Towards Interpretable and Learnable Risk Analysis for Entity Resolution

Zhaoqiang Chen[†], Qun Chen[†], Boyi Hou[†], Tianyi Duan[†], Zhanhuai Li[†] and Guoliang Li[‡]
[†]School of Computer Science, Northwestern Polytechnical University, Xi'an, China
[†]Key Laboratory of Big Data Storage and Management, Northwestern Polytechnical University, Ministry of Industry and Information Technology, Xi'an, China
[‡]Department of Computer Science, Tsinghua University, Beijing, China
{chenzhaoqiang@mail.,chenbenben@,ntoskrnl@mail.,tianyiduan@mail.,lizhh@}nwpu.edu.cn
liguoliang@tsinghua.edu.cn

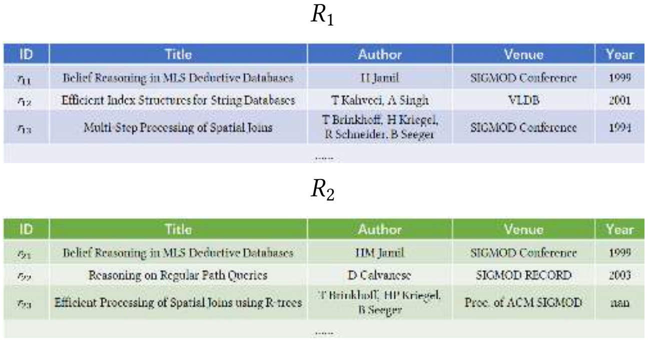
ABSTRACT

Machine-learning-based entity resolution has been widely studied. However, some entity pairs may be mislabeled by machine learning models and existing studies do not study the risk analysis problem – predicting and interpreting which entity pairs are mislabeled. In this paper, we propose an interpretable and learnable framework for risk analysis, which aims to rank the labeled pairs based on their risks of being mislabeled. We first describe how to automatically generate interpretable risk features, and then present a learnable risk model and its training technique. Finally, we empirically evaluate the performance of the proposed approach on real data. Our extensive experiments have shown that the learning risk model can identify the mislabeled pairs with considerably higher accuracy than the existing alternatives.

KEYWORDS

Entity resolution, Learnable risk model, Interpretability

ACM Reference Format:
Zhaoqiang Chen[†], Qun Chen[†], Boyi Hou[†], Tianyi Duan[†], Zhanhuai Li[†] and Guoliang Li[‡]. 2020. Towards Interpretable and Learnable Risk Analysis for Entity Resolution. In *SIGMOD'20: 2020 International Conference on Management of Data, June 14–19, 2020, Portland, Oregon, USA*. ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/1122445.1122456



R_1

ID	Title	Author	Venue	Year
r_{11}	Belief Reasoning in MLS Deductive Databases	H Jamil	SIGMOD Conference	1999
r_{12}	Efficient Index Structures for String Databases	T Kahveci, A Singh	VLDB	2001
r_{13}	Multi-Step Processing of Spatial Joins	T Brinkhoff, H Kriegel, R Schneider, B Seeger	SIGMOD Conference	1994
		......		

R_2

ID	Title	Author	Venue	Year
r_{21}	Belief Reasoning in MLS Deductive Databases	HM Jamil	SIGMOD Conference	1999
r_{22}	Reasoning on Regular Path Queries	D Calvanese	SIGMOD RECORD	2003
r_{23}	Efficient Processing of Spatial Joins using R-trees	T Brinkhoff, HP Kriegel, B Seeger	Proc. of ACM SIGMOD	nan
		......		

Figure 1: An ER running example.

1 INTRODUCTION

Entity resolution (ER) aims at identifying the equivalent records that refer to the same real-world entity. Considering the running example in Figure 1, ER aims to match the paper records between two tables, R_1 and R_2. A pair of $< r_{1i}, r_{2j} >$, in which r_{1i} and r_{2j} denote a record in R_1 and R_2 respectively, is called an *equivalent* pair if and only if r_{1i} and r_{2j} refer to the same paper; otherwise, it is called an *inequivalent* pair. In the example, r_{11} and r_{21} are *equivalent* while r_{11} and r_{22} are *inequivalent*.

ER can be taken as a binary classification problem, which aims to label the record pairs as equivalent/inequivalent. Recently several learning models have been proposed to address the ER problem (most notably among them is deep neural network (DNN) [20, 41]). Unfortunately, ER can be very challenging in real scenarios due to the prevalence of incomplete and dirty values in the records [21]. Therefore, many record pairs may be mislabeled by a learning model, and the model is hard to interpret such that (1) which pairs are mislabeled, and (2) why these pairs are mislabeled. Thus it is highly desirable to accurately analyze the risk of the labeled pairs returned by a learning model, i.e., ranking the

SIGMOD'20, June 14–19, 2020, Portland, Oregon, USA

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

labeled pairs based on their risks of being mislabeled. We call this task *risk analysis* for ER, which not only let users know the risks of the labeled results but also evaluate and tune the performance of an ER learning model based on the high-risk pairs.

Although existing learning models can compute a probability indicating the uncertainty of pair status and utilize the probability to quantify the risk, they have several limitations for risk analysis. (1) Not easily interpretable. The existing learning models are hard to interpret. (2) Not learnable. Existing learning models can not be tuned to accurately capture the risk, because they aim to minimize the inconsistency between the predicted result and the ground truth, while risk analysis aims to analyze the risks of labeled results. Therefore, it requires to design a separate model for risk analysis.

In this paper, we propose a novel framework for interpretable and learnable ER risk analysis. Our work focuses on the learning model that can accurately rank the labeled pairs based on their risks of being mislabeled. Our major contributions are summarized as follows.

- We propose an interpretable and learnable framework for ER risk analysis, which consists of risk feature generation, risk model construction and risk model training. To the best of our knowledge, this is the first interpretable and learnable approach for risk analysis.
- We design a learning model to evaluate the risks of labeled pairs in ER and propose a training algorithm to efficiently tune the model towards a specific workload.
- We present a technical solution of automatic risk feature generation for ER risk analysis.
- We empirically evaluate the performance of the proposed solution on real data by a comparative study. Our extensive experiments have shown that it can identify the mislabeled pairs with considerably higher accuracy than existing alternatives.

This paper considers risk analysis as a separate process independent of ER classifier training. Due to uncertainty and non-interpretability of DNN output, it has been well recognized in the AI community [4] that risk analysis is a vital issue to AI safety. As a result, separate risk analysis has been actively studied in the literature[31, 32, 35, 59]. Compared with DNN output, our proposed approach, referred to as LearnRisk in this paper, can effectively improve both accuracy and interpretability. Even though we focus on ER in this paper, the proposed approach can be potentially generalized to other classification tasks. On the other hand, it is worthy to point out that the applications of risk analysis are not limited to risk ranking, which can be directly used to reduce required manual cost in machine and human collaboration for high-quality entity resolution [33]. It can also be potentially leveraged to optimize the essential processes of classifier training, including active selection of training instances and model training. We discuss how to leverage risk analysis for classifier training in Section 8.

The rest of this paper is organized as follows: Section 2 reviews related work. Section 3 defines the ER task. Section 4 introduces the general framework. Sections 5 and 6 present the technical solution for ER, with Section 5 focusing on risk feature generation and Section 6 on risk model. Section 7 presents our empirical evaluation results. Section 8 discusses the potential applications of risk analysis. Finally, Section 9 concludes this paper.

2 RELATED WORK

We discuss related work from the orthogonal perspectives of risk analysis and entity resolution.

Risk Analysis. Risk analysis has been alternatively called *confidence ranking* or *trust scoring* in the previous literature. There has been a growing interest in the study of risk analysis in recent years [31, 32, 35, 59]. The authors of [31] showed that the output probabilities from softmax distributions, though may be misleading if viewed in isolation, can perform fairly well in detecting mislabeled data on various tasks including computer vision, natural language processing and automatic speech recognition. In their following work [32], they proposed to fine tune a pre-trained classifier with an auxiliary dataset of outliers and an outlier exposure module to better detect the out-of-distribution data. Alternatively, the authors of [35] employed the metric of relative cluster distance, which compares the distances of an instance to the cluster with the same label and its nearest cluster with a different label, to measure the risk. However, these proposals are not easily interpretable. Moreover, they are not trainable, thus can not be adapted to a specific workload. While the aforementioned work focused on risk analysis for the tasks of image and speech recognition, the authors of r-HUMO [33] proposed an approach of risk analysis for the task of ensuring quality guarantees for entity resolution. The proposed approach was however tailored to the problem setting of quality control. Based on the work of r-HUMO, the same team [14] has proposed a similar but more general approach for risk analysis on ER. It first estimates a pair's equivalence probability distribution by Bayesian inference and then uses the metric of Conditional Value at Risk (CVaR) to measure its risk. Unfortunately, the proposed risk model is not trainable either.

Due to the uncertainty of DNN output, there is a research field of confidence calibration [30, 42] that seeks to transform a classifier output into a probability estimate representative of true correctness likelihood. Unfortunately, the calibration techniques usually do not change the ranking order of instances as measured by classifier output. Risk analysis however needs to accurately rank the instances by measured risk. Therefore, they cannot serve as reliable risk indicators. Moreover, similar to DNN output, calibrated output also has the interpretability issue. Another research field related to risk analysis is to provide with explaining models for classification results [3, 7, 44]. However, the field of model explanation mainly put effort on providing interpretable information for human analysis. In contrast, this paper aims to provide an interpretable and learnable framework for automatic quantitative risk measurement.

Entity Resolution. ER plays a key role in data integration and has been extensively studied in the literature [16, 17, 21]. ER can be automatically performed based on rules [22, 39, 47], probabilistic theory [24, 48] and machine learning models [15, 19, 36, 46]. The progressive paradigm for ER [2, 37, 56] has also been proposed for the application scenario in which ER should be processed efficiently but does not necessarily require to generate high-quality results. Taking a pay-as-you-go approach, it studied how to maximize quality given a given budget.

It has been well recognized that automatic entity resolution can be very challenging in real scenarios due to the prevalence of dirty and incomplete values [21]. Therefore, there have been an enduring interest in involving the human in resolution process for improving the performance. Many active learning techniques have been proposed for the task of ER [5, 8, 40, 46]. It is noteworthy that the pair selection strategies employed by active learning can naturally serve as the metrics for risk analysis. However, their purpose is to improve the overall performance of an under-developed classifier, but not to evaluate the risk of individual pairs being mislabeled by a well-trained classifier. Thus, as the existing learning models for ER, they are not able to accurately capture the risk of individual pairs. In recent years, many researchers [13, 18, 25, 26, 28, 29, 38, 40, 50–53, 55, 58] have also studied how to crowdsource an ER workload. While these researchers addressed the challenges specific to crowdsourcing, we instead focus on risk analysis for ER in this paper.

3 PROBLEM STATEMENT

Given some records, entity resolution reasons about whether two records are equivalent. Two records are deemed to be equivalent if and only if they correspond to the same real-world entity. We call a pair an *equivalent* pair if and only its two records are equivalent; otherwise, it is called an *inequivalent* pair. An ER solution labels each pair in a workload as *matching* or *unmatching*. Note that given an ER workload, a perfect solution would label each equivalent pair as *matching* and each inequivalent pair as *unmatching*. However, due to the inherent challenge of entity resolution, a model may be prone to mislabeling some pairs.

Table 1: Frequently Used Notations.

Notation	Description
R	a table consisting of records
D	an ER workload consisting of record pairs
r_i	a record in a table
d_i	a record pair in D
S_i	a labeling solution for d_i
f_i	a feature of a pair
F_i	a feature set
D_f	the set of pairs with the feature f

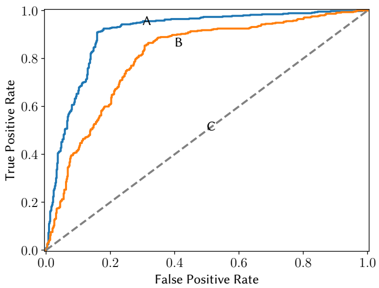


Figure 2: An examples of ROC curve: C denotes a trivial baseline model, and model A is better than B.

Given an ER workload of D and a labeling solution of S for D, the task of risk analysis is to rank the pairs in D by risk such that the mislabeled pairs can be generally ranked before the correctly labeled pairs. As in previous work [31], we employ the metric of Receiver Operating Characteristic (ROC) curve [23] to measure the performance of risk analysis.

Let *TP* denote the number of true positives, *FP* the number of false positives, *TN* the number of true negatives and *FN* the number of false negatives. Then, the true positive rate, denoted by *TPR*, is equal to $\frac{TP}{TP+FN}$, and the false positive rate, denoted by *FPR*, is equal to $\frac{FP}{FP+TN}$. Note that in the circumstance of risk analysis, a positive pair refers to a mislabeled pair and a negative pair refers to a correctly labeled pair. ROC curve illustrates *TPR* against *FPR* at different threshold settings. It depicts the relative tradeoff between benefit (true positives, i.e. the mislabeled pairs) and cost (false positives, i.e. the correctly labeled pairs) [23]. By the metric of ROC, Area Under ROC (AUROC) can be deemed as the probability that a risk model assigns higher score to a randomly chosen positive pair than a randomly chosen negative pair [23, 31]. Therefore, a model with a higher AUROC achieves better quality. Note that a trivial model without discrimination has the AUROC of 50%. An example of ROC curve has been shown in Figure 2.

For presentation simplicity, we summarize the frequently used notations in Table 1. Formally, we define the task of risk analysis as follows:

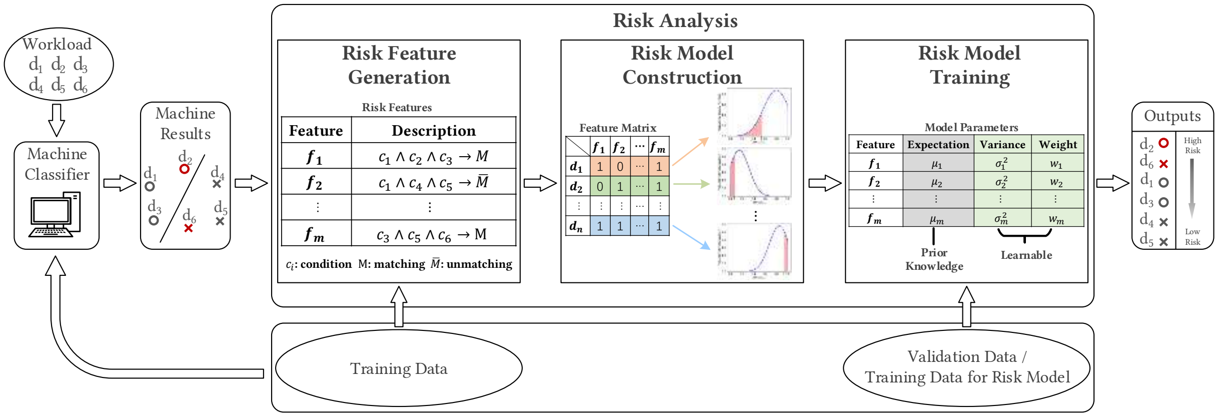


Figure 3: The framework of risk analysis: the circle and cross symbols represent different classes, and the mislabeled pairs are highlighted by red.

Definition 1. **[Risk Analysis for ER].** *Suppose that an ER workload of D consists of n record pairs,* $\{d_1, d_2, \ldots, d_n\}$*, and a machine classifier of S labels each pair in D as* matching *or* unmatching*. The task of risk analysis is to quantitatively measure the risk of each pair in D being mislabeled such that the metric of AUROC is maximized.*

4 RISK ANALYSIS FRAMEWORK

The general framework of risk analysis, as shown in Figure 3, consists of the following three steps:

- **Risk feature generation**;
- **Risk model construction**;
- **Risk model training**;

In the rest of this section, we will elaborate each of the three steps.

4.1 Risk Feature Generation

The framework measures mislabeling risk based on features. To enable interpretable and learnable risk analysis, the risk features should have the following three desirable properties:

- **Interpretable.** For a risk model to be interpretable, its risk features should be interpretable, or easily understandable to the human.
- **Discriminating.** For a risk model to be effective, its risk features should be highly discriminating, or indicative of the ground-truth labels of pairs. In contrast, a non-discriminating feature can not serve as an effective risk feature, because it can not provide valuable insights into class status.
- **High-coverage.** For a risk model to be learnable, its risk features should represent the common knowledge shared among many pairs. High coverage ensures that the knowledge learned on training data can be effectively transferred to test data. In contrast, a low-coverage feature has limited utility for risk analysis because the knowledge it embodies is not easily transferable.

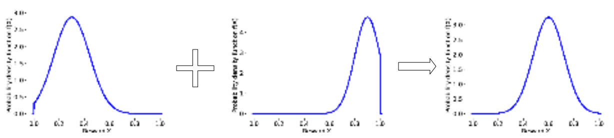
The challenge of risk feature generation is then how to automatically generate interpretable, discriminating and high-coverage risk features. In the scenario of ER, we observe that rule is the most common form of knowledge that can be easily understood by the human. Therefore, we propose to extract the rules satisfied by a pair as its risk features. For instance, in the running example shown in Figure 1, if two records have different publication years, it is unlikely that they refer to the same paper. This knowledge can be represented by the rule of

$$r_i[Year] \neq r_j[Year] \rightarrow inequivalent(r_i, r_j), \qquad (1)$$

in which r_i denotes a record and $r_i[Year]$ denotes r_i's value at the attribute of $Year$. With this knowledge, a *matching* pair whose two records have different publication years can be reasoned to be at high risk of being mislabeled. We discuss how to automatically generate risk features in Section 5.

4.2 Risk Model Construction

For each extracted risk feature, the framework models its equivalence probability by a distribution. The step of risk model construction then estimates the equivalence probability distribution of each pair based on its risk features, and quantifies its risk based on the estimated distribution. Inspired by the success of risk analysis in investment theory, we propose to construct risk model based on the theory of portfolio investment [45]. Specifically, the framework considers each risk feature as a stock and each pair as a portfolio of component stocks. It models the distribution of a risk feature (resp. a pair) by the reward distribution of its corresponding stock (resp. portfolio). As shown in Figure 4, the



Stock A Stock B Portfolio C

Figure 4: Estimation of the reward distribution of an investment portfolio: aggregating the reward distributions of its component stocks.

reward distribution of a portfolio (resp. a pair) is estimated by aggregating the reward distributions of its component stocks (resp. risk features).

On the task of ER, we take the label of each pair as a random variable, which follows the Bernoulli distribution with the parameter p representing its equivalence probability. Statistically speaking, the parameter p follows the Beta distribution $Beta(\alpha, \beta)$ [1], where α and β denote the shape parameters. If the values of α and β are large (>=10), the Beta distribution can be approximated by the Normal distribution [57]. Note that the value of $(\alpha + \beta)$ can be interpreted as sample size. In the case of ER, sample size is usually large. Therefore, we model the equivalence probability of an ER pair by normal distribution.

Specifically, the framework denotes the normal distribution of the equivalence probability of a labeled pair by $\mathcal{N}(\mu_i, \sigma_i^2)$, where μ_i and σ_i^2 denote its expectation and variance respectively. Since a valid equivalence probability should be between 0 and 1, we transform an inferred normal distribution to a *truncated form* in the range of 0 to 1 [12]. Suppose that there are m risk features, $\{f_1, f_2, \ldots, f_m\}$. Let $\mathbf{w} = [w_1, w_2, ..., w_m]^T$ denote the feature weights, and $\mathcal{N}(\mu_{f_j}, \sigma_{f_j}^2)$ denote the equivalence probability distribution of the feature f_j. Accordingly, the expectation vector of m features is denoted by $\mu_F = [\mu_{f_1}, \mu_{f_2}, \ldots, \mu_{f_m}]^T$, and the variance vector by $\sigma_F^2 = [\sigma_{f_1}^2, \sigma_{f_2}^2, \ldots, \sigma_{f_m}^2]^T$. For each pair d_i, we denote its feature vector by $\mathbf{x}_i = [x_{i1}, x_{i2}, ..., x_{im}]$, where $x_{ij} = 1$ if d_i has the jth feature, otherwise $x_{ij} = 0$. Then, the distribution of d_i is represented by $\mathcal{N}(\mu_i, \sigma_i^2)$, in which

$$\mu_i = \mathbf{x}_i(\mathbf{w} \circ \mu_F), \tag{2}$$

and

$$\sigma_i^2 = \mathbf{x}_i(\mathbf{w}^2 \circ \sigma_F^2), \tag{3}$$

where $\circ$ represents the elementwise product. Finally, provided with the distribution of a pair, the framework uses the popular metrics proposed for investment risk measurement (e.g. Value at Risk (VaR)) to quantify its risk of being mislabeled. A risk model would rank all the labeled pairs based on their quantified risks.

It is noteworthy that the existing approaches based on DNN output use a single value to represent equivalence probability, which corresponds to the expectation in our proposed risk model. However, in investment risk analysis, it has been observed that besides expected return, return fluctuation also plays an important role in risk estimation. The scenario of ER risk analysis is similar in that equivalence probability fluctuation also brings additional uncertainty. Therefore, instead of using a single value, our proposed risk model uses normal distribution to more accurately capture the uncertainty of label status. As shown in investment risk analysis [49], the metric of VaR, which estimates the highest probability of an instance being mislabeled in the majority of cases, is quite effective at capturing fluctuation risk. Therefore, we use the metric of VaR in this paper. However, it is noteworthy that other metrics can be similarly used in the framework. We discuss how to construct the risk model in Section 6.

4.3 Risk Model Training

Provided with a risk model, the final step is to tune the model towards a specific workload such that it can flexibly reflect the characteristics of the workload. Similar to traditional classifier model training, risk model training primarily involves parameter tuning. Since the framework aims to rank the labeled pairs by the risk of being mislabeled, it requires a solution of learning to rank [11].

Specifically, the proposed risk model has three types of parameters, the weight of risk feature (w_i), the expectation (μ_i) and variance (σ_i^2) of risk feature distribution. Typically, the framework considers the expectation of feature distribution as prior knowledge and estimates it based on the labeled data used for classifier training. It instead tunes the parameters of w_i and σ_i^2 for optimal risk ranking. Desirably, the training data for risk model should be directly selected from a target workload. In practice, the machine learning solutions usually require additional labeled data for classifier validation. They can be leveraged for risk model training. We discuss how to train risk model in Section 6.

5 RISK FEATURE GENERATION

The framework uses rules to represent risk features. We note that decision tree [19] or its variation, random forest [28], has been widely used to generate the ER rules. Aiming to directly label a workload, the existing techniques of decision tree construction would generate a limited number of labeling rules. Provided with a labeling rule, if a pair satisfies the condition specified at Left-Hand Side (LHS), it would be labeled as the class specified at Right-Hand Side (RHS); otherwise, it is implied that the pair would be labeled as a different class. For instance, consider the labeling rule specified

[1] https://en.wikipedia.org/wiki/Conjugate_prior

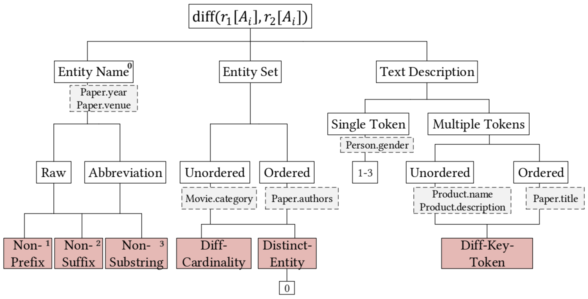


Figure 5: Summary of the difference metrics defined on string attributes.

as follows

$$sim(r_i[Title], r_j[Title]) > 0.9 \rightarrow equivalent(r_i, r_j), \quad (4)$$

in which $r_i[Title]$ denotes r_i's value at the attribute of $Title$, and $sim(\cdot,\cdot)$ measures the similarity between two attribute values. The rule reasons that if the similarity between two records' attribute values at $Title$ is larger than 0.9, they are equivalent; *otherwise, they are inequivalent.*

In contrast, the rule of risk feature is one-sided. One-sidedness means that if a pair satisfies the condition specified at LHS, it is very likely that it belongs to the class specified at RHS; however, nothing needs to be implied if a pair does not satisfy the condition. For instance, consider the rule specified in Eq. 1, which states that if two paper records have different publication years, it is very unlikely that they refer to the same paper. It is not a good labeling rule because even though two papers share the same publication year, it can not be said with high confidence that they refer to the same paper. However, it can serve as an effective risk feature.

In the rest of this section, we first describe how to design basic metrics based on attribute values for ER and then present an approach to automatically generate the interpretable, discriminating and high-coverage risk features based on the basic input metrics.

5.1 Basic Metric Design

Similar to the existing rule-based ER solutions [47], we specify rules by the comparison operations between attribute values. We note that a wide variety of similarity metrics have been proposed for different types of attribute values (e.g., string and numerical) [16]. Since the similarity metrics focus on the common part between two values, they are usually effective as the indicators of equivalence status. Since low similarity usually means low equivalence probability, similarity metrics can also be used to indicate inequivalence status. However, for the reasoning of inequivalence status, the metrics that directly capture the difference between two values may be more effective. For instance, in the running example as shown in Figure 1, if two paper records have different numbers of authors, it is very likely that they refer to different papers even though they may share most of the authors. Consider another scenario where a paper record contains a very specific token in its title, which however does not occur in the title of another record. It is very likely that these two records refer to different papers even though their titles may appear highly similar.

Therefore, besides similarity metrics, we also define *difference metrics*, which are denoted by $diff(r_1[A_i], r_2[A_i])$, to directly capture the difference between two attribute values. Since the comparisons between numerical values are straightforward (e.g., >, <, = and ≠), as usual we focus on string values in this paper. Even though there exist many off-the-shelf string similarity metrics, there are very few difference metrics defined in the literature as far as we know. Therefore, in the rest of this subsection, we present the difference metrics defined on string values. We categorize them by the type of string values, which includes entity name, set of entity names, and text description:

- **Entity name**. To specifically capture the difference between two entity names, we define the difference metrics of *non-substring* and *abbr-non-substring*, *non-prefix* and *abbr-non-prefix*, *non-suffix* and *abbr-non-suffix*. The metric of *non-substring* (resp. *non-prefix* and *non-sufffix*) indicates whether a value is a substring (resp. prefix and suffix) of another value. Similarly, the metric of *abbr-non-substring* (resp. *abbr-non-prefix* and *abbr-non-suffix*) indicates whether the first-letter abbreviation of a value is a substring (resp. prefix and suffix) of another value abbreviation. It can be expected that a metric value of 1 for these metrics usually indicates that two entity names represent different entities.

- **Entity set**. To specifically capture the difference between two entity sets, we propose the difference metrics of *diff-cardinality* and *distinct-entity*. The metric of *diff-cardinality* indicates whether the number of entity names in two sets are different, while the metric of *distinct-entity* counts the number of distinct entity names, which exist in only one of two sets.

Example 1. *Suppose that two different papers have the author list of* s_1 = *"T Brinkhoff, H Kriegel, R Schneider, B Seeger" and* s_2 = *"T Brinkhoff, H Kriegel, B Seeger" respectively, in which each author is identified by the comma splitter. The similarity metric of* entity-based JaccardIndex *is measured at 0.75, which tends to label them as an equivalent pair; while the difference metric of* distinct-entity *is 1, which captures the different author "R Schneider". It can be observed that in this example,* distinct-entity *is a more effective indicator of inequivalence status than* entity-based JaccardIndex.

- **Text description**. An attribute value of text description consists of one or multiple tokens. Unlike entity name, a value of text description usually has no abbreviation, and contains a long text (e.g., the attribute of *title* in a paper table). To specifically capture the difference between two text values, we propose the metric of *diff-key-token*. The metric of *diff-key-token* indicates the number of the key or discriminating tokens contained by one and only one record in a pair. A discriminating token can effectively identify an entity, can thus serve as an effective indicator of inequivalence status.

We have summarized the difference metrics in the hierarchical structure as shown in Figure 5. With the help of hierarchical guidance, we can easily design proper difference metrics for various string attributes.

5.2 Rule Generation

In constructing a traditional *two-sided* decision tree, a partition operation divides an ER pair set of D into two separate subsets, D_L and D_R, both of which consist of mostly equivalent or inequivalent pairs. In contrast, the process of rule generation for risk analysis is one-sided. At each iteration, it only needs to extract a subset of pairs D_i from D such that most of the pairs in D_i have the same class status. The remaining pairs in the subset of D'=D-D_i can instead have mixed labels. The extraction operation is supposed to be iteratively applied on D'.

In the construction of two-sided decision tree, the label purity of a partition operation o on a pair set D is usually measured by the metric of *Gini index* [10], which is defined by

$$G(D, o) = \frac{|D_L|}{|D|}G(D_L) + \frac{|D_R|}{|D|}G(D_R), \qquad (5)$$

where $G(D_L)$ and $G(D_R)$ denote the *Gini* values of the subsets. The *Gini* value of a pair set is defined by

$$G(D_*) = 1 - t_M^2 - t_U^2, \qquad (6)$$

where t_M (resp. t_U) denotes the proportion of real matches in a subset D_* (resp. real unmatches in D_*). It can be observed that the metric of *Gini* value measures the *impurity* of a subset. An optimal partition operation would result in the minimum value of $G(D, o)$.

Similarly, in the construction of one-sided decision tree, we design the one-sided Gini index as follows

$$\hat{G}(D, o) = min(\frac{\lambda}{|D_L|}+(1-\lambda)G(D_L), \frac{\lambda}{|D_R|}+(1-\lambda)G(D_R)), \qquad (7)$$

where λ is the weight parameter to balance the influence of size and impurity. A large λ means we prefer set size over set purity. Since the extracted subset needs to be highly discriminating, we suggest to set the value of λ at low (e.g. 0.2 in our implementation). A partition operation would generate two subsets with the minimum value of one-sided Gini index. By this setting, the construction process achieves the purpose that each partition would produce a highly pure leaf node regardless of the purity of the other one.

Algorithm. We have sketched the procedure of automatic rule generation in Algorithm 1. Given a pair set of D, it exhaustively applies each of the basic metrics on D. For each basic metric, the algorithm chooses the best condition value on the metric, which results in the minimum Gini value, to extract a subset of mostly equivalent or inequivalent pairs. Note that each partition operation on D would generate a separate decision tree. The partition operation is iteratively invoked on the subset consisting of the remaining pairs until either the remaining pairs satisfies a pre-specified purity threshold, or the depth of decision tree reaches a pre-specified threshold. We have implemented the algorithm based on the open-sourced project of two-sided decision tree construction[2]. Due to the imbalance problem in the ER task (i.e., there usually exist much more inequivalent pairs than equivalent ones), for the generation of matching rules, the procedure sets a large class weight (e.g. 1000 in our implementation) to the matching instances. However, the generated matching rules are finally filtered without class weighting. To ensure high coverage of risk feature, we also set a lower threshold on the sheer size of any extracted subset (e.g. 5 in our implementation). The algorithm would generate a forest of decision trees, in which each leaf corresponds to a rule or risk feature.

The example rules generated by constructing one-sided decision trees have been shown in Figure 6, in which f_1 and f_2 are both valid rules, and f_3 is however not valid because its purity does not exceed the threshold. In Algorithm 1, the input basic metrics are the similarity and difference metrics specifically designed for ER. However, it can be observed that the general approach for risk feature generation are applicable to other classification tasks. Provided with another classification task, we only need to replace the basic similarity and difference metrics; the algorithm similarly applies.

Complexity. Let n denote the size of training data, m the number of basic metrics, and h the pre-specified depth of decision trees. The total time complexity of rule generation can be represented by $O(h(2m)^h \cdot nlogn)$. It is worthy to point out that the number of basic metrics (m) is usually limited (e.g. dozens); to ensure interpretability, the maximum depth of decision tree (h) is usually set to a small value (e.g., $h \leq 4$). Therefore, the algorithm for automatic rule generation can

[2] https://scikit-learn.org/stable/modules/tree.html

Algorithm 1: Risk Feature Generation

```
Input: A set of labeled data $D_t$;
       Impurity threshold $\tau$;
       Tree depth $h$;
Output: Risk features $F$.
$F \leftarrow \{\}$;
$current\_depth \leftarrow 0$;
$tree \leftarrow null$;
$ConstructTree(D_t, current\_depth, tree, F)$;
Remove redundant rules in F;
return $F$;

Procedure $ConstructTree(D_c, depth, tree, F)$
    if $depth >= h$ then
        $rules \leftarrow$ traverse the $tree$, select the leaves
          whose impurities do not exceed $\tau$;
        $F \leftarrow F + rules$;
    else
        for each attribute $A_i \in \mathbb{A}$ do
            for $w_{class} \in \{weight_0, weight_1\}$ do
                $o_* \leftarrow argmin_{o \in A_i} \hat{G}(D_c, o, w_{class})$;
                Split $D_c$ into $D_L$ and $D_R$ based on $o_*$;
                Add the information of $o_*, D_L, D_R$ to
                  $tree$;
                Calculate the impurities of $D_L$ and $D_R$,
                  denoted by $\tau_L$ and $\tau_R$;
                $\tau_{min} \leftarrow min(\tau_L, \tau_R)$;
                $\tau_{max} \leftarrow max(\tau_L, \tau_R)$;
                if $\tau_{min} >= \tau$ or $\tau_{max} < \tau$ then
                    $rules \leftarrow$ traverse the $tree$, select the
                      leaves whose impurities do not
                      exceed $\tau$;
                    $F \leftarrow F + rules$;
                else
                    if $\tau_L > \tau_R$ then
                        $ConstructTree(D_L, depth + 1, tree, F)$
                    else
                        $ConstructTree(D_R, depth + 1, tree, F)$
                Remove $o_*, D_L, D_R$ information from $tree$;
```

be executed efficiently. Our evaluation in Subsection 7.5 has also shown that it scales well with the size of training data.

6 RISK MODEL

Given an ER workload of D, we denote the equivalence probability of each pair d_i in D by a random variable, p_i. The framework models p_i by a normal distribution, $\mathcal{N}(\mu_i, \sigma_i^2)$, where μ_i and σ_i^2 denote its expectation and variance respectively. As mentioned in Section 4.2, the distribution of p_i is supposed to be estimated by aggregating the distributions of risk features. In this section, we first introduce the metric of Value at Risk (VaR) to quantify the risk of a pair, and then present the techniques for risk model training.

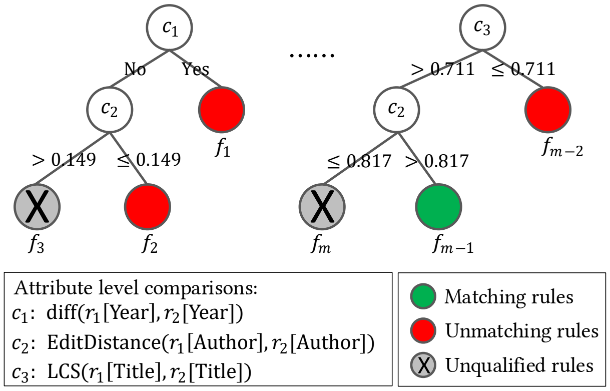


Figure 6: Illustration of automatic rule generation: each path from the root to a leaf in one-sided decision trees corresponds to a rule.

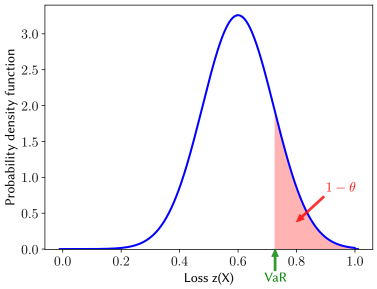


Figure 7: Visualization of the VaR risk metric.

6.1 Risk Metric

As in portfolio investment [6], we employ the popular metric of Value at Risk (VaR) to measure the risk of a pair being mislabeled by the machine. Given a confidence level of θ, VaR represents the maximum loss after excluding all worse outcomes whose combined probability is at most $1-\theta$. Formally, given the *loss* function $z(X) \in L^p(\mathcal{F})$ of a portfolio X and θ, the metric of VaR is defined as follows:

$$VaR_\theta(X) = inf\{z_* : P(z(X) \geq z_*) \leq (1-\theta)\}. \qquad (8)$$

Given a pair d_i, we denote its equivalence probability by p_i, and the inverse of its cumulative distribution function by $F_i^{-1}(\cdot)$, i.e., the quantile function. If d_i is labeled as *unmatching* by the machine, its probability of being mislabeled by the machine is equal to p_i. Accordingly, its worst-case loss corresponds to the case that p_i is maximal. Therefore, given the confidence level of θ, the VaR of d_i is the maximum value of $z = p_i$ after excluding the $1 - \theta$ worse cases where p_i is from $F_i^{-1}(\theta)$ to $+\infty$. Formally, the VaR risk of a pair d_i with the machine label of *unmatching* can be estimated by

$$VaR_\theta(d_i) = F_i^{-1}(\theta; \mu_i, \sigma_i^2). \tag{9}$$

Similarly, if d_i is labeled by the machine as *matching*, its VaR risk of a pair d_i can be estimated by

$$VaR_\theta(d_i) = 1 - F_i^{-1}(1 - \theta; \mu_i, \sigma_i^2). \tag{10}$$

We have visualized the risk metric of VaR by a pair example with the machine label of *unmatching* in Figure 7, in which the X axis represents the loss and the Y axis represents the probability density function. In Figure 7, the area of red zone is $1 - \theta$, which corresponds to the probability of a loss being larger than 0.757. In this case, the VaR value is equal to 0.757.

6.2 Model Training

In this subsection, we detail how to train a risk model. We first describe the parameter setting, and then present the objective loss function and the technique of parameter optimization.

6.2.1 Parameter Setting. It has been well recognized that although machine classifiers, if used separately, are not able to accurately measure the risk of labeled pairs, their outputs can usually provide with valuable hints on the risk. Specifically, a probability output close to 0 or 1 usually means that the target pair is at low risk of being mislabeled, while an output close to the ambiguous value of 0.5 usually indicates a high risk. Therefore, besides the rules generated by one-sided random forest, the risk model also incorporates classifier output as one of the risk features. Furthermore, it is desirable that the weight of a classifier output increases with the extremeness of its value. Therefore, we model the influence of classifier output on risk measurement by the function as follows

$$f_w(x) = -e^{\frac{-(x-0.5)^2}{2\alpha^2}} + \beta + 1.0, \tag{11}$$

where x denotes the classifier output, α and β are the shape parameters that need to be learned. An example of the influence function has been presented in Figure 8. It can be observed that as expected, the influence increases with the extremeness of the output value. With regard to classifier output, the process of model training only needs to learn two parameters, α and β, instead of the n parameters for each distinct value.

The model considers the expectations of risk feature distributions as prior knowledge and estimates them by statistical analysis on the labeled data used to train a classifier. Specifically, given a risk feature, f_i, suppose that there are totally n_i pair instances with the feature f_i in the classifier training data, and m_i instances among them have the label of *matching*. Then, the expectation of f_i is estimated at $\frac{m_i}{n_i} \times 100\%$.

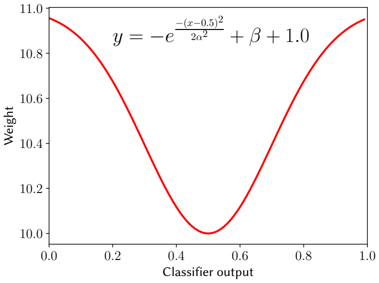


Figure 8: An example of the influence function, where $\alpha = 0.2$ and $\beta = 10$.

The feature weights and the variances of risk features instead need to be learned. It can be observed that in general, larger variance means higher risk. We use the metric of *Relative Standard Deviation* (RSD) to measure the uncertainty of feature distribution. Formally, RSD is defined by

$$RSD = \sigma_i / \mu_i, \tag{12}$$

in which μ_i and σ_i denote the expectation and standard deviation of a feature distribution respectively. The process of model learning needs to learn a value of RSD for each risk feature. For the risk feature of classifier output, we split the pairs into multiple subsets, each of which contains similar classifier outputs. The process of model learning only needs to learn a value of RSD for each subset.

6.2.2 Loss Function. The primary purpose of risk analysis is to prioritize risky pairs such that the mislabeled pair can be ranked higher than the correctly labeled ones. Therefore, we employ the technique of learning to rank [11] to optimize model parameters such that the risk model can best reflect the characteristics of a target workload.

Given any two pairs, d_i and d_j, suppose that their risks of being mislabeled are measured at γ_i and γ_j respectively. We denote $d_i \rhd d_j$ if and only if d_i is ranked higher than d_j in terms of risk, or $\gamma_i > \gamma_j$. Let p_{ij} denote the posterior probability of $d_i \rhd d_j$. As in [11], we use the logistic function to map the risk value to the posterior probability by

$$p_{ij} = \frac{e^{(\gamma_i - \gamma_j)}}{1 + e^{(\gamma_i - \gamma_j)}}. \tag{13}$$

According to Eq. 13, the posterior probability increases with the quantitative risk difference between d_i and d_j, or the value of $(\gamma_i - \gamma_j)$.

We set the desired target value of p_{ij} to

$$\bar{p}_{ij} = 0.5 * (1 + \hat{g}_i - \hat{g}_j), \tag{14}$$

where $\hat{g}_i, \hat{g}_j \in \{0, 1\}$ are the risk labels of pairs. If a pair d_i is mislabeled, $\hat{g}_i = 1$; otherwise, $\hat{g}_i = 0$. According to Eq. 14, if d_i is mislabeled and d_j is correctly labeled, the value of

$\bar{p}_{ij}$ is 1; if instead d_i is correctly labeled and d_j is mislabeled, the value of $\bar{p}_{ij}$ is 0. The general objective of parameter optimization is to minimize the difference between p_{ij} and $\bar{p}_{ij}$. In other words, if d_i is mislabeled and d_j is correctly labeled, the optimization process would maximize the value of $(\gamma_i - \gamma_j)$. It corresponds to maximally ranking d_i before d_j in terms of risk. Therefore, our value settings of p_{ij} and $\bar{p}_{ij}$ exactly maximize the metric of AUROC as described in Section 3.

Formally, given a risk training dataset D_Y, as in [11], we define the optimization objective by the cross-entropy loss function of

$$L(D_Y) = \sum_{d_i, d_j \in D_Y} -\bar{p}_{ij} \cdot log(p_{ij}) - (1 - \bar{p}_{ij}) \cdot log(1 - p_{ij}), \tag{15}$$

in which the value of $L(D_Y)$ increases with the value difference between p_{ij} and $\bar{p}_{ij}$ as desired.

6.2.3 Parameter Optimization. We have implemented the process of parameter optimization based on the platform of TensorFlow [1]. We employed the widely used technique of *Gradient Descent* to find the parameters that minimize the loss function. Since the loss function $L(D_Y)$ is a composition of differentiable functions (e.g., $log(x)$ and e^x), hence $L(D_Y)$ is also differentiable. Given a learning rate ϵ, the parameter w_i is iteratively updated as follows

$$w_i^{(j+1)} = w_i^{(j)} - \epsilon \cdot \frac{\partial L(D_Y)}{\partial w_i}. \tag{16}$$

Similarly, the parameter σ_i is iteratively updated by

$$\sigma_i^{(j+1)} = \sigma_i^{(j)} - \epsilon \cdot \frac{\partial L(D_Y)}{\partial \sigma_i}. \tag{17}$$

We set the learning rate ϵ to 0.001 in our implementation. To alleviate the overfitting problem, we also add a combination of L_1 and L_2 regularization to the objective loss function.

7 EXPERIMENTS

In this section, we empirically evaluate the performance of the proposed solution, which is denoted by *LearnRisk*, by a comparative study. To the best of our knowledge, no learnable risk model has been proposed in the literature. Therefore, we have compared the proposed solution with the following non-learnable alternatives:

- *Baseline* [31]. The baseline technique simply measures the risk of a pair by the ambiguity of its classifier output. The pairs with more ambiguous (close to 0.5) outputs (equivalence probabilities) are considered to be at higher risk compared to those with the more extreme outputs.
- *Uncertainty* [40]: In active learning for ER, the authors of [40] proposed to first train multiple classifiers based on different training sets generated by bootstrapping, and then use their predictions to estimate the equivalence probability of a pair. With the equivalence probability estimated at p, the risk is measured by the uncertainty score of $p(1-p)$.
- *TrustScore* [35]: The authors of [35] proposed a clustering-based solution for risk analysis. Firstly, it builds separate clusters to represent each class based on the training data. For a test data d_i, let ρ_Y denote its distance to the cluster whose label is the same as d_i's, and ρ_N denote its distance to the nearest cluster whose label is different from d_i's. Then, the *TrustScore* of d_i is estimated by $\frac{\rho_N}{\rho_Y}$. By *TrustScore*, the closer a test data is to its predicted class cluster, the lower risk it has.
- *StaticRisk* [14]: The authors of [14] proposed to take the equivalence probability provided by machine classifier as the prior expectation and use the human-labeled pairs as samples to estimate the posterior expectation and variance by the Bayesian inference. Not learnable, *StaticRisk* employed the metric of conditional value at risk to measure pair risk.

Besides the aforementioned techniques for risk analysis, we also compare *LearnRisk* with the rule-based solution for ER, *Holoclean* [43]. Note that the existing rule-based solutions for ER aim to label data, while *LearnRisk* ranks the risk of labeled results. They are solving different problems. We however extend Holoclean to support the problem of risk analysis.

The rest of this section is organized as follows: Subsection 7.1 describes experimental setup. Subsection 7.2 evaluates the performance of various risk analysis techniques. Subsection 7.3 compares *LearnRisk* with *Holoclean*. Subsection 7.4 evaluates the performance sensitivity of *LearnRisk* w.r.t the size of risk training data. Finally, Subsection 7.5 evaluates the scalability of *LearnRisk*.

7.1 Experimental Setup

We have used four real datasets in the empirical evaluation, whose details are presented as follows:

- DBLP-Scholar[3] (denoted by DS): the experiments match 2616 DBLP entries from DBLP publications with 64263 Scholar entries from Google Scholar.
- Abt-Buy[4] (denoted by AB): the experiments match 1081 product entries from Abt.com with 1092 product entries from Buy.com.
- Amazon-Google[5] (denoted by AG): the experiments match 1363 product entities from Amazon and 3226 product entities from Google product search service.

[3] https://dbs.uni-leipzig.de/file/DBLP-Scholar.zip

[4] https://dbs.uni-leipzig.de/file/Abt-Buy.zip

[5] https://dbs.uni-leipzig.de/file/Amazon-GoogleProducts.zip

Table 2: The statistics of datasets.

Dataset	Size	# Matches	# Attributes
DS	41416	5073	4
AB	52191	904	3
AG	13049	1150	4
SG	144946	6842	7

Unlike the AB dataset, the AG products are mainly software.

- Songs[6] (denoted by SG): the experiments match the entries within a table, which consists of around one million song records.

On all the datasets, we use the blocking technique to filter the pairs deemed unlikely to match. Due to large size of the original SG dataset, we randomly select 5 percent of the records from it as our test workload. The statistics of the four test datasets are presented in Table 2.

We have used DeepMatcher [41], a state-of-the-art deep learning solution for ER, as the machine classifier. For comparative study, as required by DeepMatcher, we split each test dataset into three parts by a pre-specified ratio (e.g. 3:2:5), which specifies the proportions of training, validation and test data respectively. In the comparative study, we have evaluated the performance of different techniques under the circumstances with various ratio settings of training, validation and test data. As expected, our experiments showed that the performance of DeepMatcher generally increases with the proportion of classifier training data.

For *Uncertainty*, we train 20 deep learning models for each test dataset. For *TrustScore*, the input features are the summaries of attribute similarities, which are represented by 300-dimensions vectors in the deep learning model. For risk feature generation of *LearnRisk*, we have designed 19 basic metrics on the attribute values in the DS workload, which include 8 $diff(\cdot)$ metrics; totally 9 basic metrics on attribute values in the AB workload, which include 2 $diff(\cdot)$ metric; totally 11 basic metrics on the attribute values in the AG workload, which include 2 $diff(\cdot)$ metrics; totally 14 basic metrics in the SG dataset, which include 5 $diff(\cdot)$ metrics. In the evaluation of *LearnRisk*, we use the assigned validation data for risk model training. We set the confidence of the VaR risk model at 0.9 and the number of epochs for parameter optimization at 1000.

7.2 Comparative Evaluation

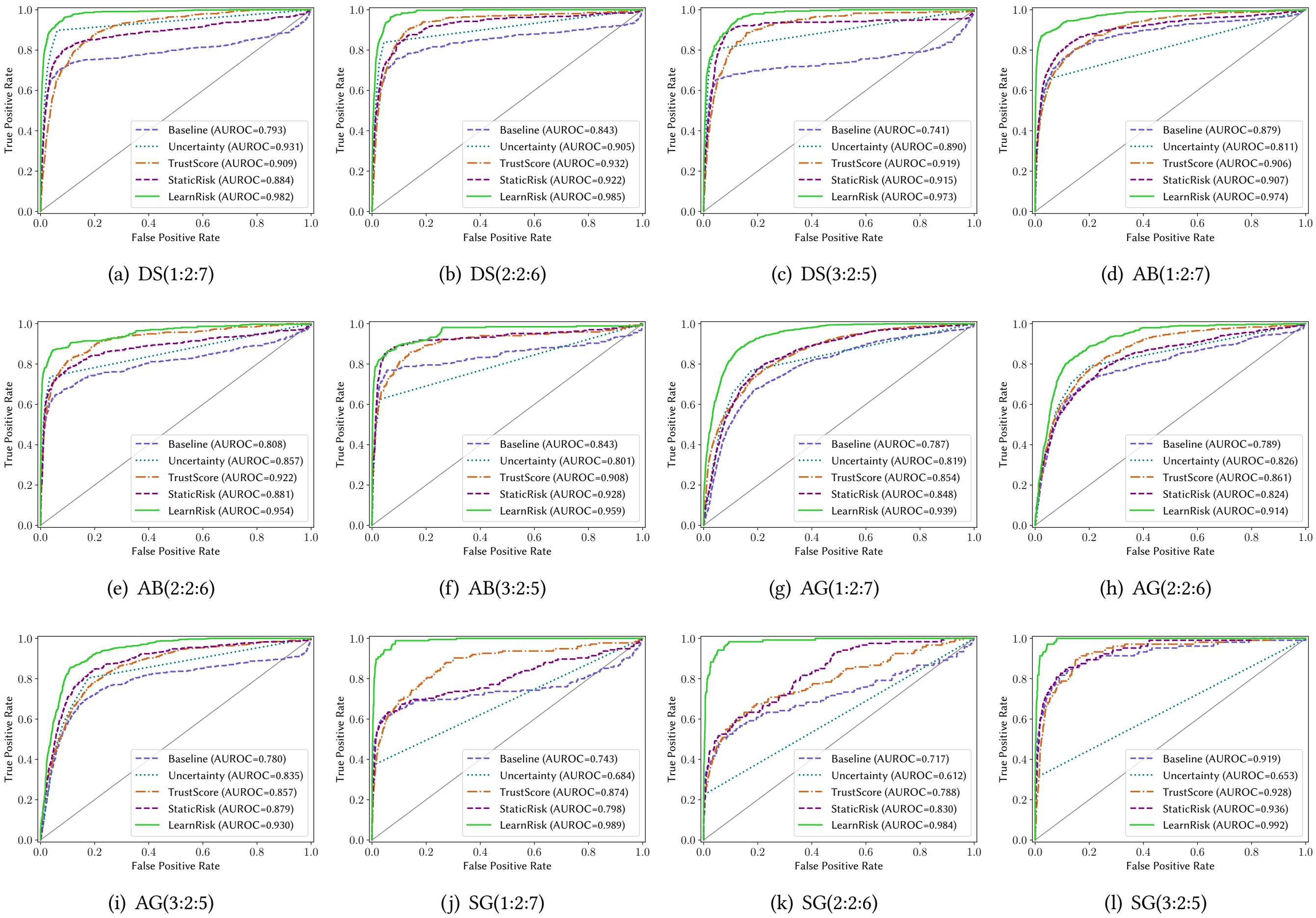
We have compared *LearnRisk* to its alternatives in the experimental settings with various ratios of train, validation and test. The purpose is to track the accuracy of risk analysis under different circumstances with various machine classifier performance. The ratio has been set at 1:2:7, 2:2:6 and 3:2:5 respectively. The detailed evaluation results are presented in Figure 9. Note that *Uncertainty* estimates the risk scores based on 20 trained models, which can only generate 21 different scores. Since many test data may share the same risk score, we have observed that *Uncertainty* achieves highly regular ROC curves.

It can be observed that *LearnRisk* consistently performs better than its alternatives under all the test circumstances. Specifically, *LearnRisk* outperforms *Baseline* and *Uncertainty* by very considerable margins (more than 10% measured by AUROC) in most cases. *TrustScore* and *StaticRisk* performs better than *Baseline* and *Uncertainty*. However, *LearnRisk* still manages to outperform them by more than 5% (measured by AUROC) in most cases. It is also worthy to point out that the performance of *LearnRisk* is much more stable compared to its alternatives. For instance, on the SG workloads with different ratios, the achieved AUROCs of *StaticRisk* are 0.798, 0.830 and 0.936 respectively, while the achieved AUROCs of *LearnRisk* fluctuate only slightly (0.989, 0.984 and 0.992 respectively). Our experimental results have evidently shown that *LearnRisk* performs better its alternatives in both accuracy and stability. They bode well for its efficacy in real scenarios.

Out-of-distribution Evaluation. To further demonstrate the advantage of *LearnRisk*, we have also evaluated the performance of various approaches in the out-of-distribution (OOD) circumstance, where the distribution of classifier training data is different from the validation and test data. This setting simulates the scenario where a pre-trained model is applied in a new environment. We have generated two OOD test workloads. In the first workload, the training data come from the DBLP-ACM dataset, while the validation and test data come from the DBLP-Scholar dataset. We denote this workload by DA2DS. In the second workload, the training data come from Abt-Buy, while the validation and test data come from Amazon-Google. As expected, our experiments showed that the performance of DeepMatcher on both OOD workloads deteriorates considerably compared to its performance on the DS and AB workloads.

The detailed comparative evaluation results are presented in Figure 10. It can be observed that similar to the same-distribution case, *LearnRisk* consistently performs better than its alternatives on all the test cases. As expected, the performance margins between *LearnRisk* and its alternatives become more considerable. It is also interesting to point out that the performance of the non-learnable alternative risk models fluctuates wildly between the two OOD workloads. For instance, *TrustScore* performs well on DA2DS with the AUROC score of 0.921, while its performance on AB2AG is much worse with the AUROC score of 0.548. It can also be observed that *StaticRisk* performs much better on AB2AG than

[6] http://pages.cs.wisc.edu/ãnhai/data/falcon_data/songs/



(a) DS(1:2:7)

(b) DS(2:2:6)

(c) DS(3:2:5)

(d) AB(1:2:7)

(e) AB(2:2:6)

(f) AB(3:2:5)

(g) AG(1:2:7)

(h) AG(2:2:6)

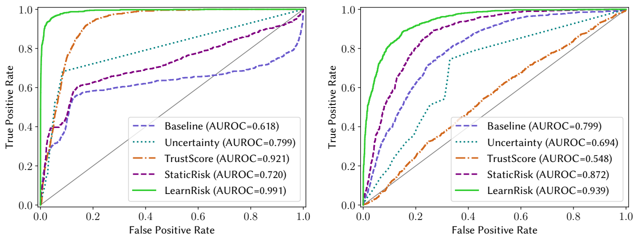
(i) AG(3:2:5)

(j) SG(1:2:7)

(k) SG(2:2:6)

(l) SG(3:2:5)

Figure 9: Comparative evaluation on four real datasets with varying ratios of (training:validation:test).



(a) DA2DS dataset.

(b) AB2AG dataset.

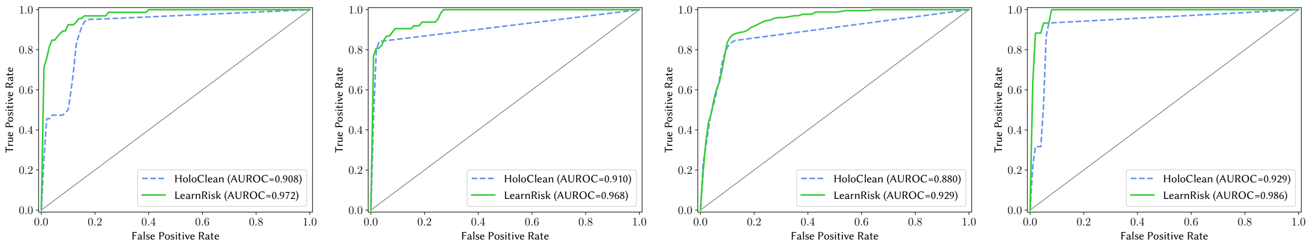
Figure 10: Comparative evaluation on the out-of-distribution datasets.

on DA2DS (0.872 vs 0.720). In comparison, the performance of *LearnRisk* is much more stable with the scores of 0.991 and 0.939 on DA2DS and AB2AG respectively. These experimental results demonstrate that the proposed risk model can effectively learn the characteristics of a target workload for improved risk measurement.

7.3 Comparison with HoloClean

HoloClean [43] is a data cleaning solution based on probabilistic inference, but can be adapted for ER. For risk analysis, we treat a candidate pair as a tuple in a relational table, with the labeling rules as its attributes. Considering the rules as a set of integrity constraints over the data, *HoloClean* infers the probabilities of suggested status for noisy machine labels, which can be used to indicate risk of being mislabeled. As in [28], we apply random forest [9] to generate the labeling rules. The input features for the random forest are the same basic metrics used by *LearnRisk*, except that it additionally uses DNN output as one of the metrics. As in LearnRisk, we set the maximum tree depth to 4 and the minimum number of samples to 5. For each test case, the number of generated labeling rules is also set to be very close to the number of one-sided rules generated by LearnRisk. We used the open-sourced implementation of *HoloClean* [7] to support risk analysis.

[7] https://github.com/HoloClean/holoclean



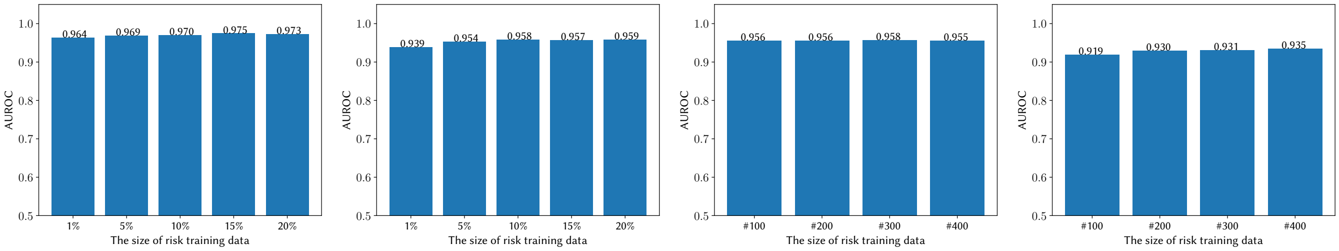
(a) DS.

(b) AB.

(c) AG.

(d) SG.

Figure 11: Comparison with HoloClean.



(a) DS (Random).

(b) AB (Random).

(c) DS (Active).

(d) AB (Active).

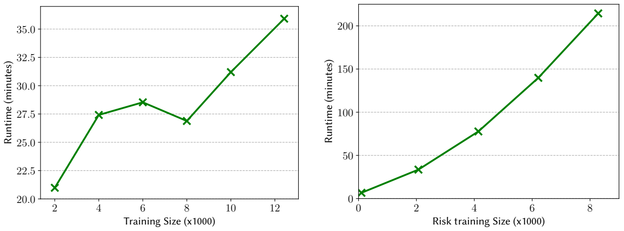
Figure 12: Performance Sensitivity Evaluation of LearnRisk w.r.t size of risk training data: in (a) and (b), the training data are selected by random sampling; in (c) and (d), they are selected by active learning.

Note that effective ER risk analysis requires a large number of rules (e.g. hundreds in our experiments) to ensure high pair coverage. Unfortunately, with hundreds of rules, HoloClean does not scale well with test data size: for instance, with only 295 rules and 2000 pairs from the DS dataset, Holoclean runs more than 24 hours on our machine with four Xeon E7-4820 CPUs, 630GB RAM, running Centos 6.1. Therefore, for each dataset, we generate the test workloads consisting of 1000 pairs, which are randomly sampled from the original dataset; the only exception is SG, in which the test workloads consist of 2000 pairs, because 1000-pair workloads contain few mislabeled instances. For each dataset, we generate 5 subsets and report the average performance over them. Figure 11 shows the comparative evaluation results. We can see that *LearnRisk* consistently performs better than *HoloClean* on all test cases. Our close scrutiny reveals that aiming to correctly label pairs in both ways, the two-sided labeling rules usually have limited efficacy in identifying risky pairs.

7.4 Performance Sensitivity

In this subsection, we evaluate the performance sensitivity of the proposed risk model with regard to the size of risk training data. For sensitivity evaluation, we fix the classifier training data and the test data, but vary the size of risk training data. In our experiments, the percentages of classifier training and test data are set to be 30% and 50% of the original dataset respectively, but the size of risk training data is varied from 100 upward.

We evaluate performance sensitivity by two experiments. In the first experiment, we select the data points for risk training by random sampling. In the second experiment, we select the data points for risk training in an active learning manner. Specifically, we iteratively select the data points with the highest ambiguity score. The detailed evaluation results on DS and AB are presented in Figure 12. The experimental results on AG and SG are similar, thus omitted here due to space limit. It can be observed that with random sampling, the performance of *LearnRisk* is very robust w.r.t the size of risk training data: in the wide arrange between 1% and 20%, the performance of *LearnRisk* only fluctuates very slightly. Even with the percentage of risk training data at 1%, *LearnRisk* outperforms all its alternatives in accuracy. With active selection, we can observe that even with only 100 pairs as risk training data, the performance of *LearnRisk* is better than all its alternatives on DS, and with only 200 training data, the performance of *LearnRisk* is highly competitive with the best of its alternatives on AB. Our experimental results have evidently demonstrated that the risk model of *LearnRisk* can be effectively learned based on only a few carefully selected labeled instances. They further bode well for its efficacy in real scenarios.



(a) Runtime of rule generation. (b) Runtime of *LearnRisk*.

Figure 13: Scalability evaluation on DS.

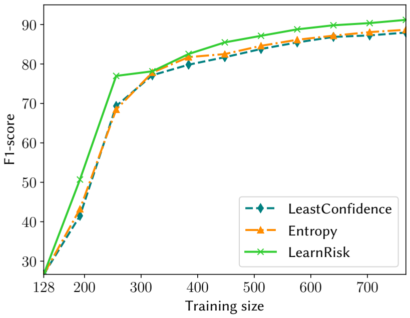


Figure 14: ER Active Learning with *LearnRisk*.

7.5 Scalability Evaluation

We first evaluate the scalability of risk feature generation. We have evaluated its scalability on various DS workloads. The detailed results are presented in Figure 13(a). The results on other datasets are similar, thus omitted here. It can be observed that the runtime consumed by rule generation generally increases linearly with the size of training data. Recall that in rule generation, the impurity of training data plays a key role in determining the number of rules generated. More training data do not necessarily result in more rules. Hence, there exist some fluctuations on runtime between different data sizes. Secondly, we evaluate the scalability of risk model training. Similarly, we have evaluated its scalability on the DS workloads. The detailed results are presented in Figure 13(b). It can be observed that similarly, the runtime consumed by model training increases approximately linearly with the size of risk training data. Our experimental results have evidently shown that *LearnRisk* scales well with workload.

8 DISCUSSION ON POTENTIAL APPLICATIONS OF RISK ANALYSIS

It is worthy to point out that even though this paper considers risk analysis as a separate process independent of ER classifier training, risk analysis can be potentially leveraged to improve classifier training. We discuss two potential applications, active selection of training instances and model training:

Active Learning. The existing techniques for active learning [34, 54] are mainly based on the metric of uncertainty, or the hybrid metrics combining uncertainty and other dimensions (e.g. representativeness). It has been empirically shown [27] that if the batch size of each iteration is set to be large (e.g. 1000), the uncertainty-based metric can usually achieve the overall best performance. Since our proposed approach for risk analysis can provide more accurate uncertainty measurement compared with the existing ones, it can be naturally leveraged for active selection of training instances. At each iteration, the algorithm can select the most risky instances for labeling.

We have conducted an experiment on the DS dataset to compare the proposed risk approach with the uncertainty-based approaches, *LeastConfidence* and *Entropy*. A DeepMatcher model is initially trained on L, then it is iteratively retrained as more data is acquired in batches. With $|L|$=128 and the batch size being equal to 64, the evaluation results on the DS dataset are presented in Figure 14. It can be observed that *LearnRisk* performs better than both *LeastConfidence* and *Entropy*. Due to space limit, further exploration and evaluation are beyond the scope of this paper. However, this preliminary experiment demonstrates that *LearnRisk* is a promising approach for ER active learning.

Model Training. The existing approaches for classifier training usually learn a model whose predictions on the training instances are most consistent with their ground-truth labels. However, the resulting classifier may not perform well on a target workload due to distribution misalignment. Since our proposed approach can provide reliable risk analysis on the classifier labels of target instances, the potential improved approach for classifier training is to train a model on both the labeled training instances and the unlabeled target instances. On the unlabeled instances, the objective is to minimize the prediction risk of classifier labels. Specifically, instead of only considering prediction accuracy on the labeled training instances, the revised objective function for model training consists of two parts, one to enforce the label consistency on the labeled training instances, the other to minimize the prediction risk on the unlabeled target instances.

9 CONCLUSION

In this paper, we have proposed an interpretable and learnable solution for ER risk analysis. Our extensive experiments on real data have validated its efficacy. Our solution focuses on ER, but the proposed framework can be potentially generalized to other classification tasks. Moreover, the risk approach can be potentially leveraged to improve ER classifier training, thus open an interesting research direction.

REFERENCES

[1] M. Abadi, P. Barham, and et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 265–283, 2016.

[2] Y. Altowim, D. V. Kalashnikov, and S. Mehrotra. Progressive approach to relational entity resolution. *Proceedings of the VLDB Endowment*, 7(11):999–1010, 2014.

[3] S. Amershi, M. Chickering, S. M. Drucker, B. Lee, P. Simard, and J. Suh. Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 337–346, 2015.

[4] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete problems in ai safety. *arXiv preprint, arXiv:1606.06565*, 2016.

[5] A. Arasu, M. Götz, and R. Kaushik. On active learning of record matching packages. *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 783–794, 2010.

[6] P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath. Coherent measures of risk. *Mathematical finance*, 9(3):203–228, 1999.

[7] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K.-R. MÃžller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(Jun):1803–1831, 2010.

[8] K. Bellare, S. Iyengar, A. G. Parameswaran, and V. Rastogi. Active sampling for entity matching. In *Proceedings of the 18th ACM international Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1131–1139, 2012.

[9] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[10] L. Breiman. *Classification and regression trees.* Routledge, 2017.

[11] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. N. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine learning (ICML)*, pages 89–96, 2005.

[12] J. Burkardt. The truncated normal distribution. *Department of Scientific Computing Website, Florida State University*, pages 1–35, 2014.

[13] C. Chai, G. Li, J. Li, D. Deng, and J. Feng. Cost-effective crowdsourced entity resolution: A partial-order approach. *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 969–984, 2016.

[14] Z. Chen, Q. Chen, B. Hou, M. Ahmed, and Z. Li. Improving machine-based entity resolution with limited human effort: A risk perspective. In *Proceedings of the International Workshop on Real-Time Business Intelligence and Analytics*, pages 1–5, 2018.

[15] P. Christen. Automatic record linkage using seeded nearest neighbour and support vector machine classification. In *Proceedings of the 14th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 151–159, 2008.

[16] P. Christen. *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*, chapter 2, pages 32–34. Springer Science & Business Media, 2012.

[17] V. Christophides, V. Efthymiou, and K. Stefanidis. Entity resolution in the web of data. *Synthesis Lectures on the Semantic Web*, 5(3):1–122, 2015.

[18] X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 1247–1261, 2015.

[19] M. Cochinwala, V. Kurien, G. Lalk, and D. Shasha. Efficient data reconciliation. *Information Sciences*, 137(1-4):1–15, 2001.

[20] M. Ebraheem, S. Thirumuruganathan, S. R. Joty, M. Ouzzani, and N. Tang. Distributed representations of tuples for entity resolution. *Proceedings of the VLDB Endowment*, 11(11):1454–1467, 2018.

[21] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 19(1):1–16, 2007.

[22] W. Fan, X. Jia, J. Li, and S. Ma. Reasoning about record matching rules. *Proceedings of the VLDB Endowment*, 2(1):407–418, 2009.

[23] T. Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[24] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[25] D. Firmani, B. Saha, and D. Srivastava. Online entity resolution using an oracle. *Proceedings of the VLDB Endowment*, 9(5):384–395, 2016.

[26] L. Getoor and A. Machanavajjhala. Entity resolution: theory, practice & open challenges. *Proceedings of the VLDB Endowment*, 5(12):2018–2019, 2012.

[27] D. Gissin and S. Shalev-Shwartz. Discriminative active learning. *arXiv preprint, arXiv:1907.06347*, 2019.

[28] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. Shavlik, and X. Zhu. Corleone: Hands-off crowdsourcing for entity matching. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 601–612, 2014.

[29] A. Gruenheid, D. Kossmann, R. Sukriti, and F. Widmer. Crowdsourcing entity resolution: When is a=b? *Technical Report/Department of Computer Science, ETH Zurich*, pages 1–33, 2012.

[30] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1321–1330, 2017.

[31] D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, pages 1–12, 2017.

[32] D. Hendrycks, M. Mazeika, and T. G. Dietterich. Deep anomaly detection with outlier exposure. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, pages 1–18, 2019.

[33] B. Hou, Q. Chen, Z. Chen, Y. Nafa, and Z. Li. r-humo: A risk-aware human-machine cooperation framework for entity resolution with quality guarantees. *IEEE Transactions on Knowledge and Data Engineering (TKDE) (Early Access)*, pages 1–12, 2018.

[34] S.-J. Huang, R. Jin, and Z.-H. Zhou. Active learning by querying informative and representative examples. In *Advances in neural information processing systems*, pages 892–900, 2010.

[35] H. Jiang, B. Kim, M. Guan, and M. Gupta. To trust or not to trust a classifier. In *Advances in Neural Information Processing Systems*, pages 5546–5557, 2018.

[36] P. Kouki, J. Pujara, C. Marcum, L. Koehly, and L. Getoor. Collective entity resolution in familial networks. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 227–236, 2017.

[37] S. Lacoste-Julien, K. Palla, A. Davies, G. Kasneci, T. Graepel, and Z. Ghahramani. Sigma: Simple greedy matching for aligning large knowledge bases. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 572–580, 2013.

[38] G. Li. Human-in-the-loop data integration. *Proceedings of the VLDB Endowment*, 10(12):2006–2017, 2017.

[39] L. Li, J. Li, and H. Gao. Rule-based method for entity resolution. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 27(1):250–263, 2015.

[40] B. Mozafari, P. Sarkar, M. Franklin, M. Jordan, and S. Madden. Scaling up crowd-sourcing to very large datasets: a case for active learning. *Proceedings of the VLDB Endowment*, 8(2):125–136, 2014.

[41] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the ACM International*

Conference on Management of Data (SIGMOD), pages 19–34, 2018.

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

[43] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré. Holoclean: Holistic data repairs with probabilistic inference. *Proceedings of the VLDB Endowment*, 10(11):1190–1201, 2017.

[44] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1135–1144, 2016.

[45] R. T. Rockafellar and S. Uryasev. Conditional value-at-risk for general loss distributions. *Journal of banking & finance*, 26(7):1443–1471, 2002.

[46] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 269–278, 2002.

[47] R. Singh, V. Meduri, A. Elmagarmid, S. Madden, P. Papotti, J.-A. Quiané-Ruiz, A. Solar-Lezama, and N. Tang. Generating concise entity matching rules. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 1635–1638, 2017.

[48] P. Singla and P. Domingos. Entity resolution with markov logic. In *Proceedings of the IEEE 6th International Conference on Data Mining (ICDM)*, pages 572–582, 2006.

[49] G. Tardivo. Value at risk (var): The new benchmark for managing market risk. *Journal of Financial Management & Analysis*, 15(1):16, 2002.

[50] V. Verroios, H. Garcia-Molina, and Y. Papakonstantinou. Waldo: An adaptive human interface for crowd entity resolution. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 1133–1148, 2017.

[51] N. Vesdapunt, K. Bellare, and N. Dalvi. Crowdsourcing algorithms for entity resolution. *Proceedings of the VLDB Endowment*, 7(12):1071–1082, 2014.

[52] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. CrowdER: Crowdsourcing entity resolution. *Proceedings of the VLDB Endowment*, 5(11):1483–1494, 2012.

[53] S. Wang, X. Xiao, and C.-H. Lee. Crowd-based deduplication: An adaptive approach. *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 1263–1277, 2015.

[54] Z. Wang and J. Ye. Querying discriminative and representative samples for batch mode active learning. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 9(3):17, 2015.

[55] S. E. Whang, P. Lofgren, and H. Garcia-Molina. Question selection for crowd entity resolution. *Proceedings of the VLDB Endowment*, 6(6):349–360, 2013.

[56] S. E. Whang, D. Marmaros, and H. Garcia-Molina. Pay-as-you-go entity resolution. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 25(5):1111–1124, 2013.

[57] M. Wise. On normalizing the incomplete beta-function for fitting to dose-response curves. *Biometrika*, 47(1-2):173–175, 1960.

[58] J. Yang, J. Fan, Z. Wei, G. Li, T. Liu, and X. Du. Cost-effective data annotation using game-based crowdsourcing. *Proceedings of the VLDB Endowment*, 12(1):57–70, 2018.

[59] P. Zhang, J. Wang, A. Farhadi, M. Hebert, and D. Parikh. Predicting failures of vision systems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3566–3573, 2014.