hadi Fanaee-T and Joao Gama. 2013. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence* (2013), 1–15. https://doi.org/10.1007/s13748-013-0040-3

[13] D. Foroni, M. Lissandrini, and Y. Velegrakis. 2021. Estimating the extent of the effects of Data Quality through Observations. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE Computer Society, Los Alamitos, CA, USA, 1913–1918. https://doi.org/10.1109/ICDE51399.2021.00176

[14] Daniel Del Gaudio, Tim Schubert, and Mohamed Abdelaal. 2023. RTClean: Context-aware Tabular Data Cleaning using Real-time OFDs. In *2023 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. IEEE. https://arxiv.org/abs/2302.04726

[15] Yoav Goldberg and Omer Levy. 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722* (2014).

[16] Kelli Ham. 2013. OpenRefine (version 2.5). http://openrefine. org. Free, open-source tool for cleaning and transforming data. *Journal of the Medical Library Association: JMLA* 101, 3 (2013), 233.

[17] Mahdi Hashemi. 2019. Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation. *Journal of Big Data* 6, 1 (2019), 1–13.

[18] Alireza Heidari, Joshua McGrath, Ihab F Ilyas, and Theodoros Rekatsinas. 2019. Holodetect: Few-shot learning for error detection. In *Proceedings of the 2019 International Conference on Management of Data*. 829–846.

[19] Jean-Nicholas Hould. 2017. *Craft Beers Dataset*. https://www.kaggle.com/nickhould/craft-cans Accessed on February 2021.

[20] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gurel, Bo Li, Ce Zhang, Costas J Spanos, and Dawn Song. 2019. Efficient task-specific data valuation for nearest neighbor algorithms. *arXiv preprint arXiv:1908.08619* (2019).

[21] Ron Kohavi et al. 1996. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid.. In *Kdd*, Vol. 96. 202–207.

[22] Pradap Konda, Sanjib Das, AnHai Doan, Adel Ardalan, Jeffrey R Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeff Naughton, Shishir Prasad, et al. 2016. Magellan: toward building entity matching management systems over data science stacks. *Proceedings of the VLDB Endowment* 9, 13 (2016), 1581–1584.

[23] Zifan Liu, Zhechun Zhou, and Theodoros Rekatsinas. 2020. Picket: Guarding Against Corrupted Data in Tabular Data during Learning and Inference. *arXiv preprint arXiv:2006.04730* (2020).

[24] Mohammad Mahdavi, Ziawasch Abedjan, Raul Castro Fernandez, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2019. Raha: A configuration-free error detection system. In *Proceedings of the 2019 International Conference on Management of Data*. 865–882.

[25] Zelda Mariet, Rachael Harding, Sam Madden, et al. 2016. Outlier detection in heterogeneous datasets using automatic tuple expansion.

[26] Hugo Mathien. 2016. European Soccer Database. https://www.kaggle.com/hugomathien/soccer accessed on January 2022.

[27] Avanika Narayan, Ines Chami, Laurel Orr, Simran Arora, and Christopher Ré. 2022. Can foundation models wrangle your data? *arXiv preprint arXiv:2205.09911* (2022).

[28] Felix Neutatz, Mohammad Mahdavi, and Ziawasch Abedjan. 2019. ED2: A case for active learning in error detection. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 2249–2252.

[29] Mourad Ouzzani, Hossam Hammady, Zbys Fedorowicz, and Ahmed Elmagarmid. 2016. Rayyan—a web and mobile app for systematic reviews. *Systematic reviews* 5 (2016), 1–10.

[30] Dessislava Petrova-Antonova and Rumyana Tancheva. 2020. Data cleaning: A case study with OpenRefine and Trifacta Wrangler. In *Quality of Information and Communications Technology: 13th International Conference, QUATIC 2020, Faro, Portugal, September 9–11, 2020, Proceedings 13*. Springer, 32–40.

[31] Abdulhakim A Qahtan, Ahmed Elmagarmid, Raul Castro Fernandez, Mourad Ouzzani, and Nan Tang. 2018. FAHES: A robust disguised missing values detector. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2100–2109.

[32] Theodoros Rekatsinas, Xu Chu, Ihab F Ilyas, and Christopher Ré. 2017. Holoclean: Holistic data repairs with probabilistic inference. *arXiv preprint arXiv:1702.00820* (2017).

[33] Felix M. Riese and Sina Keller. 2018. Introducing a Framework of Self-Organizing Maps for Regression of Soil Moisture with Hyperspectral Data. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*. Valencia, Spain, 6151–6154. https://doi.org/10.1109/IGARSS.2018.8517812

[34] Thomas Roelleke and Jun Wang. 2008. Tf-idf uncovered: a study of theories and probabilities. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. 435–442.

[35] Statista. 2016. *What significance does the gathering, analysis, and utilization of data for decision-making have for your company?* https://www.statista.com/statistics/549678/worldwide-survey-significance-of-data-by-industry/

[36] Michael Stonebraker, Daniel Bruckner, Ihab F Ilyas, George Beskales, Mitch Cherniack, Stanley B Zdonik, Alexander Pagan, and Shan Xu. 2013. Data curation at scale: the data tamer system.. In *Cidr*, Vol. 2013.

[37] D. Stuart Pope Thomas F. Brooks and Michael A. Marcolini. 2014. Nasa Airfoil Self-Noise Dataset. https://archive.ics.uci.edu/ml/datasets/airfoil+self-noise# accessed on January 2022.

[38] Larysa Visengeriyeva and Ziawasch Abedjan. 2018. Metadata-driven error detection. In *Proceedings of the 30th International Conference on Scientific and Statistical Database Management*. 1–12.

[39] David Vos, Till Döhmen, and Sebastian Schelter. 2022. Towards Parameter-Efficient Automation of Data Wrangling Tasks with Prefix-Tuning. In *NeurIPS 2022 First Table Representation Workshop*.

[40] Ji Zhang. 2013. Advancements of outlier detection: A survey. *ICST Transactions on Scalable Information Systems* 13, 1 (2013), 1–26.