# Record Fusion via Inference and Data Augmentation

ALIREZA HEIDARI, Department of Computer Science, University of Waterloo, Canada
GEORGE MICHALOPOULOS, Department of Computer Science, University of Waterloo, Canada
IHAB F. ILYAS, Department of Computer Science, University of Waterloo, Canada
THEODOROS REKATSINAS, Department of Computer Science, ETH, Switzerland

We introduce a learning framework for the problem of unifying conflicting data in multiple records referring to the same entity, we call this problem "record fusion". Record fusion generalizes two known problems:"data fusion" and "golden record". Our approach expresses record fusion as a learning problem over probabilistic models. In contrast to prior approaches, our method achieves high performance with or without the records source information, and outperforms state-of-art baselines. Furthermore, we show how our learned fusion model can solve the problem of scarcity of training data. On multiple data sets, we show that our framework fuses records with an average precision of ∼98% when source information is available, and ∼94% without source information across a diverse array of datasets. We compare our approach to a comprehensive collection of data fusion and entity consolidation methods, ranging from source information related methods to approaches that do not need any source information. We show that our approach can achieve an average improvement of ∼ 20/∼ 45 precision points with/without source information. Our data augmentation method improves previous approaches an average of ∼ 10 precision points.

## 1 INTRODUCTION

Diverse information sources often provide contradictory data, some being out-of-date, inexact, or incorrect. The increasing demand to ingest and acquire a large number of heterogeneous data sources via inexpensive connectors has been the driving force behind many studies in the field of entity consolidation and data integration [3, 13–15, 21, 29, 30, 42]. An integrated dataset can have a considerable positive impact on the quality of downstream analytics [5]. For the rest of the paper, we will refer to aggregating multiple records into a unified representation by "Record Fusion".

A good example of record fusion is *the golden record problem*; after deduplicating a dataset, a set of record clusters is created, which needs to be consolidated to produce one representation [6, 10, 17]. Another example is *data fusion*, which sometimes referred to as "Single-Choice Tasks" in crowdsource community, where the task is to aggregate multiple sources that contain information about the same set of entities, with possibly conflicting attribute values [13, 15, 30, 42, 44].

Example 1.1. *In Table 1, the data have been gathered from 4 different databases. Each database can be viewed as a source that makes a claim about the actual value of real-world entities. In this example, there are three clusters, where each tuple is generated from a unique source in the cluster. The correct values of each cluster/attribute are shown in* ***Bold****. However, pieces of information that are gathered from different sources can be conflicting. For instance, the first source of cluster* $c_3$ *claims that the Amazon headquarters is located in the state "WA", while, the last source claims that the state of the entity is "New York". In the record fusion problem, we want to resolve such conflicts and obtain the correct values for the attributes of each real-world entity.*

Many probabilistic fusion methods depend on the availability of the records source information as part of the schema, and often build models to estimate the "trustworthiness" of the sources [1, 8, 9, 13, 16, 25, 36, 39–43, 45]. However, calculating the trustworthiness of a source is not trivial and can significantly vary in different parts of the data. For example, in some instances, limited ground truth is used for calculating an initial estimation of the sources "trust score"[12, 19]. In other scenarios, all sources might be equally good/bad and these source-dependant methods are reduced to simple majority voting and fail to take other effective signals into account. In more

| Cluster Id | Company | ZIP | City | State | Webpage |
|---|---|---|---|---|---|
| $c_1$ | Google | **94043** | **Mountain View** | **CA** | google.ca |
| $c_1$ | **Google Inc.** | 940 43 | Mountains View | California | **Google.com** |
| $c_2$ | Microsoft | **98052** | **Redmond** | **WA** | MS.com |
| $c_2$ | MICROSOFT | **98052** | Seattle | Washington | **msn.com** |
| $c_2$ | **Microsoft Corp.** | 56419 | **Redmond** | **WA** | **microsoft.com** |
| $c_3$ | Amazon | *<Null>* | **Seattle** | **WA** | amazon.co |
| $c_3$ | **Amazon Inc.** | **98109** | **Seattle** | *<Null>* | amazon.ca |
| $c_3$ | AMZN INC | **98109** | **Seattle** | Washington | **amazon.com** |
| $c_3$ | Amazon | 98052 | *<Null>* | New York | **amazon.com** |

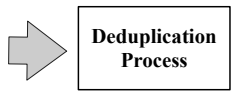
Fig. 1. An example of clusters with conflicting values for each cluster-attribute.

complex scenarios, source quality can vary for subsets of data from that source; for example, a certain US website can be more accurate for data in the US and low accurate for international data records. Furthermore, in many real-world scenarios, we may not have any explicit source information altogether, e.g., in deduplicating a single-source dataset as we show in the following example.

Example 1.2. *Figure 2 shows a tabular data set containing tuple id's, person names, occupation, and address and illustrates the task of a typical deduplication approach. A deduplication technique produces a table with the clustering of those records, where each cluster refers to the same real-world entities. The process of finding the true records for entities is called the "golden record problem". In this setting, there is no information about sources, as all records might have come from the same source.*

**Input**

| ID | Name | Occupation | Zip |
|---|---|---|---|
| $t_1$ | Bob Adams | Student | N2L 3G1 |
| $t_2$ | L. Sims | Manager | 38241 |
| $t_3$ | Bob Adams | Intern | 10011 |
| $t_4$ | Alice J. | Student | 94309 |
| $t_5$ | Lisa Sims | Manager | 38 241 |
| $t_6$ | Alice J. | Professor | 53706 |
| $t_7$ | B. Adams | Intern | 10-011 |




**Output**

| | ID | Name | Occupation | Zip |
|---|---|---|---|---|
| $c_1$ | $t_1$ | Bob Adams | Student | N2L 3G1 |
| | $t_3$ | Bob Adams | Intern | 10011 |
| | $t_7$ | B. Adams | Intern | 10-011 |
| $c_2$ | $t_2$ | L. Sims | Manager | 38241 |
| | $t_5$ | Lisa Sims | Manager | 38 241 |
| $c_3$ | $t_4$ | Alice J. | Student | 94309 |
| | $t_6$ | Alice J. | Professor | 53706 |

Fig. 2. A typical deduplication task.

In the golden record problem, the concept of "source" might not be even applicable since the data from one source can have duplicates. Current approaches either use techniques such as heuristic aggregation rules, majority vote or involve human annotators to resolve conflicts via learned transformations [10]. In this case, We show experimentally that naïve methods like choosing the value provided by the majority vote of sources often lead to inaccurate and unreliable results (see Section 6.2). Human-in-the-loop approaches have multiple challenges, including (1) they can be prohibitively expensive and time-consuming depending on the number of clusters, and difficult to resolve conflicts; (2) they often assume users do not make mistakes, or they need to involve multiple voters introducing new types of conflicts that need additional resolution mechanisms;

and (3) a global process to aggregate human decisions in a consistent way may still be needed. For example, when the human votes are conflicting, the challenge to conform their suggestions and select the correct vote still remains. In summary, the common challenge between "data fusion" and "golden record" is fusing multiple records into a smaller set of true records, which we call it "record fusion".

Recently, multiple principled machine learning (ML) approaches have been proposed for the data cleaning problems [7, 22, 37]. In this paper, we show that an ML inference model that takes into account all available signals (statistical properties and integrity constraints) can solve the record fusion problem. The problem shares some technical challenges with these ML-cleaning approaches. These include: (1) designing a rich representation to capture the structure of the data and all the signals needed to impute the fused values; and (2) learning from noisy and partial information. While we show how to solve these challenges in the clustered data settings, an additional new challenge that we also address in this paper is generating enough "cluster" training data (via data augmentation) to train the rich representation model.

### 1.1 Approach and Technical Challenges

We propose a learning framework for record fusion based on a weak supervision approach [35]; our framework automatically fuses records in the clusters by leveraging all related information, i.e., integrity constraints and data rules, quantitative statistics, and source information if available. Our ML-approach addresses multiple technical challenges:

**I [Model]** To infer the correct values, an expressive model is required to capture all data characteristics. Specifically, the representation models should cover all data context features that describe the distribution governing the generation of the dataset.

**II [Training Data]** An expressive model needs enough training data to achieve an acceptable level of confidence. However, gathering enough labeled examples is a costly, error-prone, and sometimes impossible process. In addition, each training example is a cluster of records, which needs a different labeling mechanism. Hence, we need a way to automatically generate additional training data from the limited ground truth available.

**III [Partial Information]** For designing a reliable ML-approach, we need to have complete information to generate appropriate features. Labeled data gives us limited observations of the whole truth. The model needs to learn from partial and noisy estimates of the correct values and an appropriate mechanism to improve prediction iteratively.

### 1.2 Contributions and Organization

We address the aforementioned challenges and propose an ML-based framework for record fusion. The framework iterates over multiple modules to learn the representation of the input data, generate enough training data to learn the model parameters, and produce a probabilistic representation of the final fused record (Section 2.2). We highlight the following concrete contributions:

**(1)** We introduce representation models that capture various characteristics of the data, with or without source information, providing a rich input to the inference model for predicting the values of the fused record (Section 3). **(2)** We introduce a novel generative data augmentation process, which generates additional artificial *training data clusters* for learning the model parameters. This process resolves the problem of limited training data (Section 4). **(3)** To address the partial information challenge, we introduce an iterative mechanism that works as a *Hard-Assignment Expectation Maximization* approximation scheme to estimate model parameters and provide robust estimates (Section 5).

Finally, we evaluate our framework on multiple real-world datasets, where we demonstrate its ability to determine the correct values for real-world entities, and we show that probabilistic

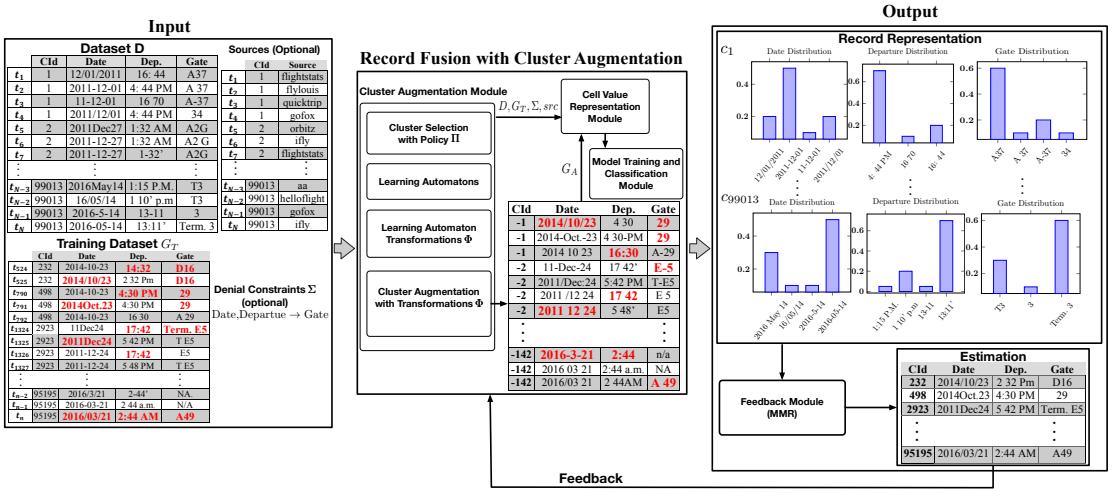



Fig. 3. Overview of Record Fusion system. The system can learn from heterogeneous formats of correct data. For example, the correct Date is shown in different formats and the inference model still can approximate the correct record representations.

inferences with sufficient training data are a valid modeling tool for the record fusion problem (Section 6). We highlight relevant previous works in Section 7, and conclude with final remarks in Section 8.

## 2 PROBLEM DEFINITION AND SOLUTION OVERVIEW

We define "Record Fusion" as the process that takes a collection of clusters as input, where each cluster contains a set of tuples and outputs a record representation for each cluster. The record representation follows a probabilistic semantics that we formally describe in the following section.

### 2.1 Problem Definition

The input is a relational dataset $D$, with a clustering $C$ that defines partitions over tuples in $D$. $D$ has a schema $S = R \cup \{cid\}$, where $R = \{r_1, r_2, \ldots, r_N\}$ denotes attributes of $D$, and $cid$ identifies the cluster of each tuple. Let Cluster $c_i \in D$ denotes the set of tuples in $D$ with the same $cid = i$, and $c_i(r_j)$ denotes the set of unique values of Attribute $r_j \in R$ in Cluster $c_i \in D$.

Let $G$ be a data set with the same schema as $D$ that provides a probabilistic representation of the clusters in $D$. For each Tuple $g_i \in G$ that corresponds to Cluster $c_i \in D$ and Attribute $r_j \in R$, $g_i.r_j$ is a probability distribution $V^*_{c_i,r_j}$ over the values $c_i(r_j)$. Finally, with $D$, we are also provided with a training dataset $G_T \subset G$. $V^*_{c_i,r_j}$ is given for each Attribute $r_j \in R$ and every Tuple $t_i \in G_T$. Optionally, a set of data rules expressed as denial constraints $\Sigma$ are provided. We define $G_U = G \backslash G_T$ as the set of unknown true record representations. The problem definition is stated as follows:

Definition 1 (Record Fusion).
*For each Cluster $c_i \in D$, Record Fusion produces a unified representation tuple $g_i = (f_{i,1}, f_{i,2}, \ldots, f_{i,N})$, where $f_{i,j}$ is a probability distribution over values $c_i(r_j)$. Specifically, $f_{i,j}$ is an estimator for the ground truth $V^*_{c_i,r_j}$ that minimizes a given distance function $\Delta(f_{i,j}, V^*_{c_i,r_j})$ conditioned on the observed input data $D$, the partial ground truth $G_T$, and the set of data rules $\Sigma$ (if given).*

Since the record fusion provides a probability distribution over possible values in each attribute of each cluster, there are multiple ways to consume it. One way is to provide the most likely value(s) of each attribute in each cluster as our best approximation for the golden record problem.

Definition 2 (Marginal MAP Record (MMR)).
*Given the output of Record Fusion (Def. 1) for a cluster $c_i \in D$ and a record representation $g_i = (f_{i,1}, f_{i,2}, \ldots, f_{i,N})$; $\hat{g}_i = (\hat{v}_{i,1}, \hat{v}_{i,2}, \ldots, \hat{v}_{i,N})$ can be produced, where $\hat{v}_{i,j}$ is the maximum likelihood values from the distribution $f_{i,j}$.*

Note that if the partial ground truth $G_T \subset G$ was provided as a set of correct values for each attribute and tuple in $G_T$, we can induce a uniform distribution over these values. Note also that one of the attributes $r_i \in R$ can correspond to the source of records in $D$.

## 2.2 Solution Overview

We now present the architecture of our framework. These modules are used (1) to learn a generative process to produce more training data by adding artificial clusters; and (2) use an iterative approach to learn a representation model of signals jointly with a classifier that is used to find record representations of the input dataset. Our framework takes as input a noisy dataset $D$, a training data $G_T$, (optionally) a set of denial constraints $\Sigma$, and (optionally) source information. The four core modules are described in the following:

**Module 1: Cell Value Representation** This module combines different properties and signals to capture the distribution of true record representation, which maps each cell in $D$ to a fixed-dimension real-valued vector. To obtain this vector, we concentrate on the output of different representation models, each of which targets a specific context (i.e., attribute, tuple, or dataset context) (see Section 3.1).

**Module 2: Model Training and Classification** This module is responsible for training a multi-class classifier, which given the representation of cells of a cluster, generates a distribution with the size of the distinct values of each attribute of clusters (see Section 3.2).

**Module 3: Cluster Augmentation** Due to the need for training data, this module learns a generative process to generate additional training data by sampling and transforming the correct representation records, $G_T$. The output of this module is a set of additional clusters $G_A$ that are legitimate to be used for training fusion model parameters. This module generally can be used for any fusion algorithm that uses statistical properties of data. (see Section 4).

**Module 4: Feedback** Due to learning with noisy and partial information (correct predictions of other attributes of cluster), we perform multiple iterations of a *Hard-Assignment Expectation Maximization* process [27] (see Section 5).

An overview of how the different modules are connected is shown in Figure 3. First, Module 1 augments training data with additional artificial clusters. Then, Module 2 grounds the representation model of our record fusion model. Subsequently, the representation model is connected with the multi-class classifier model in Module 3. After generating a record representation, the model gets feedback from Module 4, and so it can modify the representation and the predictions.

## 3 REPRESENTATION OF DATA CELLS

From Definition 1, given $D$ and the training data $G_T \subset G$, we want to estimate the probability distribution $V^*_{c_i,r_j}$ for each cell $g_{i,j} \in G \setminus G_T$. To estimate $V^*_{c_i,r_j}$, we learn a representation model $Q$ that approximates data distribution properties on the attribute, tuple, and cluster level, and couple them with a prediction model $M$ that maps a value representation based on $Q$ to a probability distribution. Model $Q$ builds upon a variety of features computed either based on the initially observed values (static features) or based on intermediate inferences of the true cluster values (dynamic features). We require $Q$ to be such that the likelihood of true record representations in $G_T$ given $Q$ to be maximum, while the likelihood of other record representations be low. This property is necessary for a model $M$ to assign high-probability to correct cluster values and low-probability to incorrect cluster values. We rely on representation learning techniques to learn $Q$ jointly with model $M$.

### 3.1 Representation Models

The representation model $Q$ needs to capture the statistical characteristics of cells inside each cluster in attribute-level, tuple-level, and cluster-level contexts. $Q$ is formed by concatenating the outputs of different models and signals, and maps each cell $D_{t_i,r_j}$ of tuple $t_i$ and attribute $r_j$ to a fixed-dimension real-valued vector $h_{t,r} \in \mathbb{R}^d$ with dimension $d$. Next, we review the representation models used in each of these three contexts. It should be noted that the introduced models correspond to a bare-bone set that captures all settings and is currently implemented in our prototype. Our architecture can trivially accommodate additional models (e.g. auto-encoders) or more expressive variants of the current representation models.

**Attribute-level Representation** Models for this context capture the distributions governing the values and format for an attribute. For example, the value "New Yorx" in attribute *City* has a low frequency and with respect to language models, also obtains low scores. Separate models are used for each attribute $r_i \in R$ in dataset $D$. We consider three types of models: (1) *Format model*, which captures the probability distribution governing the format of the values. In our implementation, we consider an n-gram model for each attribute that captures the format sequence over the cell value (empirically, we set $n = 2$). Each n-gram is associated with a probability learned directly from the dataset by using a reduced model (for example we replace all the letters with $A$, all the numbers with $N$, special characters represent themselves, etc.). For each value $c_i(r_j)$, we calculate a probability by taking the product of the likelihood of each n-gram constructed from the reduced model string value. (2) *Running cluster-value feature*, as our model uses an iterative algorithm, this signal captures the compatibility of the attribute value in a record with the running cluster value in $c_i(r_j)$ (after inference). For each value in $c_i(r_j)$, we have a flag that indicates if it was the predicted value in an earlier iteration. In the first iteration, the current Running cluster-value feature is set to zero for each value in $c_i(r_j)$ to prevent any bias to the learning parameters. (3) *Character and token sequence* is a distance metric on the character-based representation of each value in $D$ with the average representation of all the values for each entity. We train a word-embedding model for each attribute, where each column in the dataset $D$ is considered to be a corpus and each possible value a sentence. We decompose each possible value into its respective characters, and we learn an embedding vector for each value. Furthermore, after the model is trained for each attribute of a cluster, we compute the average embedding vectors of Attribute $r_p$ in Cluster $c_q$:

$$avg_{c_q(r_p)} = \frac{\sum_{j \in c_q(r_p)} embedding_j}{|c_q(r_p)|}. \quad (1)$$

Finally, we calculate the distance of the embedding vector of each value from the average embedding vector using the following equation, $h_i = -2 \times \cos(embedding_i, avg_{c_q(r_p)}) + 1$ and if $h_i > 0$ we set $h_i = h_i/3$ to normalize the feature value.

**Tuple-level Representation:** Models for this context capture the joint distribution of different attributes. These representations perform weak predictions for the possible values of each cell. For example, if the value "New York" in attribute *City* appeared more often with "United States" of attribute *Country*, the Tuple-level signals reveal this effect. We consider two types of models: (1) *Co-occurrence model* that captures the empirical joint distribution over pairs of attributes. As we have multiple initial values for each attribute, we use the values that are predicted in a previous iteration for each attribute in our calculations. This model is updated in every iteration. In the first iteration, we select the majority vote value as the initial value. Finally, it should be noted that we use one co-occurrence feature per pair of attributes. (2) *Vote model* that captures the empirical distribution of the values $c_i(r_j)$ of Attribute $r_j$ associated with Cluster $c_i$. These can be learned directly from the input dataset $D$. This representation is a scalar, which is the empirical probability of the values in each cluster.

**Dataset-level Representation:** Models for this context capture a distribution that governs the compatibility of tuples and values in the dataset $D$. Specifically, any functional dependency that is valid on our correct records, Dataset-level signals capture this data integrity rule. We consider three types of models: (1) *Neighborhood-based representation* that is a distance metric on the neighborhood-based representation of each value in $D$ with the average representation of all the values for each entity. We train a word-embedding model, where the input dataset $D$ is considered to be a document, and each tuple in $D$ is a sentence. As we mentioned in *character and token sequence* signal, after the model is trained for each attribute of a cluster, we calculate the average embedding vector of all the values for this specific attribute-cluster pair. Finally, we can calculate the distance of the embedding vector of each value from the average embedding vector by using the equation 1. In our dataset, if the source information is given we can follow the same procedure. The semantic of the source information signal is the level of irregularity of the given information of a source. (2) *Source model* captures the source from which each value in $c_i(r_j)$ came from. Specifically, for each value, we have an indicator that specifies the source that it came from. This signal is available if the source information is provided. (3) *Constraint-based models* capture the number of violations of every possible value with the predicted values that compose the current cluster values table. For each denial constraint in $\Sigma$ (if given), we create the equivalent relaxed denial constraints [31, 37], and for each random variable (pair of cluster-attribute) and values in $c_i(r_j)$, we have one feature per denial constraint that captures the number of violations. As we have multiple initial values, we use the values that are predicted in a previous iteration for each random variable in our calculations. This model is updated in every iteration, and for the first iteration, we consider the most frequent value as the initial value. The outputs of all signals and representations are concatenated into a single vector that is given as input to classifier $M$. To achieve a high-quality record fusion model, features from all contexts need to be combined to form model $Q$.

### 3.2 Record Representation Distributions

The classifier $M$ uses a dense layer followed by *Softmax* to find record representations for random variables in the dataset $D$. In this setting, for each attribute $r_j \in R$ and cluster $c_i \in D$, we consider a random variable $v_{c_i,r_j}$ which takes values from $c_i(r_j)$. Let $Z$ denotes random variables for attributes in $R$ and clusters in $D \cup D_A$ where $D_A$ is an additional artificial cluster that is generated by the augmentation module (Section 4). The training process sets the dense layer parameters $\theta$ to values that maximize the likelihood, $\mathcal{L}(\theta) = \log P(G_T \cup G_A | Z; \theta)$, where $G_A$ is the set of augmented clusters (see Section 4). The model uses random variables that are observed in the training data $G_T$

and augmented training data $G_A$ as labeled examples to learn the parameters $\theta$. After obtaining the record representations, the probabilistic inference is used to estimate the true values of tuples in $G_U$. To this end, we rely on an Expectation-Maximization (EM) algorithm to learn the distribution (given the unobserved random variables). To perform EM we set the unobserved cluster values to their maximum a posteriori (MAP) values.

More importantly, we calibrate the confidence of the predictions of $M$ using *Platt Scaling* [20, 34] in order to remove the false prior effect of the artificial training clusters that we used as training data. Specifically, an additional linear layer is added after *Softmax* with size equal to the maximum random variables domain, $\max_{c_i \in D, r_j \in R} |c_i(r_j)|$. The parameters of this layer are learned by using data from the original distribution, $G_T$ after $\theta$ have been learned and fixed. In prediction, we use all layers as an end-to-end entity.

## 4 DATA AUGMENTATION

We propose a data augmentation approach to increase the amount of training data available in the training phase of our model by learning a generative process. The augmentation process introduces new clusters in the dataset, by transforming an existing cluster to another cluster format of values. The statistical properties of Attribute $r$ in Cluster $c$ reflect signals like frequency, co-occurrence, and column-based language models. Thus, assuming that the generative process has fixed parameters during the cluster generation, we can observe an empirical distribution $\hat{I}_{c_i,r_j}$ for Cluster $c_i$ and Attribute $r_j$ in the original data. Using $\hat{I}_{c_i,r_j}$, we can generate a different cluster $c_i'$ that gives us a new format for the incorrect values in $c_i$, and $c_i'$ keeps all the statistical properties of $c_i$. This approach helps us discriminate between the actual distribution of records and other possible distributions of the cluster in more general forms. A format translation process that generates these new values is important for the performance as it biases the system to identify syntactic variations and does not focus only on instance-level variations. Hence, it promotes generalization. Our goal is to find a generative process that produces valid values based on a given Cluster $c_i$. Since changing the source authority introduces bias with format translation, we suggested a separate setup for generating source information attributes (see Section 4.4) and this approach applies to attributes other than source information.

**Algorithm 1:** Overview of Data Augmentation

**Input:** Original cluster $c_i$ with $g_i \in G_T$, policy $\Pi$
**Output:** Augmented cluster $c_i'$

1 Sample from $D \backslash c_i$ with policy $\Pi$ to get candidate format;
2 Learn format translation process $\Phi$;
3 Apply $\Phi$ on $c_i$ to produce $c_i'$;
4 [OPTIONAL] Add source information to the generated cluster

We present our augmentation policy $\Pi$ in Section 4.1. In Section 4.2, we discuss how to design a *format translator function*. Generating a cluster with a new format is described in Section 4.3. Section 4.4 explains how to include the source information attribute in augmented clusters.

### 4.1 Augmentation Policy

We pick a cluster following a sampling policy over the initial clusters and generate a new cluster of values with a new representation format. We have a set of policies $\Pi = \{\Pi_c, \Pi_{c_t}, \Pi_a, \Pi_v\}$, which supports the training process by selecting the clusters that in prediction have the lowest

confidence. The policy selects two clusters, an original cluster $c$ and a target cluster $c_t$, using the sampling policy $\Pi_c$ and $\Pi_{c_t}$ respectively. $\Pi_c$ selects the first cluster which is used to generate a new artificial cluster $c'$ from its values and $\Pi_{c_t}$ selects the cluster that the algorithm wants to capture its format for cluster $c$. As we said, we use the output of the running model $M$ to select the clusters with the lowest confidence in predictions. We select cluster $c_i$ with probability $1 - \min_{r_j \in R} \min_{v' \in c_i(r_j) \setminus \arg\max_{c_i(r_j)} M(v)} |M(v') - \max_{v \in c_i(r_j)} M(v)|$ which is the gap between the two most probable values, and it has a direct relationship to the model confidence. In the first iteration (Section 5), the model is not trained, so we select the original cluster($c$) randomly. For each value in $c$, a new $c_t$ is selected randomly, because we assume no prior information. $\Pi_a$ is the selecting policy of an attribute among the attributes of $c$. In this work, to prevent generating in convenient formats, we use the same attribute of the value currently is selected to change in $c$. $\Pi_v$ is the policy of choosing a value among distinct values in the selected attribute of $c_t$.

### 4.2 Format Transformation

To produce a translation function of string formats, we need to generalize string values to a more format-informative string and then learn a function that takes as input the original format and produces a string in the target format. We also need to determine a policy that will apply this translation to a corresponding string.

*4.2.1 Automaton Representation.* Each $v \in c_i(r_j)$ in cluster $c_i$ and attribute $r_j$ is considered as a string. It should be noted that many regular expressions (RE) can generate $v$.

Example 4.1. *All the following regular expressions parse the string 'aab'. $A_1 : a^* - b, A_2 : a - a^* - b$, $A_3 : a^* - a^* - b, A_4 : a - a - b, A_5 : a^* - a - b$. Also, we can replace $b$ with $b^*$, $b - b^*$ and $b^* - b$. In this model, we only considered two alphabets, and "+" operation was not used.*

Therefore, learning formats from strings is not a well-defined task. To solve this problem, we need an unambiguous grammar that for every valid string has a unique leftmost derivation or parse tree. We use a similar approach presented in [18], where we combine the elements that cannot be parsed, known as the *Glushkov automaton* [3]. For a language $L$ over alphabet $\Lambda$, we define: (1) $first(L) = \{b \in \Lambda | \exists w \in \Lambda^* : bw \in L\}$( where $^*$ determines kleene star). (2) $last(L) = \{b \in \Lambda | \exists w \in \Lambda^* : wb \in L\}$. (3) $follow(L, a) = \{b \in \Lambda | \exists v, w \in \Lambda^* : vabw \in L\}$. Let $Q_E$ be a finite set of states, $\Lambda$ be a finite set of alphabet symbols, $\delta_E$ be a transition function, $q_l \in Q_E$ determines the start state, and $F_E \subseteq Q_E$ be a set of accept states. The Glushkov automaton $G_E = (Q_E, \Lambda, \delta_E, q_l, F_E)$ has the following properties: (I) $Q_E = \mu(\Lambda) \cup \{q_l\}$, where $q_l \notin \mu(\Lambda)$ is the initial state. $\mu(.)$ puts marking index on element of a set e.g. $\mu((a|b)^*.a.(b^+.a)^*) = (a_1|b_1)^*.a_2.(b_2^+.a_3)^*$. (II) For $a \in \Lambda$, $\delta_E(q_l, a) = \{\chi \in first(\mu(E)) | \sigma(\chi) = a\}$. $\sigma(.)$ is the dropping of subscripts reverse of $\mu(.)$. (III) For $a \in \Lambda$, $\chi \in \mu(\chi)$, $\delta_E(\chi, a) = \{y \in follow(\mu(E), \chi) | \sigma(y) = a\}$. (IV) $F_E = last(\mu(E))$ is the set of final states; add $q_l$ to $F_E$ iff $\lambda \in L(E)$. We also know that the regular expression $L$ is unambiguous iff $G_E$ is deterministic. For all the types of finite automatons considered in this paper, we use notions like initial or final state as introduced for the Glushkov automatons.

*4.2.2 Learning Automaton.* Since we have a unique automaton for each string under the assumption of unambiguous regular expressions, we can learn formats from strings. The learning goal is to obtain a regular expression that represents the string's format. Let $\alpha = \{a, A, \ldots, z, Z\}$, $\beta = \{0, 1, \ldots, 8, 9\}$, and $C_s = \{?, !, \#, \$, \%, ;, :, space, \ldots\}$ be the set of special characters. We define the following sets $\mathcal{A} = \{w \in \alpha^+; |w| = 1\}$, $\mathcal{B} = \{w \in \alpha^+; |w| > 1\}$, $\mathcal{N} = \{w \in \beta^+; |w| = 1\}$, and $\mathcal{M} = \{w \in \beta^+; |w| > 1\}$. We assume $\Lambda = \{\mathcal{A}, \mathcal{B}, \mathcal{N}, \mathcal{M}, \} \cup C_s$ and $Q_E = \mu(\Lambda) \cup \{q_l\}$. We assume all examples generated from the Glushkov automaton with $\Lambda$ and $Q_E$.

**Prepossessing of training data** The example of the introduced model should conform with alphabet $\Lambda$, so when we get value $v$ in a cluster, we consider it as *string* and we apply the process $\tau : \left[[a-z]^*.[A-Z]^*.[0-9]^*.C_s\right]^+ \rightarrow \Lambda^+$ to translate the $v$ to a $\hat{v} = \tau(v) \in \Lambda^*$. This process is outlined in Algorithm 2. We keep the previous character $l_s$ and a Kleene star flag $f_{Kleene}$, and based on seeing more characters, numbers, or special characters, we determine the states. This process is called *blocking training examples*. For example string $s =$ "$New\ York - \#401H3$" blocks to $\tau(s) = \left[[New][\ ][York][-][\#][401][H][3]\right] = \mathcal{B}space\mathcal{B} - \#\mathcal{MAN}$.

**Algorithm 2:** String To States ($\tau$)

**Input:** String $v = s_0 s_1 s_2 \dots s_n$, Automaton Alphabet $\Lambda$, The Set of alphabet $\alpha$, The set of numbers $\beta$
**Output:** String $\hat{v} \in \Lambda^*$,Function $\tau_v : String \rightarrow \Lambda^*$

$l_s \leftarrow s_0$, $f_{Kleene} \leftarrow False$, and $\hat{v} \leftarrow \emptyset$;
**for** $s \in [s_1, s_2, \dots, s_n, \emptyset]$ **do**
  **if** $l_s \neq s$ **then**
    **if** $l_s \in \alpha \wedge f_{Kleene}$ **then**
      $\hat{v} \leftarrow \hat{v}.\mathcal{B}$
    **else if** $l_s \in \alpha \wedge \neg f_{Kleene}$ **then**
      $\hat{v} \leftarrow \hat{v}.\mathcal{A}$
    **else if** $l_s \in \beta \wedge f_{Kleene}$ **then**
      $\hat{v} \leftarrow \hat{v}.\mathcal{M}$
    **else if** $l_s \in \beta \wedge \neg f_{Kleene}$ **then**
      $\hat{v} \leftarrow \hat{v}.\mathcal{N}$
    **else**
      $\hat{v} \leftarrow \hat{v}.C_s$
    **end**
    $f_{Kleene} \leftarrow False$;
  **else**
    $f_{Kleene} \leftarrow True$;
  **end**
  $l_s \leftarrow s$;
**end**
**return** $\hat{v}, \tau_v$

**String matching** After translating the input string to state strings, we have only one halt state, so the generated regular expressions are linear. Using the linear property, we can consider the representation of the states of the automaton as strings. We generate a *state string* for each value. Informally, the similarity between state strings indicates that they have similar formats. Algorithm 3 performs a transition between the two-state strings. From Algorithm 3, we have a translation from automaton $A$ to the second automaton $A'$. We use this translation as a protocol to transform the corresponding strings that are generated from these automatons to each other.

*4.2.3 Transformation Policy.* Applying translations between machines might lose some string information because some parts of the original automaton string are not compatible with the target automaton format. On the other hand, we do not inject any information into the original automaton string, because we do not want to alter the original value distribution (adding symbols make bias in the target distribution, especially in the language models). Therefore, we normalize the transformations, by ignoring the transformations that inject information. This is equivalent to $r \mapsto r'; r \neq \emptyset$ at the index $pos$, so the set of these transformations makes a surjective function.

After we apply a transformation to automaton $A$, we apply the same transformation to the corresponding original string. The new string is a translation of the original string parsed with the target automaton (new format). If the string's new format is the same as before we reject that and repeat the process with another value. Note that the source of the new formats can be given as input by the domain expert, in that case, $\Pi_{c_t}$, $\Pi_a$ and $\Pi_v$ are useless, and we select new formats from external information provided.

**Algorithm 3:** Transformation Learning (TL)

**Input:** Automaton pair $e = (A, A')$, Alphabet $\Lambda$, The set of special characters $C_s$
**Output:** A list of valid transformations $\Phi_e$

**if** $A = \emptyset \wedge A' = \emptyset$ **return** $\emptyset$ ;
$l \leftarrow$ Longest Common Substring$(A, A')$;
**if** $l = \emptyset$ **then**
  **if** $(Length(A) \geq 2 \wedge \{\exists a \in A \wedge a \notin C_s\}) \vee (Length(A') \geq 2 \wedge \{\exists a \in A' \wedge a \notin C_s\})$ **then**
    $A_c \leftarrow replace_A(a, "C") : \forall a \in A \wedge a \notin C_s$;
    $A'_c \leftarrow replace_{A'}(a, "C") : \forall a \in A \wedge a \notin C_s$;
    $norm(A_c) \leftarrow replace_{A_c}(C^+, "C")$ and $norm(A'_c) \leftarrow replace_{A'_c}(C^+, "C")$;
    TL$(norm(A_c), norm(A'_c))$];
  **else**
    Add $[(A \longmapsto A', pos)]$ in $\Phi_e$ /*$pos$ shows the position of applying the transformation */;
  **end**
**else**
  $l_A, r_A \leftarrow A \setminus l$ /* Generate left and right substrings */;
  $l_{A'}, r_{A'} \leftarrow A' \setminus l$;
  **if** $similarity(l_A, l_{A'}) + similarity(r_A, r_{A'}) > similarity(l_A, r_{A'}) + similarity(l_{A'}, r_A)$ **then**
    TL$(l_A, l_{A'})$ and TL$(r_A, r_{A'})$;
  **else**
    TL$(l_A, r_{A'})$ and TL$(r_A, l_{A'})$;
  **end**
**end**
Remove all identity transformations from $\Phi_e$;
**return** $\Phi_e$

Formally, let string $s$ be the original string with a corresponding learned automaton (RE) $A_s$. Applying Algorithm 3 gives us $\Phi_{(A_s,A_t)}$ a set of transformations from $A_s$ to a target automaton $A_t$. We normalize the transformations and obtain $\Phi^{norm}_{(A_s,A_t)}$. If $A'_s = \Phi^{norm}_{(A_s,A_t)}(A_s)$ be the automaton after translation, since we already have $\tau_s$, the function that translates string to state, so we can calculate $s' = \tau_s^{-1}(A'_s)$, which is a new format of the string.

Example 4.2. *We have the string* $s =$ "$2016 - 04 - 12$", *so* $A_s = \tau_s(2016 - 04 - 12) = \mathcal{M} - \mathcal{M} - \mathcal{M}$ *and our target automaton is* $A_t : \mathcal{M}/\mathcal{M}/\mathcal{M}$ *then we obtain two transformations* $\Phi_{A_s \to A_t} = \{[- \to /, [5, 6)], [- \to /, [8, 9)]\}$. *The normal version of* $\Phi_{(A_s,A_t)}$ *is the same, so* $s' = \tau_s^{-1}(\Phi^{norm}_{(A_s,A_t)}(A_s)) =$ "2016/04/12". *As another example, Let* $s =$ "2016 *May* 12" *and the same* $A_t$, *so we have,* $A_s = \tau_s(2016\ May\ 12) = \mathcal{M}_{\text{space}}\mathcal{B}_{\text{space}}\mathcal{M}$ *and* $\Phi_{A_s \to A_t} = \{[\text{space} \to /, [5, 6)], [\text{space} \to /, [9, 10)]\}$, *so* $s' = \tau_s^{-1}(\Phi^{norm}_{(A_s,A_t)}(A_s)) =$
"2016 04 12".

In summary, using transformations creates the surjective function $\Phi^{norm}_{(A_s,A_t)}(.)$.

### 4.3 Cluster Augmentation Algorithm

Let $c_i[r_j]$ denote all records in attribute $r_j$ in cluster $c_i$. For each attribute $r_j \in R$, we compute the number of occurrences of each $v \in c_i(r_j)$ in $c_i[r_j]$, as with $n_{(c_i,r_j)}(v)$. The automaton translation applies to all values $v \in c_i(r_j)$ except the correct value of $c_i(r_j)$. Afterward, we reconstruct the statistical representation of each augmented value $v' = \tau_v^{-1}(\Phi^{norm}_{(A_v, A_{\Pi(G_T \setminus \{c_i\})})}(A_v))$ with information of $n_{(c_i,r_j)}(v), v \in c_i(r_j)$ which $\sum_{v \in c_i(r_j)} n_{(c_i,r_j)}(v) = |c_i[r_j]| = |c_i|$. This process is outlined in Algorithm 4.

**Algorithm 4:** Cluster Augmentation (CA)

**Input:** Dataset $D$, Training Dataset $G_T$, Automaton Transformation $\Phi$, String to State Function $\tau$, Cluster Selection Policy $\Pi$
**Output:** Augmented cluster $c'$

Empty cluster $c'$ with Schema $S$;
$c_i, c_t \sim \Pi(G_T | c_i \neq c_t)$;
**for** $r_j \in R$ **do**
  Dictionary $Cor \leftarrow \emptyset$;
  $tr_{c_i} \leftarrow gt(c_i, r_j), tr_{c_t} \leftarrow gt(c_t, r_j)$;
  **for** $v \in c_i(r_j)$ **do**
    **if** $v \neq tr_{c_i}$ **then**
      Select $r_k \in R$, then $v_t \in c_t(r_k) \backslash tr_{c_t}$ with policy $\Pi$;
      $A_v \leftarrow \tau(v), A_t \leftarrow \tau(v_t)$;
      $old \leftarrow True$;
      **while** $old$ **do**
        $v' \leftarrow \tau_v^{-1}(\Phi^{norm}_{(A_v, A_t)}(A_v))$;
        **if** $v' \notin D$ **then**
          $c'(r_j) \leftarrow c'(r_j) \cup v'$;
          Append $Cor[v] \leftarrow v'$;
          $old \leftarrow False$;
        **end**
      **end**
    **else**
      $c'(r_j) \leftarrow c'(r_j) \cup v$;
      Append $Cor[v] \leftarrow v$;
    **end**
  **end**
  **for** $v \in c_i[r_j]$ **do**
    Append $Cor[v]$ to $c'[r_j]$;
  **end**
**end**
**return** $c'$

In Algorithm 4, $gt(c, r)$ provides the correct value of Attribute $r$ in Cluster $c$. The statistical property of the augmented cluster is the same as the original cluster which gives us robustness for the calculation of the statistical signals. If we have some information $\mathcal{I}$, we can apply a conditional distribution $\Pi \sim \mathcal{D}(G_T | \mathcal{I})$ for our selection policies.

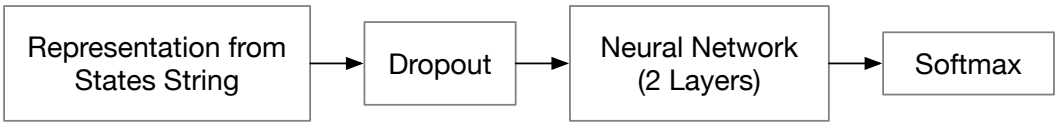



Fig. 4. The architecture of source classifier

### 4.4 Source Information Augmentation

The augmentation algorithm generates artificial clusters. As you can imagine, the generative process cannot be used for the source information attribute. Therefore, After the augmentation algorithm generates all attributes but the source information, we can use language models to extract the format of generated rows and then use the sources' signatures to find the most probable source for them. We use *format* and *Character and token models* over state strings of the original clusters, and use these representations to train a 2-layers neural network with a *Softmax* that generates a probability distribution over all sources of the dataset (see Figure 4). For model optimization, we use logistic loss.

In the prediction, we generate the representation model for the augmented cluster's row using the representation models, and by passing them through the learned model, we obtain a distribution over all sources. We use MAP to assign a source to the input row. We do this prediction for every row in the new cluster to fill its source attribute. The augmented data with source information attribute could also be used for previous approaches. Specifically, approaches that use the statistical properties of the input dataset for data fusion.

## 5 ITERATIVE LEARNING

In our framework, we use an iterative process to improve the learning performance; the key idea here is to use *Hard-Assignment Expectation Maximization* [27]. The framework, at the end of each learning phase, evaluates its output using Hard EM, to maximize the probability of expected values.

### 5.1 Hard Expectation Maximization

In the case of complete data, we have enough *statistical sufficiency* to obtain a conditional probability distribution, but this might not be the case when we have missing values. When some observations are missing, one approach is to randomly choose a value from the random variables domain or use default values. However, such an approach introduces a bias, and the estimation is skewed toward the applied policy for filling in missing values. Moreover, in the random assignment, we will not learn any dependencies between hidden variables because they are independently generated.

An alternative algorithm for optimizing a likelihood function is *Expectation Maximization*. This approach is useful when we have a hidden random variable or incomplete data. The *Expectation Maximization* algorithm starts with an arbitrary point, either parameters $\theta$ or hidden variables $u_o \in U$, where $U$ is the set of cells in $G$ that are missing observations. The algorithm then repeats two steps: first, it uses the current parameters to complete the data, using a probabilistic completion of the different data instances to estimate the expected value of the sufficient statistics (Expectation step or E-step), then it treats the completed data as if it were observed and learn a new set of parameters for the next iteration (Maximization step or M-step).

In our framework, we assume the true value of a record to be a random variable, and the classes (for the prediction task of model $M$) to correspond to the different values from the domain of the record (the values $c_i(r_j)$). Each class $c \in c_i(r_j)$ is associated with a probability distribution over the features of the instances in the class, $P(\mathbf{x}|c)$. This approach views the data as the result of a

*mixture distribution* of classes and attempts to use the hidden variable to separate the mixture into its components. In this setup, the E-step involves computing the probability of different values of the class variables for each instance, but since we only need one value from the estimation of true record representation for the next iteration, we should avoid a soft classification. Therefore, we consider "hard clustering", where each instance contributes all of its weight to the cluster to which it most likely belongs, and if we have more than one we select one randomly. Given parameters $\theta^t$, the cluster of instance $m$ is $c_m = random(\arg\max_c P(c|h(m), \theta^t))$, where $h(.)$ is the featurization model described in Section 3.1, and $m$ is an attribute of $R$ in a cluster of $C$. By filling the unobserved values (which are clusters) with no true record representation ($\mathbf{u_o}$) and adding them to the set of observed true record representation, $G_{C\setminus T}$, to have $(\mathcal{D}^+)^t$, which is the completed set of the data at step $t$. In M-step, we calculate $\theta^{t+1}$ using $(\mathcal{D}^+)^t$ by maximizing likelihood estimation (MLE), $\theta^{t+1} = \arg\max_\theta \mathcal{L}(\theta : (\mathcal{D}^+)^t)$.

### 5.2 Recurrent Process

As we mentioned above, our model is built using a variety of static and dynamic features, which use current cluster values, as they change in each iteration (see Section 3.1). In fact, in every iteration, we will get more accurate predictions, so we will be able to recalculate all the dynamic features more accurately and proceed to the inference of classification variables once again. This approach is inspired by Neville et al. [32], which uses upon the framework's prediction.

In EM, one can start by initializing parameters or use an auxiliary method to complete the data. The first method can make trouble if the initial point does not converge into the global optima. Therefore, we use *Majority Vote* to complete the data with the most probable answers. Notice that *Majority Vote* is not a perfect algorithm, but it can deal with the problem of the parameter initialization that is far from feasible space. Given the input source observations $G_T$, augmented training data $G_A$ and $U$, the framework firstly using *Majority Vote* (MV) estimates true record representation for unobserved random variables, $G_U = MV(D_U)$, and the completed data initialized as $(\mathcal{D}^+)^0 = G_T \cup G_A \cup MV(U)$. Then, the model $M$ parameters $\theta$ can be obtained by MLE, $\mathcal{L}_M(\theta) = \log P(\theta|(\mathcal{D}^+)^0)$. Afterward, the probabilistic inference is used to predict true record representations from the domain of objects $u_o \in U$. We need all true record representations to calculate dynamic representation models, but for some random variables, the true record representation is unknown. Algorithm 5 shows how we calculate the expectation of our model $M^t$ at iteration $t$ to predict its true record representations and how we use this prediction to maximize its likelihood. In all iterations, we fix the true record representation of the observed random variables.

## 6 EXPERIMENTS

We evaluate our record fusion framework using real datasets with various rules. We answer the following questions: (1) how well does the record fusion framework work as a data fusion system compared to the state-of-the-art data fusion systems when the source information of the entries is known? (2) how well does it perform when the source of entries is not available compared to [10] and Majority Vote? (3) what is the impact of different representation contexts on data fusion? (4) is cluster augmentation a valid approach to solving the lack of training data for record fusion problems? (5) how well our iterative algorithm can solve the problem of learning from noisy and incomplete data?

**Algorithm 5:** Iterative Learning (IL)

**Input:** Set of observed random variables $G_T$, Set of augmented random variables $G_A$, Set of unknown random variable $U$, Domain of Random Variables $P$, Number of Iterations $I$, Featurizer Function $h$, Classifier $M$

**Output:** Empirical estimation $\hat{Y}$

Initialize $(\mathcal{D}^+)^0 \leftarrow G_T \cup G_A \cup MV(U)$;
**for** $t = 1$ *to* $I$ **do**
  $X^{(t-1)} \leftarrow h((\mathcal{D}^+)^{(t-1)}, P)$ (calculate the representation vectors);
  Obtaining MLE of $\mathcal{L}_M(\theta) = \log P(\theta : X^{(t-1)})$;
  $(\mathcal{D}^+)^t \leftarrow M(X^{(t-1)}, P; \theta)$ (new predictions using model $M$);
  $(\mathcal{D}^+)^t \leftarrow (\mathcal{D}^+)^0[G_T \cup G_A]$ (replace original values of observed random variables);
**end**
$\hat{Y} \leftarrow (\mathcal{D}^+)^I$;
**return** $\hat{Y}$

Table 1. Datasets used in our experiments.

| Dataset | Size | Clusters | Attributes | # Sources | has constraints? |
|---|---|---|---|---|---|
| Flight | 57,222 | 2,313 | 6 | 37 | No |
| Stock 1 | 113,379 | 2,066 | 10 | 55 | No |
| Stock 2 | 107,260 | 1,954 | 8 | 55 | No |
| Weather | 43,003 | 13,689 | 6 | 11 | Yes |
| Address | 3,287 | 494 | 6 | N/A* | Yes |
| S_Rel | 28,291 | 4,460 | 2 | 523 | No |
| S_Adult | 10,537 | 1,517 | 2 | 603 | No |

* N/A = Address dataset basically has been generated without sources.

Table 2. Precision's Median, Average, and Standard Error of different methods for different datasets.(Refrences: DS[8], ZC[9], GLAD[40], MM[45], BCC[25], CBCC[39], LCF[36], and PM[1])

| Dataset ($G_T$ size) | Prec | $RF_S$ | NB | ACCU | CATD | SSTF | SF | DS | ZC | GLAD | MM | BCC | CBCC | LCF | PM | $RF^+_W$ | $RF_W$ | MV | UM |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Flight (5%) | Med | **0.958** | 0.885 | 0.878 | 0.912 | 0.739 | 0.220 | 0.333 | 0.492 | 0.449 | 0.354 | 0.424 | 0.423 | 0.449 | 0.394 | **0.939** | 0.853 | 0.296 | 0.305 |
| | Avg | 0.949 | 0.882 | 0.892 | 0.909 | 0.732 | 0.241 | 0.376 | 0.517 | 0.418 | 0.333 | 0.374 | 0.419 | 0.449 | 0.369 | 0.947 | 0.859 | 0.176 | 0.337 |
| | StE | 0.001 | 0.009 | 0.005 | 0.013 | 0.021 | 0.003 | 0.036 | 0.024 | 0.028 | 0.018 | 0.046 | 0.003 | $2 \times 10^{-4}$ | 0.024 | 0.008 | 0.041 | 0.202 | 0.105 |
| Stock 1 (5%) | Med | **0.985** | 0.815 | 0.906 | 0.921 | 0.688 | 0.323 | 0.429 | 0.331 | 0.394 | 0.329 | 0.347 | 0.424 | 0.352 | 0.307 | **0.935** | 0.921 | 0.051 | 0.180 |
| | Avg | 0.977 | 0.862 | 0.867 | 0.941 | 0.632 | 0.343 | 0.399 | 0.370 | 0.377 | 0.376 | 0.353 | 0.460 | 0.351 | 0.260 | 0.929 | 0.899 | 0.035 | 0.062 |
| | StE | 0.002 | 0.021 | 0.029 | 0.023 | 0.005 | 0.006 | 0.025 | 0.038 | 0.015 | 0.042 | 0.015 | 0.036 | 0.021 | 0.044 | 0.003 | 0.017 | 0.040 | 0.034 |
| Stock 2 (5%) | Med | **0.938** | 0.840 | 0.853 | 0.823 | 0.779 | 0.767 | 0.35 | 0.325 | 0.317 | 0.36 | 0.428 | 0.397 | 0.361 | 0.325 | **0.939** | 0.889 | 0.745 | 0.796 |
| | Avg | 0.941 | 0.825 | 0.812 | 0.737 | 0.652 | 0.795 | 0.396 | 0.356 | 0.283 | 0.335 | 0.438 | 0.431 | 0.343 | 0.304 | 0.945 | 0.901 | 0.782 | 0.856 |
| | StE | 0.009 | 0.016 | 0.013 | 0.009 | 0.031 | 0.028 | 0.036 | 0.028 | 0.029 | 0.025 | 0.008 | 0.029 | 0.015 | 0.018 | 0.013 | 0.029 | 0.064 | 0.107 |
| Weather (5%) | Med | **0.797** | 0.669 | 0.602 | 0.606 | 0.437 | 0.417 | 0.349 | 0.479 | 0.334 | 0.322 | 0.31 | 0.455 | 0.405 | 0.329 | **0.774** | 0.794 | 0.511 | 0.558 |
| | Avg | 0.822 | 0.719 | 0.662 | 0.607 | 0.513 | 0.441 | 0.347 | 0.430 | 0.368 | 0.367 | 0.346 | 0.451 | 0.405 | 0.353 | 0.867 | 0.787 | 0.641 | 0.672 |
| | StE | 0.004 | 0.018 | 0.009 | 0.017 | 0.011 | 0.003 | 0.002 | 0.044 | 0.034 | 0.035 | 0.025 | 0.003 | $3 \times 10^{-4}$ | 0.023 | 0.021 | 0.044 | 0.091 | 0.053 |
| Address (10%) | Med | n/a* | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | **0.936** | 0.883 | 0.757 | 0.772 |
| | Avg | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | 0.919 | 0.894 | 0.740 | 0.780 |
| | StE | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | 0.013 | 0.037 | 0.231 | 0.183 |
| S_Rel (5%) | Med | **0.932** | 0.719 | 0.592 | 0.693 | 0.301 | 0.247 | 0.615 | 0.486 | 0.538 | 0.581 | 0.607 | 0.564 | 0.657 | 0.610 | **0.843** | 0.754 | 0.542 | 0.615 |
| | Avg | 0.927 | 0.699 | 0.562 | 0.707 | 0.283 | 0.211 | 0.631 | 0.453 | 0.515 | 0.579 | 0.614 | 0.526 | 0.647 | 0.593 | 0.847 | 0.761 | 0.542 | 0.652 |
| | StE | 0.003 | 0.018 | 0.009 | 0.017 | 0.011 | 0.003 | 0.020 | 0.017 | 0.026 | 0.037 | 0.013 | 0.008 | 0.021 | 0.024 | 0.014 | 0.024 | 0.000 | 0.043 |
| S_Adult (5%) | Med | **0.871** | 0.479 | 0.562 | 0.552 | 0.337 | 0.227 | 0.338 | 0.351 | 0.35 | 0.365 | 0.347 | 0.339 | 0.355 | 0.361 | **0.807** | 0.734 | 0.361 | 0.452 |
| | Avg | 0.892 | 0.359 | 0.551 | 0.561 | 0.323 | 0.241 | 0.324 | 0.354 | 0.354 | 0.363 | 0.361 | 0.352 | 0.366 | 0.372 | 0.827 | 0.727 | 0.359 | 0.422 |
| | StE | 0.011 | 0.032 | 0.027 | 0.070 | 0.053 | 0.025 | 0.009 | 0.002 | 0.003 | 0.001 | 0.012 | 0.013 | 0.008 | 0.009 | 0.016 | 0.023 | 0.003 | 0.013 |

* n/a = Address dataset basically has source information, so source-needed algorithms cannot be executed.

Table 3. Precision's Median, Average, and Standard Error of different methods for different datasets with source information augmentation.(+ shows the method process perform over augmented data)(References: *DS*[8], *ZC*[9], *GLAD*[40], *MM*[45], *BCC*[25], *CBCC*[39], *LCF*[36], and *PM*[1])

| Dataset ($G_T$ size) | Prec | $RF_S^+$ | $NB^+$ | $ACCU^+$ | $CATD^+$ | $SSTF^+$ | $SF^+$ | $DS^+$ | $ZC^+$ | $GLAD^+$ | $MM^+$ | $BCC^+$ | $CBCC^+$ | $LCF^+$ | $PM^+$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Flight (5%) | Med | **0.981** | 0.907 | 0.883 | 0.951 | 0.875 | 0.417 | 0.426 | 0.527 | 0.513 | 0.368 | 0.441 | 0.477 | 0.508 | 0.416 |
| | Avg | 0.979 | 0.911 | 0.931 | 0.949 | 0.885 | 0.429 | 0.465 | 0.553 | 0.470 | 0.337 | 0.382 | 0.441 | 0.537 | 0.415 |
| | StE | $7 \times 10^{-4}$ | 0.002 | 0.004 | 0.01 | 0.018 | 0.001 | 0.031 | 0.019 | 0.026 | 0.015 | 0.041 | 0.002 | $4 \times 10^{-5}$ | 0.019 |
| Stock 1 (5%) | Med | **0.988** | 0.855 | 0.952 | 0.942 | 0.823 | 0.519 | 0.523 | 0.416 | 0.437 | 0.405 | 0.362 | 0.451 | 0.398 | 0.363 |
| | Avg | 0.989 | 0.872 | 0.903 | 0.962 | 0.803 | 0.530 | 0.484 | 0.426 | 0.422 | 0.359 | 0.358 | 0.494 | 0.453 | 0.308 |
| | StE | $8 \times 10^{-4}$ | 0.017 | 0.024 | 0.023 | 0.005 | 0.004 | 0.021 | 0.030 | 0.013 | 0.035 | 0.011 | 0.032 | 0.017 | 0.033 |
| Stock 2 (5%) | Med | **0.984** | 0.859 | 0.905 | 0.852 | 0.906 | 0.899 | 0.556 | 0.406 | 0.350 | 0.368 | 0.447 | 0.467 | 0.440 | 0.350 |
| | Avg | 0.980 | 0.862 | 0.855 | 0.779 | 0.652 | 0.942 | 0.574 | 0.391 | 0.352 | 0.339 | 0.443 | 0.467 | 0.424 | 0.360 |
| | StE | 0.006 | 0.013 | 0.008 | 0.005 | 0.027 | 0.022 | 0.029 | 0.025 | 0.021 | 0.020 | 0.002 | 0.023 | 0.014 | 0.016 |
| Weather (5%) | Med | **0.886** | 0.780 | 0.753 | 0.636 | 0.591 | 0.512 | 0.424 | 0.497 | 0.393 | 0.338 | 0.342 | 0.498 | 0.462 | 0.351 |
| | Avg | 0.941 | 0.869 | 0.799 | 0.732 | 0.784 | 0.745 | 0.417 | 0.463 | 0.435 | 0.360 | 0.360 | 0.486 | 0.498 | 0.419 |
| | StE | 0.003 | 0.015 | 0.007 | 0.01 | 0.007 | 0.002 | 0.002 | 0.039 | 0.028 | 0.032 | 0.024 | 0.002 | $7 \times 10^{-5}$ | 0.023 |
| S_Rel (5%) | Med | **0.959** | 0.797 | 0.639 | 0.723 | 0.454 | 0.417 | 0.696 | 0.526 | 0.608 | 0.518 | 0.624 | 0.601 | 0.723 | 0.648 |
| | Avg | 0.961 | 0.773 | 0.606 | 0.741 | 0.459 | 0.396 | 0.725 | 0.490 | 0.564 | 0.563 | 0.625 | 0.559 | 0.733 | 0.647 |
| | StE | 0.003 | 0.016 | 0.005 | 0.014 | 0.010 | 0.002 | 0.017 | 0.013 | 0.022 | 0.034 | 0.012 | 0.006 | 0.018 | 0.022 |
| S_Adult (5%) | Med | **0.892** | 0.531 | 0.621 | 0.578 | 0.480 | 0.441 | 0.429 | 0.385 | 0.436 | 0.360 | 0.361 | 0.379 | 0.417 | 0.389 |
| | Avg | 0.902 | 0.440 | 0.596 | 0.594 | 0.507 | 0.432 | 0.409 | 0.395 | 0.416 | 0.367 | 0.373 | 0.384 | 0.564 | 0.432 |
| | StE | 0.009 | 0.029 | 0.021 | 0.063 | 0.044 | 0.019 | 0.006 | 0.002 | 0.002 | $9 \times 10^{-4}$ | 0.009 | 0.012 | 0.005 | 0.008 |

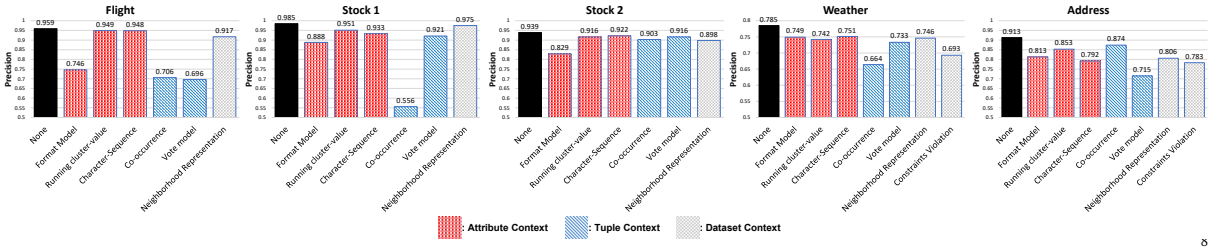



Fig. 5. Ablation studies to evaluate the effect of different representation models.

## 6.1 Experimental Setup

We describe the datasets, metrics, and settings that we use in our experiments.[1] We use seven datasets with different domain properties and usage described in Table 1.

*Flight* [30] is a benchmark dataset that contains flights information from the flight domain. The dataset focused on 2,313 flights departing from or arriving at the hub airports of the three airlines. The ground truth is created by domain experts. *Stock 1 and 2* [30] contain data from popular financial aggregators such as *Yahoo! Finance*, *Google Finance*, and *MSN Money*, official stock-market websites such as *NASDAQ*, and financial-news websites such as *Bloomberg* and *MarketWatch*. the ground truth of *Stock 1* is created by assuming that *NASDAQ* always provides the correct value. the ground truth of *Stock 2* is created by taking the majority value provided by five stock data providers. *Weather* is collected for 30 major USA cities from 11 websites about every 45 minutes. We consider *(city, time)* as the (cluster) key. There are in total 32 collections in a day. The attributes are manually mapped, and there are 6 distinct attributes. The ground truth is provided by domain experts. *Address* reflects applications for discretionary funding to be allocated by the New York City Council. For each record, the attributes that represent legal information, address, and geographical properties of location are selected. The minimum size of each cluster is two and the ground truth has been extracted from the *ISBNsearch* organization website. An interesting feature of the *Address* dataset is that it does not contain any source information (*S_Rel*) [4]. Each cluster contains a topic and a document, and sources are asked to choose the relevance of the topic w.r.t. the document by selecting one out of four choices: 'highly relevant', 'relevant', 'non-relevant', and 'broken link'. *S_Adult* [23]. Each task contains a website, and workers are asked to identify the adult level of the website by selecting one out of four choices: 'G' (General Audience), 'PG' (Parental Guidance), 'R' (Restricted), and 'X' (Porn). For the last two datasets, we only select clusters that we have their ground truth.

**Methods** We compare our approach, referred to as $RF_S$ when we have source information, $RF_W$ when source information is not available and there is no data augmentation, and $RF_W^+$ when source information is not available and we use data augmentation, against several data fusion methods. First, we consider 13 baseline data fusion and crowdsourcing models that need source information (i.e. which source generated each row in the cluster). *NB*: This corresponds to Naive Bayes. Source accuracies are estimated as the fraction of times a source provides the correct value for an object in the ground truth. *ACCU* [12] is the Bayesian data fusion method without source information copying. *CATD* [29] extends source reliability scores with confidence intervals to account for sparsity in source observations. *SSTF* [43] leverages semi-supervised graph learning to exploit the presence of ground truth data. *SlimFast[SF]* [38] is a data fusion based on statistical learning over

[1] https://github.com/HoloClean/RecordFusion

discriminative probabilistic models. *DS* [8] maximizes the likelihood of the observed labels given sources using the EM algorithm. *ZC* [9] uses PGMs without considering priors and estimates the correctness of sources. *GLAD* [40] extends the *ZC* idea with a model that gives difficulty coefficient to each cluster. *MM* [45] gives a different score to each source and uses *Minimax* algorithm to learn a probability for each source. *BCC* [25] maximizes the posterior joint probability distribution to find the correct labels. *CBCC* [39] extends *BCC* by considering the community of sources and calculating a membership score for each cluster into these communities. *LCF* [36] extends *DS* to incorporate prior distribution on source score modeling. *PM* [1] gives a score $[0, +\infty)$ to each source and iteratively determines the score to maximize the likelihood of the observation.

We also compared our method with two approaches that require no source information. *Majority Vote (MV)* considers the maximum frequency value as the true record representation in each cluster attribute. *UM* [10] is an entity consolidation method that uses human-in-the-loop to request the user to verify the equivalence of records, *Majority Vote* can also be used to obtain correct records. To see the rest of the methods refer to [44].

**Evaluation Setup:** To measure precision, we use Precision (P) defined as the fraction of true record representation predictions that are correct. For training, we split the available ground truth into three disjoint sets: (1) a training set $T$, used to find model parameters; (2) a validation set, which is used for hyperparameter tuning; and (3) a test set, which is used for evaluation. To evaluate different dataset splits, *we perform* 50 *runs with different random seeds for each experiment* to ensure that we gain robust results for Precision, we report the median performance. The mean performance along with standard error measurements is also reported. *Seeds are sampled at the beginning of each experiment, and hence, a different set of random seeds can be used for different experiments.* We use *ADAM* [26] as the optimization algorithm for all learning-based models and train all models for 500 epochs with a batch size of ten examples. We run Platt Scaling for 50 epochs. All experiments were executed on a 12-core Intel(R) Xeon(R) CPU E5-2603 v3 @ 1.60GHz with 64GB of RAM running Ubuntu 14.04.3 LTS.

### 6.2 End-to-End Performance

We evaluate the performance of our approach and competing approaches on data fusion in all five datasets. Table 3 summarizes the precision's Median, Average, and Variance of each method. For *Flight*,*Stock 1*, *Stock 2*, and *Weather*, we set the amount of training data to be 5% of the total dataset. For Address, we set the percentage of training data to be 10% (corresponding to 40 clusters) since it is fairly small. The number of iterations is 15, and in the case of using the data augmentation, the amount is 5% of clusters.

As Table 3 shows, our method consistently outperforms all other methods. In the no sources information case, we see improvements of $\sim$ 65/55 points for *Flight* with/without data augmentation. More importantly, we find that our method is able to achieve low standard error in all datasets despite the different clusters and true record representation distributions in each dataset. This is something that seems challenging for prior data fusion methods and reduces the consistency and reliability of their results. Despite the fact that source information is an important factor for other algorithms, *RF* can obtain high precision. This is because the *RF* model estimates the actual data distribution by extracting source signatures using attribute correlation from datasets. For instance, for *Address*, we see that MV can find many true record representations—it has high precision—indicating that most true record representations correspond to the statistical frequency. *Overall, our method achieves an average precision of* 93/87% *without sources information with/without data augmentation, and an average precision of* 99% *when sources information is available across these diverse datasets*, while the performance of competing methods varies significantly.

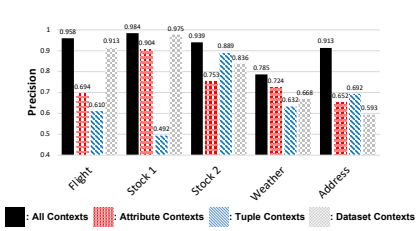



Fig. 6. Ablation studies to evaluate the effect of different representation model groups with data augmentation.

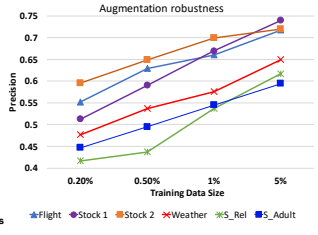



Fig. 7. The effect of increasing the number of clusters via data augmentation without iteration.

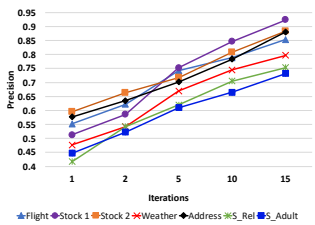



Fig. 8. The effect of increasing the number of iterations without data augmentation.

Table 3 shows the result of the data augmentation with source information, as you can see the data augmentation on average improve the precision of methods by 6 percent. The method that more rely on the statistical sufficiency of the data or their model is more complicated get more increase.

For *CATD*, we see that it achieves relatively high-precision results in *Flight* and *Stock 1*, but it is not consistent on all datasets. Similar performance is exhibited by *Count*. We see that *Count* achieves high precision when the correct answers have significant Bayesian support (i.e., occur relatively often), due to the fact that this dataset has a strong Co-occurrence signal (see Figure 5). Finally, the results of *SlimFast* and *SSTF* vary drastically from 0.220 for *Flight* to 0.779 for *Stock 2*, and under no settings, they give the best performance.

**Takeaway**: Our method with/without cluster augmentation outperforms all other methods. Our data augmentation approach improves the precision of other methods that use the statistical property for data fusion.

### 6.3 Representation Ablation Study

We perform an ablation study to evaluate the effect of different representation models on the quality of our system. Specifically, we compare the performance of *RF* when all representation models are used versus variants of *RF*, where one or a set of representation models is removed at a time.

*6.3.1 Single Representation Effect.* In Figure 5, we report the precision of the different variants as well as the original *RF*. It is shown that removing any feature has an impact on the quality of the predictions of our model. More importantly, we find that different representation models have a different impacts on various datasets. For instance, the most significant drop for *Stock1* and *Weather* is achieved when the co-occurrence model is removed, while for *Flight* and *Address*, the highest drop is achieved when the voting model is removed. Therefore, the representation models that we considered have a positive impact on the performance of our system. As it can be observed in Figure 5, for example, in the dataset *Fight*, removing *Running-cluster value* representation from the model has a minimum impact on its performance, and the parameters of this representation after the model are trained, have values that have a minimum impact.

*6.3.2 Group Contexts Effect.* Figure 6 shows the effect of a group of representation models corresponding to different contexts. It can be observed that removing any context group has an impact on the quality of predictions of our model. Furthermore, for datasets that have various properties, different context groups have the most prominent effect on the performance of *RF*. For *Flight*, *Stock 1*, and *Weather*, the largest drop is achieved when the tuple contexts group models are removed,

while for *Address*, the highest drop is achieved when the dataset contexts group models are removed. This validates our design of considering representation models from different contexts.
**Takeaway**: It is necessary to leverage cell representations that are informed by different contexts to provide robustly and high-quality representation model of data fusion solutions.

### 6.4 Effects of Augmentation on Performance

We evaluate the effectiveness of data augmentation to counteract the lack of training data. Figure 7 shows that using data augmentation yields high-quality record fusion models for datasets with varying sizes and properties (as they were described in section 6.1). Hence, data augmentation is robust to different domains of properties. We ignore Address dataset because it is small to make an adequate augmentation model. We also evaluate the effect of excessive data augmentation: We manually set the ratio between the initial clusters and the lately generated cluster in the final training examples and use augmentation to materialize this ratio. Our results are reported in Figure 7. We see that peak performance is achieved when the ratio between the two types of clusters is about 10% to 30% for all datasets.
**Takeaway**: Data augmentation is an effective and robust way to counteract the lack of enough training data.

### 6.5 Effects of Iterations on Performance

In this experiment, we validate the importance of the iterative process to improve learning performance. Figure 8 shows the results of *RF* for a various number of iterations. The results validate that as the number of iterations increases, we were able to get more accurate predictions. This observation has significant meaning for the performance of *RF* as getting more accurate predictions in each iteration results in recalculating the dynamic features more accurately in each round. For instance, in Weather, *RF* was able to achieve precision less than 0.7 with only one iteration; however, after 15 iterations, the precision was improved over 10 points.
**Takeaway**: The recurrent process is an effective approach for calculating accurate dynamic features.

## 7 RELATED WORK

Several pieces of research have been done on combining data from multiple sources. [2] surveyed existing strategies for resolving inconsistencies in structured data. Specifically, the data fusion methods can be categorized into four main regimes: (I) **Naïve method**: In this method, all sources have a vote, and the correct value for each object is decided by choosing the value that has the maximum votes among all the conflicting values. In these models, there is no worker(source) modeling, so there is no need for source information to discriminate records. (II) **Source-based** main goal is to calculate how accurate is each source(worker). More specifically, in this framework, the votes of the sources(workers) do not have the same "weight" or "quality", so it assigns a weight $q^w$ to worker $w$ and estimates the quality of the worker based on $D$. The importance of each vote depends on the quality of the source. In the crowdsourcing community, this is called worker modeling [24, 44]. The necessity of using these methods is to have minimal source information so that the model can discriminate records. (III) **Relation-based** methods use the main idea of Source-based methods. They also consider the correlation between the sources(worker) (e.g., if a pair of sources copy from each other). (IV) **Transformation-based** methods reduce cluster size by transforming values to each other and use human-in-the-loop to fuse the remained set. (V) **Signature-based** methods use representation models to find the latent signature of the truth pattern. Our model, among other methods, can learn from source-less data and accurately identify patterns by analyzing a portion of human-annotated training data.

In the field of discovering dependencies between data sources, a lot of work has been done as well. In [8, 12, 25, 39], an Bayesian analysis is applied to decide on the dependencies between sources. Some researches [9, 11, 40], considered various types of copying on different data records. Moreover, [13] explores the idea of integrating data and determining the way the sources are interacting with each other by examining the updated history of the sources. There has been a lot of research in the field of evaluating the trustworthiness of the sources resulting in algorithms such as *PageRank* which assigns trust based on link analysis [45] and TrustFinder [42] which decides the importance of a source based on its behavior in a P2P network. Moreover, [15] examined the problem of selecting a subset of sources before integration. The authors claim that by choosing only the sources that can be beneficial for their algorithm, they can achieve higher performance than by using all the available sources and data.

Furthermore, [33] solve the data fusion problem by creating an iterative model. The main idea of their fact-finding algorithm was to incorporate prior knowledge from the users into their process, to integrate data from conflicting claims. The SLiMFast [38] proposes a framework to solve the data fusion problem as a learning and inference problem over discriminative probabilistic graphical models. SliMFast was also the first that came with guarantees on its error rate for the estimation of the source accuracy.

In [10, 28] human-in-the-loop is used to solve entity consolidation; instead of using sources, the system simulates source information for entity resolution by asking for information from an oracle.

## 8 CONCLUSION

We introduce a machine learning framework for *record fusion*, where the underlying challenge is in two well-known classical problems: *data fusion* and *golden record*. We learn rich data representation models and resolve the training data shortage via data augmentation. We iteratively obtain and apply a model that can predict the record representations for unlabeled data. Our proposal outperformed previously proposed models.

## REFERENCES


[1] Bahadir Ismail Aydin, Yavuz Selim Yilmaz, Yaliang Li, Qi Li, Jing Gao, and Murat Demirbas. 2014. Crowdsourcing for Multiple-Choice Question Answering.. In *AAAI*. Citeseer, 2946–2953.

[2] Jens Bleiholder and Felix Naumann. 2006. *Conflict handling strategies in an integrated information system.* Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät ....

[3] Anne Brüggemann-Klein and Derick Wood. 1998. One-unambiguous regular languages. *Information and computation* 142, 2 (1998), 182–206.

[4] Chris Buckley, Matthew Lease, Mark D Smucker, Hyun Joon Jung, Catherine Grady, Chris Buckley, Matthew Lease, Mark D Smucker, Catherine Grady, Matthew Lease, et al. 2010. Overview of the trec 2010 relevance feedback track (notebook). In *The nineteenth text retrieval conference (TREC) notebook*.

[5] Chengliang Chai, Ju Fan, Guoliang Li, Jiannan Wang, and Yudian Zheng. 2019. Crowdsourcing database systems: Overview and challenges. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2052–2055.

[6] Xu Chu, Ihab F Ilyas, and Paraschos Koutris. 2016. Distributed data deduplication. *Proceedings of the VLDB Endowment* 9, 11 (2016), 864–875.

[7] 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.

[8] Alexander Philip Dawid and Allan M Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 28, 1 (1979), 20–28.

[9] Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. 2012. ZenCrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proceedings of the 21st international conference on World Wide Web*. 469–478.

[10] Dong Deng, Wenbo Tao, Ziawasch Abedjan, Ahmed Elmagarmid, Guoliang Li, Ihab F Ilyas, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2017. Unsupervised String Transformation Learning for Entity Consolidation. *arXiv preprint arXiv:1709.10436* (2017).

[11] Xin Luna Dong, Laure Berti-Equille, Yifan Hu, and Divesh Srivastava. 2010. Global detection of complex copying relationships between sources. *Proceedings of the VLDB Endowment* 3, 1-2 (2010), 1358–1369.

[12] Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava. 2009. Integrating conflicting data: the role of source dependence. *Proceedings of the VLDB Endowment* 2, 1 (2009), 550–561.

[13] Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava. 2009. Truth discovery and copying detection in a dynamic world. *Proceedings of the VLDB Endowment* 2, 1 (2009), 562–573.

[14] Xin Luna Dong and Theodoros Rekatsinas. 2018. Data integration and machine learning: A natural synergy. In *Proceedings of the 2018 International Conference on Management of Data*. ACM, 1645–1650.

[15] Xin Luna Dong, Barna Saha, and Divesh Srivastava. 2012. Less is more: Selecting sources wisely for integration. *Proceedings of the VLDB Endowment* 6, 2 (2012), 37–48.

[16] Xin Luna Dong and Divesh Srivastava. 2013. Compact explanation of data fusion decisions. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 379–390.

[17] Ahmed K Elmagarmid, Panagiotis G Ipeirotis, and Vassilios S Verykios. 2006. Duplicate record detection: A survey. *IEEE Transactions on knowledge and data engineering* 19, 1 (2006), 1–16.

[18] Henning Fernau. 2009. Algorithms for learning regular expressions from positive data. *Information and Computation* 207, 4 (2009), 521–541.

[19] Alban Galland, Serge Abiteboul, Amélie Marian, and Pierre Senellart. 2010. Corroborating information from disagreeing views. In *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 131–140.

[20] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1321–1330.

[21] Alireza Heidari, Ihab F Ilyas, and Theodoros Rekatsinas. 2019. Approximate Inference in Structured Instances with Noisy Categorical Observations. *arXiv preprint arXiv:1907.00141* (2019).

[22] Alireza Heidari, Joshua McGrath, Ihab F Ilyas, and Theodoros Rekatsinas. 2019. HoloDetect: Few-Shot Learning for Error Detection. *arXiv preprint arXiv:1904.02285* (2019).

[23] https://github.com/ipeirotis/Get Another-Label/tree/master/data. [n.d.]. Adult Datset.

[24] Yuan Jin, Mark Carman, Ye Zhu, and Yong Xiang. 2020. A technical survey on statistical modelling and design methods for crowdsourcing quality control. *Artificial Intelligence* (2020), 103351.

[25] Hyun-Chul Kim and Zoubin Ghahramani. 2012. Bayesian classifier combination. In *Artificial Intelligence and Statistics*. 619–627.

[26] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[27] Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

[28] Evgeny Krivosheev, Siarhei Bykau, Fabio Casati, and Sunil Prabhakar. 2020. Detecting and preventing confused labels in crowdsourced data. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2522–2535.

[29] Qi Li, Yaliang Li, Jing Gao, Lu Su, Bo Zhao, Murat Demirbas, Wei Fan, and Jiawei Han. 2014. A confidence-aware approach for truth discovery on long-tail data. *Proceedings of the VLDB Endowment* 8, 4 (2014), 425–436.

[30] Xian Li, Xin Luna Dong, Kenneth Lyons, Weiyi Meng, and Divesh Srivastava. 2012. Truth finding on the deep web: Is the problem solved? *Proceedings of the VLDB Endowment* 6, 2 (2012), 97–108.

[31] Ester Livshits, Alireza Heidari, Ihab F Ilyas, and Benny Kimelfeld. 2020. Approximate Denial Constraints. *arXiv preprint arXiv:2005.08540* (2020).

[32] Jennifer Neville and David Jensen. 2000. Iterative classification in relational data. In *Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data*. 13–20.

[33] Jeff Pasternack and Dan Roth. 2010. Knowing what to believe (when you already know something). In *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, 877–885.

[34] John Platt et al. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers* 10, 3 (1999), 61–74.

[35] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment* 11, 3 (2017), 269–282.

[36] Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning from crowds. *Journal of Machine Learning Research* 11, 4 (2010).

[37] Theodoros Rekatsinas, Xu Chu, Ihab F Ilyas, and Christopher Ré. 2017. Holoclean: Holistic data repairs with probabilistic inference. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1190–1201.

[38] Theodoros Rekatsinas, Manas Joglekar, Hector Garcia-Molina, Aditya Parameswaran, and Christopher Ré. 2017. Slimfast: Guaranteed results for data fusion and source reliability. In *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 1399–1414.

[39] Matteo Venanzi, John Guiver, Gabriella Kazai, Pushmeet Kohli, and Milad Shokouhi. 2014. Community-based bayesian aggregation models for crowdsourcing. In *Proceedings of the 23rd international conference on World wide web*. 155–164.

[40] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan, and Paul L Ruvolo. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in neural information processing systems*. 2035–2043.

[41] Minji Wu and Amélie Marian. 2011. A framework for corroborating answers from multiple web sources. *Information Systems* 36, 2 (2011), 431–449.

[42] Xiaoxin Yin, Jiawei Han, and S Yu Philip. 2008. Truth discovery with multiple conflicting information providers on the web. *IEEE Transactions on Knowledge and Data Engineering* 20, 6 (2008), 796–808.

[43] Xiaoxin Yin and Wenzhao Tan. 2011. Semi-supervised truth discovery. In *Proceedings of the 20th international conference on World wide web*. ACM, 217–226.

[44] Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. 2017. Truth inference in crowdsourcing: Is the problem solved? *Proceedings of the VLDB Endowment* 10, 5 (2017), 541–552.

[45] Dengyong Zhou, Sumit Basu, Yi Mao, and John C Platt. 2012. Learning from the wisdom of crowds by minimax entropy. In *Advances in neural information processing systems*. 2195–2203.