Exploratory Training: When Annotators Learn About Data

RAJESH SHRESTHA, Oregon State University, USA
OMEED HABIBELAHIAN, Oregon State University, USA
ARASH TERMEHCHY, Oregon State University, USA
PAOLO PAPOTTI, EURECOM, France

Data systems often present examples and solicit labels from users to learn a target model, i.e., active learning. However, due to the complexity of the underlying data, users may not initially have a perfect understanding of the effective model and do not know the accurate labeling. For example, a user who is training a model for detecting noisy or abnormal values may not perfectly know the properties of typical and clean values in the data. Users may improve their knowledge about the data and target model as they observe examples during training. As users gradually learn about the data and model, they may revise their labeling strategies. Current systems assume that users always provide correct labeling with potentially a fixed and small chance of annotation mistakes. Nonetheless, if the trainer revises its belief during training, such mistakes become significant and non-stationarity. Hence, current systems consume incorrect labels and may learn inaccurate models. In this paper, we build theoretical underpinnings and design algorithms to develop systems that collaborate with users to learn the target model accurately and efficiently. At the core of our proposal, a game-theoretic framework models the joint learning of user and system to reach a desirable eventual stable state, where both user and system share the same belief about the target model. We extensively evaluate our system using user studies over various real-world datasets and show that our algorithms lead to accurate results with a smaller number of interactions compared to existing methods.

CCS Concepts: • **Information systems → Data cleaning**; *Data analytics*; • **Human-centered computing → User models**.

Additional Key Words and Phrases: data exploration; human in loop; human learning; hypothesis-testing model; bayesian model; functional dependencies; active learning

ACM Reference Format:
Rajesh Shrestha, Omeed Habibelahian, Arash Termehchy, and Paolo Papotti. 2023. Exploratory Training: When Annotators Learn About Data. *Proc. ACM Manag. Data* 1, 2, Article 135 (June 2023), 25 pages. https://doi.org/10.1145/3589280

1 INTRODUCTION

Data Systems often actively present examples and solicit labels and training examples from users to learn a target model, e.g., in *active learning* [5, 15, 23, 46, 47]. In each interaction, the system selects an example for labeling based on the training data already given by the user and the effectiveness of the current model. This improves the flexibility of choosing examples and may significantly reduce the amount of training data, which is arguably the most expensive resource in learning a model [5].

Authors' addresses: Rajesh Shrestha, Oregon State University, Corvallis, USA, shresthr@oregonstate.edu; Omeed Habibelahian, Oregon State University, Corvallis, USA, habibelo@oregonstate.edu; Arash Termehchy, Oregon State University, Corvallis, USA, termehca@oregonstate.edu; Paolo Papotti, EURECOM, Biot, France, papotti@eurecom.fr.

2836-6573/2023/6-ART135 $15.00
https://doi.org/10.1145/3589280

Due to the sheer volume and complexity of data, in many settings labeling is often exploratory: users have to explore and learn about the underlying data to label examples correctly. This often happens when the target models are complex or labeling an example requires knowledge about other examples in the dataset. Users learn about the data by repeatedly inspecting it to improve the knowledge and revise the hypotheses about the target model based on the results of preceding interactions. As an example, consider a user who prepares a dataset about patients for some downstream analysis. The goal is to clean the original dataset as this is known to be noisy, i.e., it contains erroneous values. In this task, the user may use an error detection system that needs training data, i.e., labeled examples of clean and noisy tuples, to build its internal model for error detection [24, 37, 50]. After checking a few records, the user starts labeling some values as incorrect as they report abnormally high values for one health indicator. These training samples steer the system to label tuples with high values for such indicator as likely to be mistakes. However, after some interactions, the user may find out that these "incorrect" values are for patients in an age range when those values are actually common. This new information makes some of the labels for the early iterations incorrect, as the user learnt about the data during the annotation process.

As another relevant scenario, users may also have to refresh their knowledge about the data and target model due to rapid and frequent data evolution. For example, consider a data security analyst who annotates data to train a model that detects suspicious activities in large evolving log data. She may discover new types of suspicious activities after exploring a sufficiently large subset of the data every time she plans to (re)train the model.

As users explore and interact with the data, their knowledge about the data and the target concept evolves. Thus, they may modify their labeling and training strategies. It is known that human learning in interactive settings is highly non-stationary [11, 20, 40, 41, 49, 54]. For example, they may have a prior belief (distribution) about the properties of positive or negative examples for the target concept based on their background information and expertise. They may gradually revise their belief, and in turn labeling and training strategies, after observing new data items, to then settle on a relatively fixed strategy after gaining sufficient knowledge about their actions and environment [11, 20, 41, 49].

Current active learning systems often assume that users have perfect knowledge about the target model and underlying data. They usually assume that users provide reliable feedback or training data for target insights with potentially a very small and fixed amount of noise, i.e., stationary user model [5, 15, 21, 33, 48, 55, 56]. However, as users' knowledge about the data and insight may significantly change during the analysis, it is not clear which inputs are sufficiently reliable for predicting the target models. Due to the significance and non-stationarity of errors, current systems use incorrect labels and learn inaccurate models. Stationary learners are too biased to the initial observations and fail to adapt subsequently. Thus, current methods may learn a model mainly based on early interactions and not use users' updated knowledge during training.

As an alternative approach, users may first thoroughly explore the data to form their final belief about the target model and then annotate examples to train the model. This is, however, very time consuming over large and complex datasets. Also, users have to decide when to stop such exploration based on their own intuitions. Since users may not be sufficiently familiar with the underlying data and target model, it is not clear whether their decisions on stopping the exploration will be correct. Hence, after such data exploration, users might need to stop training in the middle, further explore the data, and restart training from the beginning. This, in turn, may considerably increase the users' annotation effort.

Thus, exploratory training leads to two main challenges. For the active learning system, it is challenging to understand or find reliable users' inputs in the interaction to learn an effective model for their target concepts. For the human, the examples are selected by the system without

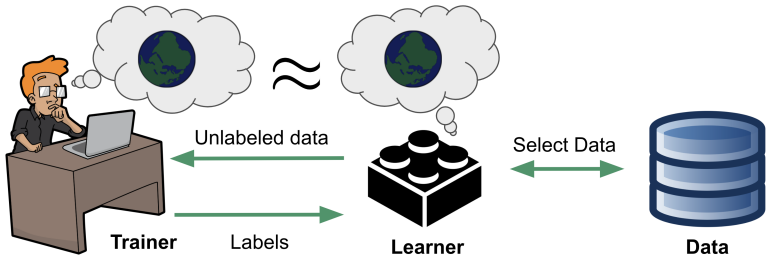


Fig. 1. High Level Diagram of Exploratory Training

considering the user learning. Both challenges may lead to long interactions with learning inaccurate models.

We argue that, to support users in the best way possible, we should enable them to develop accurate understanding of the target model during training by building learning methods that are aware of how humans learn. In this paper, we set forth novel principles and methods to build systems that enable effective exploratory training. Our key idea is to redesign current learning methods to recognize and adapt to users' learning effectively. We leverage the extensive body of work on human learning in psychology and experimental economics to model users' learning and reasoning during training [11, 20, 41, 49, 54]. As depicted in Figure 1, this novel approach views exploratory training as the interactive game with identical interest of two learning agents, i.e., user (as trainer) and system (as learner), for achieving the common goal of learning users' desired models. We leverage the tools in the field of learning in games to develop and analyze our proposed learning methods. The problem of exploratory training may appear in various context. In this paper, we focus on using exploratory training for learning beliefs (expressed as functional dependencies) over datasets that contain dirty tuples. Particularly, our contributions are as follows.

1. We develop the theoretical underpinnings for exploratory training and interactive joint learning of users and learning systems (Section 3).
2. We study models of human learning during exploratory training validated by extensive user studies (Section 4).
3. We introduce novel algorithms for the system to quickly and effectively adapt to user learning (Section 5).
4. We evaluate the proposed methods through extensive empirical studies over real-world data (Section 6).

2 USE CASE: APPROXIMATE FD DISCOVERY

Our approach is independent from the target application and from the internals of the learning system as long as such system, i.e., learner, exposes examples and consumes labels. In this paper, we illustrate our approach with a system that aims at learning approximate functional dependencies (FDs) [31].

Approximate FD Discovery over Real-world Data. We focus on the challenging case of a learner finding approximate FDs over *a dataset that may contain dirty tuples*. If the dataset is completely clean and does not contain any dirty/erroneous tuple, its set of approximate FDs can be learned with an unsupervised method [29, 30]. Nonetheless, real-world datasets often contain dirty tuples. In this case, the learner requires some supervision to accurately learn approximate FDs [1]. This requires user input to find out whether a violation of an FD in the data is due to some erroneous tuple. In other words, users should provide the learner with sufficiently many labeled tuples, i.e., annotated as clean or dirty.

	Player	Team	City	Role	Apps
t_1	Carter	Lakers	L.A.	C	4
t_2	Jordan	Lakers	Chicago	PF	4
t_3	Smith	Bulls	Chicago	PF	4
t_4	Black	Bulls	Chicago	C	3
t_5	Miller	Clippers	L.A.	PG	3

Table 1. Sample instance of a dataset D.

Users' Input. In this kind of interaction, the user only annotates examples, without providing (approximate) FDs explicitly. Users provide labels according to their beliefs on approximate FDs, which is not observed by the learner directly. We conduct a user study (Section 4) to analyze how users, i.e., trainers, learn about FDs while providing annotated examples interactively. To evaluate different models of users' learning and how their beliefs improve over time, we gather the ground truth on users' beliefs. Thus, in addition to annotated examples, we ask participants to provide the set of FDs they believe to be the most accurate in each interaction. This is limited to our user study with the aforementioned goal. In our framework and algorithms, we assume that trainers provide the learner only with annotated examples as depicted in Figure 1.

Applications of Approximate FDs. This learned approximate FDs can be used for detecting errors in unlabeled or future tuples [1, 2, 13, 16, 17, 36, 43, 52]. In this paper, we focus on learning and discovering approximate FDs. The study of new algorithms to make the best use of these learned models is out of the scope of this paper.

Why Approximate FDs. For our learner module, we focus on declarative rules, like FDs, rather than a "black-box" ML method for two main reasons. First, many popular models in data management, e.g., data cleaning systems, use declarative rules. Second, as explained above, to conduct a user study and investigate the connection between users beliefs and the labels they provide, one may have to solicit the participants to provide both labeled examples and their belief about the accurate model in each interaction. It is more convenient for humans to describe their beliefs in form of (simple) declarative rules, such as FDs, rather than other models.

2.1 Approximate Functional Dependencies

Given a relation r over a schema R with attribute sets $X, Y \subseteq R$. We represent the projection of a tuple t to a set of attributes X as $t[X]$. We denote with $X \rightarrow Y$ an FD with left-hand side (LHS) X and right-hand side (RHS) Y if we have $t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$ for all pairs of distinct tuples $t_1, t_2 \in r$. An $X \rightarrow A$ is minimal if no subset of X determines attribute A. We focus our study to FDs that are minimal, non trivial, i.e., $X \cap Y = \emptyset$, and normalized, i.e., the RHS is a single attribute.

Due to the presence of errors, in most real datasets FDs are rarely completely satisfied by the data. *Exact* FDs can therefore be relaxed to allow a certain degree of violation. We refer to such relaxed FDs as *approximate* FDs. To quantify the degree of approximation, we use the g_1 measure in its scaled version, which represents the percentage of violating pairs of tuples.

$$g_1(X \rightarrow A, r) = \frac{|\{(t_1, t_2) | t_1, t_2 \in r, t_1[X] = t_2[X], t_1[A] \neq t_2[A]\}|}{|r^2|}$$

Example 2.1. Consider dataset D in Table 1. Let f_1 be the approximate FD $Team \rightarrow City$. The pair of tuples t_3 and t_4 satisfy the FD, but tuples t_1, t_2 do not. The g_1 for f_1 over D is therefore $\frac{1}{25} = 0.04$.

FD discovery is the problem of finding all minimal, nontrivial functional dependencies that hold in a given relation.

2.2 Detecting Errors

Every pair of tuples involved in G_1 represents a *violation* of the given FD. In the data cleaning literature [6], violations are also identified at a finer granularity, i.e., at the cell value level. A violation is defined by the set of attribute cells for X and Y over the two violating tuples. Given a relation r, the set of all violating cells that can be determined through FDs over T is denoted by C_v.

Given one or more violations, there is plethora of algorithms that, given a set of FDs, try to identify the tuples (or cells) that contain erroneous values [2, 13, 36, 43]. As we focus on approximate FDs, any algorithm for error detection in the presence of approximate FDs can be used in our framework. Intuitively, for an FD f over r and a pair tuples in r that satisfy f, we convert the scaled g_1 measure m of f into the probability of being dirty for the two tuples. For a violating pair of tuples, we consider $1 - m$ as the probability of being dirty for the two tuples. To handle multiple FDs, in our implementation we adopt a simplified version of the probabilistic approach in Holoclean [43], where we approximate the probability of tuples of being dirty, rather than cells.

Example 2.2. Consider again dataset D in Table 1. Given approximate FD f_1 over Team and City, t_1 and t_2 are in violation as they satisfy f_1 LHS and do not satisfy its RHS. They are assigned a 0.96 probability of being dirty. Indeed, at least one of the tuples contains an erroneous cell value, but this is not identified by the FD itself.

3 GAME-THEORETIC FRAMEWORK

The trainer and learner interactively collaborate to learn the accurate target model (Figure 1). Before the interaction starts, both trainer and learner hold *prior beliefs* about the accurate target model and data. In each interaction, first, the learner selects a set of examples from the data and presents them to the trainer. The learner may select these examples based on its belief, e.g., picks examples that helps it learn the target model faster than others. The trainer may modify its (prior) belief about the data and target model after observing such examples, i.e., the trainer's belief may evolve over time as it learns more about the data with exposure to more samples. The trainer labels the presented examples according to its potentially updated belief. Using these labeled examples, the learner updates its belief about the target model. If the learner deems that it requires more labeled examples, it starts another interaction by selecting and presenting another set of examples to the trainer. The goal of both agents is for their beliefs to convergence to the same accurate belief about the target model with fewest interactions as possible. Next, we formalize this interaction as a collaborative game.

3.1 Agents, Actions and Policies

The game of exploratory training, (*ET* for short) has two agents: trainer, i.e., user, and active learner (learner for short). The game is a sequence of interactions at discrete times $t > 0$. Each interaction t starts by the learner presenting a given fixed number k of tuples, i.e., examples, from the dataset to the trainer. The trainer labels these examples according to its current knowledge about the data and the target model, i.e., the model of inconsistent data. Hence, in each interaction, the *actions of learner* are (to pick) k tuples from the dataset and the *actions of trainer* are (to deliver) k labeled tuples in which each tuple is assigned a label from set $\{0, 1\}$, i.e., *clean* or *dirty*. We assume that the learner provides a fresh tuple in each interaction that has not been presented to the trainer in any preceding interaction. We call each example and its label a *labeling*.

The *policy* of an agent in each interaction decides which actions the agent performs in that interaction. More precisely, a policy of the trainer in interaction t is a mapping from the set of examples to the set of labels, which assigns a single label to each example. We denote this policy as $\pi_t^{\mathcal{T}}$. The policy of the learner in interaction t, shown as $\pi_t^{\mathcal{L}}$, is a probability distribution over the

set of examples, where the probability of being dirty of each example indicates the chances that it will be presented to the trainer in interaction t.

Example 3.1. Consider again dataset D in Table 1. The learner's policy indicates that tuples with ids $t_1, \ldots, t_4$ have higher probability of tuple t_5 of being dirty. Assuming k=4, the first four tuples are exposed to the trainer for annotation as a batch. The trainer's policy leads to label clean (0) for t_1, t_3, t_4 and dirty (1) for tuple t_2.

3.2 Payoff and Belief

The objective of the trainer policy in each interaction is to provide accurate labeling and the goal of the learner is to use these labels to build an effective target model. Thus, both agents should prefer policies that result in accurate labels in each interaction, i.e., such policies should receive *higher payoff (utility).* As none of the agents knows the accurate labels, at least in early interactions, they may not be able to evaluate precisely the payoff of their policies. They have to rely on *prior beliefs* about the data or *observations from past interactions* to estimate the payoff of policies.

Next, we define beliefs and payoffs for both agents.

3.2.1 Beliefs. Due to their incomplete information about accurate labelings, the agents evaluate the ability of a policy to result in accurate labels based on their current *beliefs* about the target model. The trainer and learner maintain their own beliefs about the model. Let $\mathcal{T}$ and $\mathcal{L}$ denote the trainer and learner agents, respectively. Every agent $i \in \{\mathcal{T}, \mathcal{L}\}$ maintains a *belief* θ_t^i about the target model in interaction t. Given an example, each belief provides a probability distribution over its possible labels, where the probability of each label reflects the agent's confidence in its accuracy. Both agents start the interaction with a *prior belief* θ_0^i. The learner presents and trainer labels examples based on their beliefs.

To improve the computational efficiency of learning, learners usually limit their beliefs to the hypotheses from a fixed set called the hypothesis space [39]. As explained in Section 2, we assume that each hypothesis is a set of probabilistic functional dependencies.

Example 3.2. Let us assume that D has more tuples (100), with $g_1 = 0.3$, and the belief of the learner is that Team implies City (f_1: T $\rightarrow$ C) holds with probability 0.7 and that Player is a key (f_2: P $\rightarrow$ T, C, R, A) with probability 1. All tuples satisfy f_2. For f_1, t_1 and t_2 satisfy its LHS and do not satisfy its RHS, so the belief of being dirty is of 0.7 (with clean probability of 0.3). Tuples t_3 and t_4 satisfy f_1 and have probability of being dirty of 0.3 (clean 0.7) as they do not violate the FD.

The trainer has a belief over the tuples because of its prior over the values in the tuple: t_1 is dirty with probability 0.2, while t_2 is dirty with 0.8. Both tuples t_3 and t_4 are dirty with probability 0.2.

3.2.2 Payoff of Trainer Policy. The *payoff of each labeling* (x, y) by the trainer in interaction t is the probability by which $\theta_t^{\mathcal{T}}$ deems y to be the accurate label for x, i.e., $\theta_t^{\mathcal{T}}(y \mid x)$. The *payoff of a policy* $\pi_t^{\mathcal{T}}$ in interaction t is the sum of payoffs of its labelings. More precisely, it is $u_T(\theta_t^{\mathcal{T}}, \pi_t^{\mathcal{T}}) = \sum_x \theta_t^{\mathcal{T}}(\pi_t^{\mathcal{T}}(x) \mid x)$ where x goes over all examples shown in interaction t.

Example 3.3. The payoff of the trainer is the probability from the belief; the payoff for the labelings in Example 3.2 is 0.2 (t_1 dirty) + 0.8 (t_2 dirty) + 0.2 (t_3 dirty) + 0.2 (t_4 dirty).

3.2.3 Payoff of Learner Policy. The goal of the learner is to form a belief that is in agreement with the trainer's and predicts the labels provided by the trainer accurately. Thus, the closer the learner's belief is to the trainer's labelings in an interaction, the higher the payoff of the learner should be. Let the trainer assign label y to the presented example x in interaction t. This example receives the payoff of $\theta_t^{\mathcal{L}}(y \mid x)$, which is the probability that learner's belief predicts the label of x to be y. We define the payoff of a learner's policy according to the probability by which each example is

selected in the policy. Let $\pi_t^{\mathcal{L}}$ be the policy of learner in interaction t and $u_a(\theta_t^{\mathcal{L}}, \pi_t^{\mathcal{L}})$ denote such payoff. The payoff $u_a(\theta_t^{\mathcal{L}}, \pi_t^{\mathcal{L}})$ is $\sum_x \theta_t^{\mathcal{L}}(y \mid x)\ \pi_t^{\mathcal{L}}(x)$ where y is the label of x and x goes over all presented examples in interaction t. It is the expected value of the payoff of every labeled example in interaction t computed over the probability distribution of $\pi_t^{\mathcal{L}}$.

Example 3.4. The learner starts the interaction by showing examples to the trainer. Assume a policy that picks tuples at random, therefore every tuple (over 100 in D) is picked with probability 0.01. The payoffs of the learner for the four tuples are: t_1 gets 0.7 (probability of being dirty because of being in an FD violation) times 0.01; t_2 gets 0.7 times 0.01; t_3 gets 0.3 (probability of being dirty because not in a FD violation) times 0.01; t_4 gets 0.3 times 0.01.

Assuming a different policy that picks tuples that are likely to be dirty with higher probability, the payoffs of the learner for the four tuples are: t_1 gets 0.7 times 0.1; t_2 gets 0.7 times 0.1; t_3 gets 0.3 times 0.001; t_4 gets 0.3 times 0.001. Thus the payoff would be higher with this policy.

The payoff function $u_a(.)$ encourages the learner to build a belief over all the hypotheses that is consistent with and accurately generalizes to the (future) trainer's labeling. Nevertheless, it may lead the learner to build a model that is biased toward a subset of the data and present examples only from that subset. For instance, assume a learner has a belief that accurately predicts labels for a small subset of examples from interaction t onward. It can use a policy that puts 0 probability on the examples outside of this subset and receive maximum payoff for each interaction after t. Such a learner does not have any incentive to learn a model that accurately predicts the labels of examples outside the subset. Thus, all other conditions being the same, policies that provide more varieties of examples may get higher payoff. Moreover, as explained before, the more *representative* and *informative* the presented examples are, the more knowledge the trainer may achieve about the data.

There are multiple methods to measure the coverage and representativeness of samples from a dataset. We use *entropy* of the learner's policy in each interaction to quantify such properties for its examples. Let $u_L(\theta_t^{\mathcal{L}}, \pi_t^{\mathcal{L}})$ denote the total payoff of the learner policy $\pi_t^{\mathcal{L}}$ in interaction t. We have $u_L(\theta_t^{\mathcal{L}}, \pi_t^{\mathcal{L}}) = u_a(\theta_t^{\mathcal{L}}, \pi_t^{\mathcal{L}}) - \gamma \sum_x \pi_t^{\mathcal{L}}(x) \ln \pi_t^{\mathcal{L}}(x)$ where x is an example and $\gamma > 0$ is a real number that sets the balance between two aspects of payoff. The lower the value of γ is, the less exploratory the selected set of examples by the learner are. The parameter γ expresses both the lack of knowledge of the trainer about the data and the lack of confidence of the learner about its belief. The trainer may setup the value of γ according to its background information about the data.

Example 3.5. With an incentive for exploration, tuples involved only in f_2 (the key with $g_1 = 0$), such as t_5, can be picked by the learner policy. This is done even if t_5 has a zero probability of being dirty, according to the FDs, and may change the learner belief if the trainer marks it as dirty, i.e., key constraint f_2 is not exact.

3.3 Equilibria

For trainer and learner, the goal is to learn and maintain the same accurate belief about the target concept. The interaction should lead to a stable state belief-wise in which both agents share the same accurate belief and none of the agents has any incentive to modify its belief. The stable states, i.e., equilibria, of games are conventionally defined over policies of agents. Formally, a *Nash equilibrium* (equilibrium for short) of the game is a pair of policies ($\pi^{\mathcal{T}}$,$\pi^{\mathcal{L}}$) where $\pi^{\mathcal{T}}$ and $\pi^{\mathcal{L}}$ are the policies of trainer and learner, respectively, such that none of the agents can change its policy individually and increase the payoff of the game.

Nonetheless, the stability of policies per se is not the final goal of our agents. Thus, we define the equilibria of the game according to agents' common beliefs. Before formally defining this state,

we note that since the response strategy of each agent is a function of its belief, if an agent changes its belief, it may change its policy in the subsequent interaction(s). This, in turn, may modify the payoff of the game. The *belief-wise equilibrium* of the game is a pair of beliefs $(\theta^{\mathcal{T}}, \theta^{\mathcal{L}})$ such that none of the agents can modify its belief and increase the payoff of the game. Since the response strategy of each agent is a function of its belief, the policies of agents are fixed and stable in a belief-wise equilibrium. Hence, each belief-wise equilibrium is also an equilibrium for the game. Unless otherwise noted, we use the term equilibrium to refer to belief-wise equilibrium of the game.

As explained in Section 3.2, because none of the agents have complete information about accurate labeling, they cannot evaluate the payoff of the game in some interactions effectively and have to use their current belief to estimate the payoff. Thus, the game may have equilibria other than the ideal one in which both agents hold the accurate belief, i.e., *accurate equilibrium*. If agents maintain the same inaccurate belief, they may estimate the payoff of the game incorrectly and do not modify their belief and policies. As an extreme example, if the learner presents biased samples of the data to the trainer, the trainer may learn an inaccurate belief about the target model. This may reinforce the learner's belief and causes both agents to maintain the same wrong belief.

Example 3.6. In an equilibrium, the beliefs of both the learner and the trainer is that f_1 (Team $\rightarrow$ City) holds with probability 0.9 and that f_2 (key) holds with probability 1.

3.4 Interactive Learning and Adaptation

Agents leverage the information gained in each interaction to gradually update and improve the accuracy of their beliefs according to some learning method. Improved beliefs in turn guide agents to choose policies with higher payoffs and to learn collaboratively an accurate target model. That is, with accurate beliefs, the trainer provides more effective labelings and the learner presents more informative examples whose labeling may lead to learning an accurate model faster. A desired property of learning methods for trainer and learner is to guide the interaction to an accurate equilibrium.

Each learning method consists of two components: a *prediction model*, P, that updates the agents' belief and a *response model*, R, using which the agent selects its policy based on the updated belief [54]. Let the *history* of interactions up to time $t \geq 0$, denoted as h_t, be the sequence of labelings $(x, y)_0 \ldots (x, y)_{t-1}$ of interactions 0 to $t-1$. The prediction model of agent i, shown as P^i, is a stochastic mapping from h_t and current belief θ^i to the set of agent's updated beliefs. The response model of agent i, denoted as R^i, is a stochastic mapping from the set of its beliefs to the set of its policies.

3.4.1 Trainer's Prediction & Response Models. Suppose the trainer's belief is $\theta^{\mathcal{T}}_{t-1}$ at interaction $t-1$. This belief is a result of (i) the trainer's prior belief before observing any data and (ii) the samples of data that were presented to the trainer till time $t-1$. Let X_t denote the set of examples presented to the trainer at interaction t. After observing X_t, the trainer updates its belief based on how much X_t accords or contrasts with $\theta^{\mathcal{T}}_{t-1}$. This updated belief of the trainer is a function of $\theta^{\mathcal{T}}_{t-1}$ and previous observed examples $X_1, X_2, \ldots, X_t$ as

$$\theta^{\mathcal{T}}_t = P^{\mathcal{T}}(\theta^{\mathcal{T}}_{t-1}, \mathbf{X^1}, \mathbf{X^2}, ..., \mathbf{X^t})$$

In interaction t, the trainer labels every example $x_t \in X_t$ with labels y_t based on policy $\pi^{\mathcal{T}}_t = R^{\mathcal{T}}(\theta^{\mathcal{T}}_t)$.

3.4.2 Learner Prediction & Response Models. The learner uses its predictive model to predict the trainer's current belief. Let the trainer provide a set of labeled examples Y_t in interaction t. Using its prediction model, the learner uses Y_t and updates its belief to match the trainer's belief such that

the learner can predict future labeling as accurately as possible. We have $\theta_t^{\mathcal{L}} = P^{\mathcal{L}}(\theta_{t-1}^{\mathcal{L}}, \mathbf{X^t}, \mathbf{Y^t})$. The learner selects and shows examples from the underlying dataset in interaction t using policy $\pi_t^{\mathcal{T}}$. Before showing examples, the learner updates it policy using its response model as $\pi_t^{\mathcal{T}} = R^{\mathcal{L}}(\theta_t^{\mathcal{L}})$.

Example 3.7. The trainer is exposed to the four tuples, selected by the learner in Examples 3.4, and its prediction model updates its belief. Now t_1, t_3 and t_4 are considered dirty with probability 0.1, while t_2 is dirty with probability 0.9. The response model selects a policy that leads to mark tuple t_2 as an error and therefore the violation defined by t_1, t_2 as correct.

Given the labels from the trainer, the learner prediction model updates its belief by decreasing g_1 for f_1 to 0.05 (from the original 0.3). Therefore, for the policy that picks tuples at random, the payoff over the remaining four tuples becomes: t_1 gets 0.05 times 0.01; t_3 gets 0.05 times 0.01; t_4 gets 0.05 times 0.01; t_5 gets 0.05 times 0.01.

In the next section, we discuss and report results on how humans learn, i.e., form their belief, in an interactive setting. Then, in Section 5, we use such learning schemes to develop effective prediction and response models for the learner.

4 HOW DO HUMANS LEARN TO TRAIN?

If we know the schemes by which humans learn from data to revise their beliefs, we may design an effective prediction and response models for the learner. In this section, we leverage the body of research on human learning in interactive environments [19, 20, 45, 49, 54] and extensive user studies to find the learning methods by which human trainers revise their beliefs during the validation of violations detected using approximate FD discovery. In particular, our goals are to find out whether humans trainers in this setting learn; and which known learning method models the trainers' learning most accurately.

4.1 Models of Human Learning

Researchers have generally categorized human learning in interactive games as *model-based* learning, i.e., belief learning, and *model-free*, i.e., reinforcement-based [20, 49, 54]. We have explained model-based learning methods in Section 3.4. In model-free methods, the agent updates its response strategy without constructing any belief about the state of the game. The agent directly reinforces policies based on the observed payoff in past interaction and selects the ones with the largest reinforcement value. These methods are appropriate for cases where agents do not form any belief about the setting or other agents in the game. They better model the cases where agents have lower degree of rationality, domain expertise, or prior knowledge than the ones that use model-based methods.

We believe that model-free methods do not accurately model the learning of human trainers in our setting. First, due to the nature of FD discovery and FD-based error detection, users who train the models ought to construct beliefs about the FDs and characteristics of clean data. Second, there is not any direct observed payoff in interactions, therefore, trainers have to form and rely on their beliefs to estimate payoff of policies. Third, in our setting, users are domain experts and usually form and maintain beliefs about data. Thus, we focus on prominent model-based learning methods: *fictitious play*, *Bayesian learning*, and *hypothesis testing* [49, 54].

4.1.1 Fictitious Play. Fictitious play (FP for short) is a fundamental model-based learning method in interactive games. In FP, the agent assumes the other agent's strategy does not change in the course of interaction, i.e., it is stationary. Its belief about the other player is a probability distribution on the other player's actions. The agent updates the probability of each action in its belief using the observed empirical frequencies of that action so far. In each interaction, it picks a strategy

that delivers the highest payoff assuming the other players performs the action with the highest probability according to the belief. The agent may have a prior belief before starting the interaction, which represents its domain knowledge. In the absence of domain knowledge, the agent may use an uninformative uniform prior. As FP is simple and requires relatively less degree of rationality, information, and computational resources, it is a popular method to model human learning.

Next, we explain how a trainer, which follows FP, learns to discover FDs. Let E be a minimal cover of FDs over the given dataset [3]. Let the trainer believe that FD $f \in E$ holds in the data with probability $\frac{p_f}{p}$ before the interactions start. Assume that the trainer observes k tuples up to interaction t such that only k_f pairs of them follow f. Following FP, the trainer updates its belief p_f using the observed frequencies k_f and k. For instance, if the trainer's prior belief for FD f is $\frac{1}{2}$ and we have $k = 4$ and $k_f = 8$, the updated belief is $\frac{1+8}{2+(4\times3)} = \frac{9}{14}$. The trainer updates its belief after observing the (fresh) tuples in each interaction. Then, the trainer labels the samples in the interaction according to its belief on how likely they are dirty. These annotations reflect the trainer belief over the FDs in E, which is then consumed by the learner.

4.1.2 Bayesian Learning. In Bayesian learning, the agent maintains a prior distribution over the likelihood of performing actions by the other player. After each interaction, it observes the other agent's actions and modifies its belief using Bayesian updating. As opposed to FP where a belief is a probability distribution over actions of the other player, a belief in Bayesian learning is a distribution over the probabilities (parameters of probability distributions) of actions of the other player. Similar to FP, in each interaction, the agents picks the action or strategy that delivers the highest payoff given its belief.

Consider again a minimal cover of FDs E over the dataset. The belief of the trainer about each FD $f \in E$ is a probability distribution over θ_f, i.e., the confidence of f. Let $h(\theta_f)$ be the prior belief of the trainer about f before the interaction starts. Let $g(\theta_f \mid X)$ denote the updated belief, i.e., posterior, of the trainer about f after it observes the set of tuples X in the first interaction. The update is done as $g(\theta_f \mid X) = \frac{l(X|\theta_f)}{\int l(X|\theta_f)h(\theta_f)d\theta_f} h(\theta_f)$ where l is the likelihood function that represents the probability of generating X given θ_f. This process of Bayesian updating will continue in the subsequent interactions as the posterior of the trainer after each interaction becomes its prior for the next interaction.

The likelihood function expresses the probability of how many pair of tuples in X comply with f given its confidence θ_f. Given that tuples in X are independent, the likelihood function follows binomial distribution with parameters $\mid X \mid (\mid X \mid -1)$ and θ_f, where $\mid X \mid (\mid X \mid -1)$ is the number of pairs of tuples in X. The probability that c pairs of tuples in X comply with f is $\binom{|X|(|X|-1)}{c} \theta_f^c (1-\theta_f)^{|X|(|X|-1)-c}$. For computational efficiency and convenience, we assume that the prior is in form of beta distribution, which is the conjugate prior of binomial distribution. Let the initial prior $h(\theta_f)$ be a beta distribution with parameters α_f and β_f as $\frac{\theta_f^{\alpha_f-1}(1-\theta_f)^{\beta_f-1}}{B(\alpha_f,\beta_f)}$. Let c and v denote the number of pairs of tuples in X that comply with and violate f, respectively. Based on Bayesian rule and properties of beta distribution, the posterior $g(\theta_f \mid X)$ will be a beta distribution with parameters $\alpha_f + c$ and $\beta_f + v$ [18, 22]. Given its belief distribution over θ_f after observing X, the trainer labels tuples in X based on the expectation of its belief, which is $\frac{\alpha_f+c}{\alpha_f+c+\beta_f+v}$. The expectation of the prior distribution is $\frac{\alpha_f}{\alpha_f+\beta_f}$. Interestingly, this is equivalent to the labeling according to FP if the prior probability (belief) for θ_f is $\frac{\alpha_f}{\alpha_f+\beta_f}$ [19]. Thus, in our setting, FP and Bayesian method are equivalent. We use FP and Bayesian in this paper interchangeably.

4.1.3 Hypothesis Testing. In hypothesis testing, the agent starts with an initial hypothesis and belief on how the other agent will act in the interaction. The agent frequently evaluates the performance of its belief and rejects the current one if it does *not* explain sufficient amount of recent data with some tolerance. If the current hypothesis is rejected, the agent picks another hypothesis based on its relative performance on the recent data. This is a popular model in both economics and cognitive psychology [54]. It somewhat resembles hypothesis testing in statistics.

More precisely, the trainer starts with an initial FD model for the data. It checks the validity of its current model every f interactions using the data it has observed in the last w interactions. If the accuracy of the current model over the data is greater than the tolerance threshold τ, the trainer keeps the model. Otherwise, it picks the FD model that explains the current data most accurately.

4.2 User Study Setup

4.2.1 Users & Interface. Our user study involves a group of 20 undergraduate and graduate students with different background in Computer Science. We have implemented a user interface that informs users about FDs, FD violations, and tests their knowledge. Then, the user is presented with the dataset schema and asked what they believe is the FD(s) that holds over the dataset with the fewest exceptions. We use this model as the prior belief of the users about the dataset in the methods described in Section 4.1. The user also has the option of indicating that they are not sure what the most accurate initial model is, in which case we use a uniform prior. Afterwards, the interface iteratively presents examples to the users for marking FD violations.

In each iteration, our system shows a random sample of ten (10) tuples from the dataset to the user. The user can mark violations of their hypothesized FD(s) in the presented examples according to their belief in that interaction. We instructed the users to label errors detectable by FDs. After marking violations in an interaction, the users specify their current hypothesized FD(s). Hence, we have direct access to the user belief in each iteration during user study, which enables us to measure the accuracy of human learning schemes effectively - such FDs are not fed to the learner. Each user must interact for at least 9 iterations and can continue up to 15 iterations for each scenario.

Our goal is to model how user interactively learn the FDs that hold over the data. It is time-consuming to ask users to specify models with multiple FDs in every interaction, this could discourage users from participating in the study. Hence, we ask users to specify explicitly the FD that they deem is hold over the observed data so far most accurately and label the presented samples accordingly.

4.2.2 Datasets & Scenarios. We use two real-world datasets: *AIRPORT* (Alaska Airport Data from *www.kaggle.com/datasets/, jamestollefson/alaskaairfields*) describes airports, heliports, and seaplane bases throughout the U.S. state of Alaska, while *OMDB* (Open Movie Database from *www.omdbapi.com*) contains information about English-language movies and TV shows. We use these datasets to cover both relatively familiar, i.e., movies, and unfamiliar domains to users in the study. We design five scenarios based on these two datasets whose information is shown in Table 2. In each scenario, we ask the participants to identify the FD(s) that hold over the underlying data with the fewest violations (exceptions), i.e., *target FD(s)*. Table 2 illustrates examples of other FDs in each scenario, i.e., *alternative FD(s)*. The participants may believe some of these FDs may hold over the dataset with fewest exceptions, e.g., given the schema and an initial looks at the data. The exact attributes present in each dataset vary slightly by scenario.

We introduce violations to each dataset in every scenario with an error generation tool that scrambles values w.r.t. the target FD [6]. The *violation ratio* of $\frac{m}{n}$ is to introduce n violations in every alternative FD per each m violations in the target ones. We use ratios of $\frac{1}{3}$ and $\frac{2}{3}$ for the first three and last two scenarios, respectively. We use different ratios to investigate the impact of the

#	Domain	Attributes	FDs
1	Airport	facilityname, type, manager	**Target**: *(facilityname, type)* ⇒ *manager* **Alternative**: *facilityname* ⇒ *(type, manager)*
2	Airport	sitenumber, facilityname, owner, manager,	**Target**: *sitenumber* ⇒ *(facilityname, owner, manager)* **Alternative**: *facilityname* ⇒ *(sitenumber, owner, manager)*
3	Airport	facilityname, owner, manager,	**Target**: *manager* ⇒ *owner* **Alternative**: *facilityname* ⇒ *(owner, manager)*
4	OMDB	title, year genre, type	**Target**: *(title, year)* ⇒ *(type, genre)* **Alternative**: *title* ⇒ *(year, type, genre)*
5	OMDB	title, rating, type	**Target**: *rating* ⇒ *type* **Alternative**: *title* ⇒ *(rating, type)*

Table 2. Scenarios used in the users study

relative quantity of violations on human learning. The smaller the violation ratio is, the easier it may be for the participant to pinpoint the target FD(s). We use the smaller ratios for the scenarios from the *AIRPORT* domain as it is relatively less familiar for the participants.

We randomize both the order that the scenarios are presented to participants and the arrangement of the attributes in the samples presented to participants for each scenario. This ensures that our findings are agnostic to order of presentation.

4.2.3 Configuration of Learning Methods. We use 20% of our collected information to tune the hyper-parameters of different methods as follows. Given the prior belief of the user about the most accurate FD, we build a prior beta distribution for that FD to be used as the prior in the FP and Bayesian method. We set its mean to $\epsilon = 0.85$. We call FD $X \rightarrow Z$ a *superset* of $XY \rightarrow Z$ where X, Y, Z are non-empty sets of attributes. FD $XY \rightarrow Z$ is a *sub-set* of $X \rightarrow Z$. If FD f_1 is a super-set of FD f_2, f_2 is implied by f_1. If f_1 is a sub-set of f_2, they are semantically close. Thus, we use two types of prior configurations for evaluating the learning methods. In the first one, we set the mean value of the prior distributions for all FDs other than the one specified by the user to the same value, which is set to 0.15. In the second one, we treat the prior distributions of FDs that are superset or subset of the one specified by the user differently than other FDs. We set their mean values to 0.8. We use the same prior for non-minimal FDs entered by the users, e.g., $XY \rightarrow YZ$. The standard deviation of all prior distributions are set to the same value of 0.05.

We set the initial values of α and β using the mean and variance equations of beta distributions to $\mu = \frac{\alpha}{\alpha+\beta}$ and $\sigma^2 = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$.

For our hypothesis testing learning method, we use the same model space and initial model setting as the one used in FP and Bayesian method. We set the testing frequency to every interaction. In our experiments, hypothesis testing have performed best over the majority of our training data when the users test their hypothesis using the samples presented in the preceding interaction. If the user rejects the current model given the aforementioned subset of tuples, they pick the model that performs the best over this subset.

4.2.4 Evaluation Metric. The aim of all learning methods is to predict the user specified FD accurately based on the labeling provided by the user. To measure the accuracy of prediction for each method, we first sort the top-k predicted FDs by the method in each interaction based on their confidence. Let p be the position of the user specified FD, i.e., ground truth, in the list. The **Reciprocal Rank** (RR) of the method is $\frac{1}{p}$. We use the mean of RR values, i.e., **Mean Reciprocal Rank** (MRR), across all interactions to measure their overall accuracy in each scenario. WE set k to 5 in our experiments.

We also report and measure the accuracy of each method by also considering the predicted FDs that are subset/ supersets of the ground truth ones. We penalize matches of subset/ superset FDs. Let $c(f)$ and c_g denote the set of compliant tuples with FD f and the set of clean tuples in the ground truth labeled data, respectively. The *precision* and *recall* of an FD f are $\frac{|c(f) \cap c_g|}{|c(f)|}$ and $\frac{|c(f)|}{|c_g|}$, respectively. *F1 score* of FD f is the harmonic mean of its precision and recall. We discount matches of subset/ superset FDs using their *F1 score* differences with the user supplied one. In our results, we distinguish the cases where we consider subset/superset matches as well as exact matches by including a "+" in their names.

4.3 Results

4.3.1 Users' Learning Activity & Labeling Errors. To quantify the extent of learning activities and labeling errors, we report the degree by which the f1-score of the user's hypotheses changes between iterations to determine whether these changes are due to noise or significant changes to users' beliefs. Indeed, as users label tuples in each interaction according to their current belief, when users hold an inaccurate belief about the data, e.g., wrong FDs, they provide erroneous labels. We can see in Table 3 that in all but one scenario, the average change in f1-score is relatively large, especially when considering that a change of 0.33 is the difference between an FD that explains only $\frac{2}{3}$ of the violations in the dataset and an FD that explains almost all of the violations. This suggests that changes in the user's hypotheses from iteration to iteration are likely not due to simple noise but is rather due to the user updating their hypothesis of which FD best explains the violations in the data.

Scenario #	Average change in f_1-score
1	0.1144
2	0.3280
3	0.2301
4	0.2843
5	0.1767

Table 3. Average $\mathbf{f_1}$-score change between any two rounds of user labeling, grouped by scenario

4.3.2 Learning Models. Figure 2 shows the accuracy of different methods in predicting changes in the users' hypotheses over all interactions. Results show that in all but one scenario the Bayesian (FP) model significantly outperforms hypothesis testing in modeling participants' behavior. This trend holds true as the metric of success is loosened to accept subsets or supersets of the user's hypothesis. In most scenarios and interactions, the user's hypothesis is among the top-1 or 2 returned FDs by the Bayesian model. Given the size of the set of possible FDs, this shows that Bayesian (FP) model accurately models users' learning in most cases. This confirms the popularity of FP in empirical game theory and the general belief that as it is simple and does not require much (computational) resources, it accurately models learning behaviour in many settings. We observe a similar trend when grouping predictions based on participants: Bayesian (FP) model significantly outperform hypothesis testing for all our participants except for two.

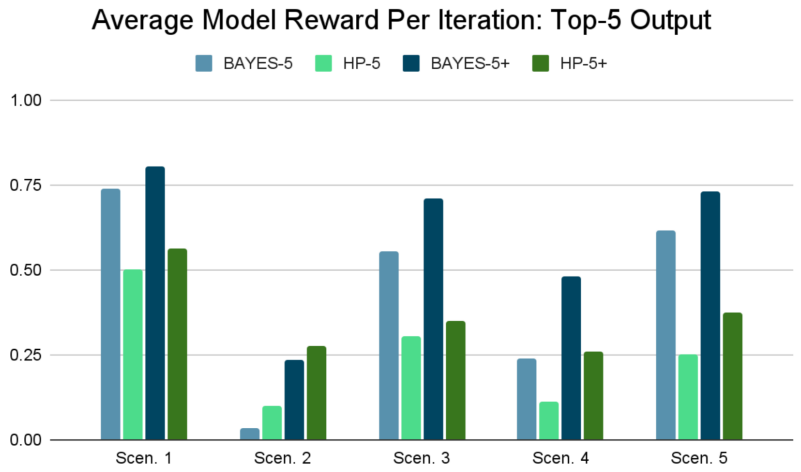


Fig. 2. MRR computed over all interactions for each learning model with $k = 5$

One important exception to the general trend is scenario 2 in which none of the available learning models are able to accurately predict participant's belief. While we have observed some degree of non-monotone learning behaviour in other scenarios, participants show a significantly less monotone learning in scenario 2 as they often moved from more accurate beliefs to less accurate ones. This may be due to the fact that this scenario is rather more difficult than others. Since current models for human learning expect some level of monotonicity in learning behaviour, they may not be able to predict human's learning in such settings. One may further improve these models by considering the probability of noise in decision making [54]. As Bayesian (FP) model accurately models human learning in our study in most cases, we will use it as the model for trainer's learning in our framework.

5 LEARNING METHODS FOR LEARNER

Our goal is to design settings and systems that help both agents to form accurate beliefs collaboratively and quickly. This leads the learner to learn the accurate target model. It is not clear whether one is able to modify the humans' learning schemes. Hence, we use the results of our user study in the preceding section to make realistic assumptions about their human trainers' learning schemes. Using these assumptions, we develop predictive models and response strategies for the learner that is aware of human learning and lead the learner to learn the accurate target model.

We are interested in the prediction and response models for the learner that guide the interaction to the accurate equilibrium. We first define precisely the notion of convergence to an equilibrium using concepts introduced in the theory of learning in games [19, 54]. Then, we introduce possible learner's predictive and response methods and investigate if they converge to a desired equilibrium.

5.1 Convergence to Equilibria

Given a pair of learning methods for the trainer and learner, we would like to know whether the interactions converges to some equilibrium. Let Φ^i_t denote the empirical frequencies, i.e., empirical distribution, of actions performed by agent i up to and including interaction t. Each member of Φ^i_t, denoted as $\Phi^i_t(x)$, is the observed number of occurrences of action x in interactions up to and including time t normalized by t. Let π^i is a policy of agent i. The following definition of convergence is from previous work on interactive games [54].

Definition 1. *Empirical behaviour of agent i converges to π^i as $t \to \infty$ if Φ^i_t converges to π^i almost surely ($P(\lim_{t\to\infty} \Phi^i_t = \pi^i) = 1$).*

Intuitively speaking, if we compute empirical distribution of the realized actions of an agent in all interactions, we find that it converges to a fixed policy in the long run.

The *empirical behaviour of a game* converges to an equilibrium if the empirical behaviours of both agents converge to policies $\pi^{\mathcal{T}}$ and $\pi^{\mathcal{L}}$ such that $(\pi^{\mathcal{T}}, \pi^{\mathcal{L}})$ is an equilibrium of the game. Let $\mathbf{L}^i$ denote the learning method of agent i. The *empirical behaviour of a learning scheme* $(\mathbf{L}^{\mathcal{T}}, \mathbf{L}^{\mathcal{L}})$ converges to an equilibrium if the empirical behaviour of every game in which the trainer and learner follow $\mathbf{L}^{\mathcal{T}}$ and $\mathbf{L}^{\mathcal{L}}$, respectively, converge to an equilibrium. Intuitively speaking, convergence in empirical behaviour expresses certain type of long-term and average-case stability.

Generally speaking, the learner uses its observed empirical frequencies of the trainer's labeling to update its belief. In particular, let us assume that the learner uses (Bayesian) FP to updated its belief. Thus, convergence in the empirical frequencies of the trainer's labeling leads to convergence of the learner observations and consequently its belief. Given that our learner selects its belief from a fixed hypothesis space, it picks the hypothesis that best describes the empirical distributions of its observations. Similarly, as the trainer also leverages its observations from the learner's behaviour to updates its belief using FP, convergence in empirical frequencies for learner actions will result in the convergence in belief of the trainer in properties of the data. Hence, in cases where the trainer and learner use FP as their predictive models, if the game converges in empirical frequencies to the desired equilibrium in which trainer's labeling and its belief about the data are accurate, then the beliefs of both agents will converge to the accurate and desired beliefs about the model and data. Since the main goal of agents is to reach a common and accurate belief about the target model, the convergence in empirical frequencies is sufficient.

In practice, to find out when to terminate the interactions, the learner may check the the difference in its beliefs in a given number of consecutive interactions. If the difference goes below a given sufficiently small threshold, it may terminate the interaction.

5.2 FP With Uncertainty Sampling

To understand the shortcoming of current active learning methods and their possible solutions, it is useful to investigate them to our setting more carefully. As mentioned in Section 5.1, current active learning methods, in particular the ones used to learn a model of FDs for data cleaning, use FP to update their belief and model [50]. As for the response strategy, research in the area of active learning indicate that some examples may be more helpful than others to learn a target model quickly [21, 34, 46]. For example, after some initial accurate labelings, it may be more useful for the learner to acquire labels on the examples that are close to the border of different classes [46]. The precise calculation of such measures usually requires perfect knowledge about the target model and classes, which is of course not available to the learner. Hence, active learning methods define proxy measures based on the current belief of the learner about the target model [21, 34, 46].

Active learning literature proposes some heuristics for picking examples for the learner. Using such examples, the learner may learn the correct belief using fewer labelings than random selection [21, 34]. We choose uncertainty sampling heuristic due to its popularity [46]. In uncertainty sampling, the learner picks the examples about which it is the most uncertain. In our setting, they are the tuples whose probability of being dirty according to the learner's belief are the closest to 0.5, with ties broken arbitrarily.

Thus, a typical active learning method updates its belief according to FP and picks the example(s) in each interaction that maximizes the proxy metric of uncertainty. We denote such a method as (FP, Uncertainty). This approach has a deterministic response strategy as it returns the examples

with the highest level of uncertainty. Thus, it may not provide the informative set of examples necessary for the trainer to improve its understanding and updates its belief about the data. In other words, in each interaction, its policies do not have the maximum payoff as their entropy is zero. It may present a biased view of the data to the trainer, therefore, the trainer may not gain a sufficiently accurate insight about the data. This may cause the trainer to provide biased labeling and guide the agents to learn a biased model for the target concept eventually. Also, as it assumes that the trainer's belief does not change, it does not revisit and pick examples from the regions for which it has shown some examples before. Hence, it may form inaccurate belief.

5.3 FP With Fixed Random Response

The learner may use a response strategy completely different from that one described in the preceding section and in each interaction select examples from the underlying dataset uniformly at random. This approach will select a fixed policy in every interaction independent of the learner's belief. In this policy, every example has equal probability of being selected and shown to the trainer in each interaction. We call this response strategy *fixed random* (*random* for short). Since this response method does not use the current learner belief, the learner (and trainer) may need too many samples to reach a common belief. Nonetheless, it is useful to investigate this approach to figure out whether it reaches a stable state as it provides background to more sophisticated techniques.

The random strategy presents representative examples from the data and maximizes the term representing the entropy in its payoff function. It, however, at least in the short-run, may not pick the policies that have the highest payoff in each interaction as it ignores the term in payoff related to the agreement between the labeling of the trainer and the belief of the learner. However, the hope is that after sufficiently many interactions, the trainer observes enough samples from the data and form an accurate belief about the data and the target model. Since the trainer predict the exact data distribution as $t \rightarrow \infty$ and the learner has a fixed response method, we have the following proposition. The *best response* (*best* for short) strategy picks the policy that has the highest payoff given the current belief of the agent.

Proposition 1. *Let the trainer and learner follow (FP, Best) and (FP, Random) learning schemes, respectively. The empirical behaviour of the game converges to an equilibrium.*

In the games of Proposition 1, as the trainer always picks the policy with highest payoff and given that after sufficiently many interactions ($t \rightarrow \infty$) it forms the accurate belief, the trainer will use the accurate policy. Thus, the learner eventually learns the accurate belief and both agents have accurate beliefs in the equilibrium.

Although the random response strategy has strong convergence properties to reach the desired equilibrium of the game, it may take a long time for the agents to get to this equilibrium. However, it shows that the idea of randomization may provide useful convergence properties to desired equilibria. Next, we leverage this idea and ideas from current research on active learning to design methods that converge more quickly than random.

5.4 Stochastic Best Response

Assume that the learner follows FP as its prediction model. Consider the response strategy that selects a policy in interaction t that picks example x according to the probability $\frac{e^{u_a(\theta_t^{\mathcal{L}}, x)/\gamma}}{\sum_{x' \in D} e^{u_a(\theta_t^{\mathcal{L}}, x')/\gamma}}$ where x and x' are examples, D is the dataset, and γ is the hyper-parameter defined in Section 3.2.3. This policy maximizes the payoff function of the learner in interaction t given its belief [19]. It establishes a balance between selecting examples with higher payoff of $u_a(.)$ and showing

an informative and diverse set of examples via stochastic choosing. Thus, it may present a more representative set of examples than the ones the deterministic policies explained in Section 5.2 show to the trainer, ultimately addressing their shortcoming. We call this response strategy *stochastic best response* (*stochastic best* for short). We denote the learning method that uses FP for prediction and stochastic best for response strategy as (FP, Stochastic Best).

Proposition 2. *Assume that the trainer and learner use learning methods (FP, Best) and (FP, Stochastic Best), respectively. The empirical behaviour of the game converges to an equilibrium.*

Proof. Since the training game is a game of identical interest, it is also a potential game [54]. As best response is a special case of stochastic best response, it follows from Theorem 6.1 in [27]. □

Our user study in the preceding section indicates that (FP, Best) is the dominant learning scheme of human trainers in our setting, Proposition 2 shows that if the learner uses (FP, Stochastic Best) against such trainer, its belief converges to the one of the trainer.

5.5 Stochastic Uncertainty Sampling

Stochastic best strategy is rather greedy and short-sighted as it aims at collecting immediate reward by presenting examples that it is confident about their labels. This may significantly prolong the number of interactions for learning the accurate belief. To address this issue, the learner may use active learning heuristics, such as uncertainty sampling, to pick examples while preserving the stochastic nature of these policies. That is, it may replace the term $u_a(\theta_t^{\mathcal{L}}, x)$ in the payoff function by the value of uncertainty of x given its belief. We call this response strategy *stochastic uncertainty sampling* (*stochastic uncertainty* for short). This strategy may not pick the policies with the highest reward in early interactions. Nonetheless, stochastic uncertainty retains the stochastic element and the stochastic best strategy. Therefore, it provides the trainer with the same level of information about the data as stochastic best strategy via presenting representative samples from the data. Also, the policies in this strategy are stochastic, each example has a non-zero probability of being selected and it avoids the significant bias in strategies described in Section 5.2. Additionally, research in active learning indicates that learners that use uncertainty sampling in stationary environments learn accurate target models [46]. Therefore, there are evidences that stochastic uncertainty may learn the accurate belief in the long run. In fact, our empirical analysis reported in Section 6 confirms this claim. This approach approximates uncertainty sampling when γ is close to zero.

The hyper-parameter γ reflects the familiarity of the user with the underlying data. The smaller the value of γ is, the more accurate the user prior belief is. As the trainer observes more data, it may gradually improve the accuracy of its belief. Thus, the learner may reduce the degree of randomization in stochastic uncertainty sampling by decreasing the value of γ over time. Using this technique, the trainer and learner may reach a common accurate belief using fewer interactions and examples.

6 EMPIRICAL STUDY

We evaluate and compare the number of interactions and samples each method in Section 5 required to learn a common belief with a learning trainer in the average-case.

6.1 Experimental Setting

Datasets & Models. We use four datasets: *OMDB* and *Airport* from our user study in Section 4, *Tax* and *Hospital* from the error detection literature [6, 13, 37]. *Hospital* is a real-world dataset with 19 attributes and six exact FDs, while *Tax* is a synthetic dataset with 15 attributes and four exact

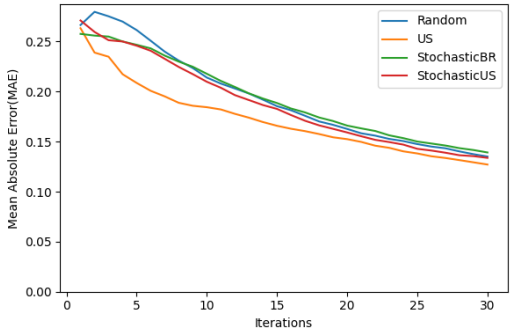


Fig. 3. Mean Absolute Error between Trainer and Learner models for *OMDB* dataset with ≈ 10% violations, trainer's prior model=Random, learner's prior model=Data-estimate

FDs. As opposed to learning one target model in Section 4, in this experiment the learner learns a model for 38 approximate FDs for each dataset, i.e., a distribution over the confidence of each FD. Each FD has at most four attributes. The goal for the learner is to reach an agreement with the model of the learning trainer as fast as possible, as shown in Figure 1.

State-of-the-art & Proposed Methods. We compare our proposed methods with the state-of-the-art active learning technique of *Uncertainty Sampling (US)*. As explained in Section 5.2, US is widely used in current active learning settings and assume the trainers have a fixed belief and do *not* learn during labeling. We also implement the baseline method of *Fixed Random Sampling (Random)* method that picks examples uniformly at random and present them to the trainer for labeling. We implement our proposed methods *Stochastic Best Response (StochasticBR)* (Section 5.4) and *Stochastic Uncertainty Sampling (StochasticUS)* (Section 5.4). For measure of uncertainty in both US and StochasticUS, we use entropy as $entropy(x, \theta_t) = -p_{\theta_t}(x)\log(p_{\theta_t}(x)) - (1-x)\log(1-p_{\theta_t}(x))$, where $p_{\theta_t}(x)$ is probability of x being a clean tuple based on belief θ_t. The value of γ for the *StochasticBR* and *StochasticUS* is set to be 0.5 in all experiments. This value reduces the greediness in sampling compared to the softmax function where $\gamma = 1$. As explained in Section 2, FD violations are defined and validated over pairs of tuples. Hence, we have modified all response and sampling methods to select a pair of tuples instead of a single one.

Trainer's Method. Based on our findings in Section 4, we simulate the trainer's learning using FP (Bayesian).

Learner's & Trainer's Priors. We test a set of prior beliefs (*Uniform-d, Random, and Data-estimate*) for the trainer to measure the sampling methods' sensitivity of convergence to the different types of prior beliefs. In *Uniform-d*, the confidence of all the FDs in the hypothesis space is initialized to the same value $d \in [0, 1]$. In *Random*, the confidence of each FD is sampled randomly from $[0, 1]$. For *Data-estimate*, the prior is set to the average confidence computed based on the initial unlabeled dataset. This prior represents the case where the learner computes its prior by treating the unlabeled dataset to be completely clean, which is often used in practice.

Degrees of FD Violation. We evaluate the aforementioned methods by injecting different degrees of violations ($< 35\%$) for FDs in the original data. For every dataset, we identify a subset of the tuples so that the fraction of tuple pairs that are violations of the FDs in this sampled dataset is equal to the desired degrees of violations. The aim is to study the influence of the amount of FD violations on the performance of different response and sampling methods.

Interactions. At each iteration, Bayesian models corresponding to the learner and trainer are initialized from the set of prior beliefs. Based on the sampling method, the trainer serves a sample data of size $k = 10$. Upon receiving a sample, the trainer updates its model and labels the data based

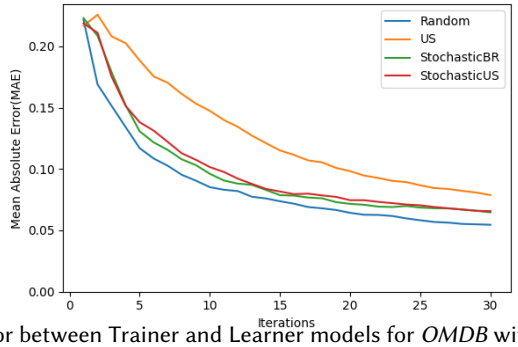


Fig. 4. Mean Absolute Error between Trainer and Learner models for *OMDB* with ≈ 10% violation, trainer's prior model=Random and learner's prior model=Uniform-0.9

on its updated model. Using the violation labels from the trainer, the learner updates its model accordingly. This interaction continues for $N = 30$ iterations.

Evaluation Metrics: We evaluate the aforementioned methods in terms of **their speed of convergence** and **effectiveness**. We measure the speed of convergence by computing how close the model of the learner is to the trainer's model at each iteration. For this, we use the *mean absolute error* (MAE) of confidence of the models over the hypothesis space. MAE measures the average difference between the confidence of the trainer model and the learner model over the FDs in the hypothesis space. This value can range from 0 to 1; lower value implying closeness between the models of these two agents. To evaluate the effectiveness of each method in error detection, we report *F1 score* for the learner's model in each interaction. We separate 30% of each dataset as the test set and compute F1 score using these test sets.

6.2 Experimental Results

6.2.1 Convergence. In this section, we analyze convergence of different methods.

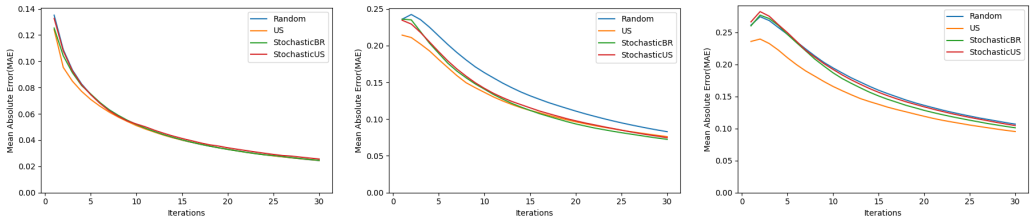
Sampling methods and Prior Models. *Fixed Random Sampling* method is the simplest sampling method as it completely ignores the gained information about the data and does not form a model. At the other extreme, *Uncertainty Sampling* fully relies on its current model and samples tuples greedily based on that.

We consider the case where Trainer's and Learner's prior models are different. Figures 3 and 5 show the results for the case with the trainer starting with a Random prior model and learner's with a prior based on Data-estimate. Results show that for all datasets, while *Random Sampling* reduces the mean error on average, its convergence is slower than with *Uncertainty Sampling*. Results for the stochastic sampling methods lie between these two methods because, while they exploit the model, they use stochastic sampling instead of acting greedily.

Figures 4 and 6 show that results change significantly when the learner's prior is not informed about the data at hand, i.e., when learner uses a uniform distribution. In this case, instead of helping, the wrong model can hurt the performance w.r.t. Random.

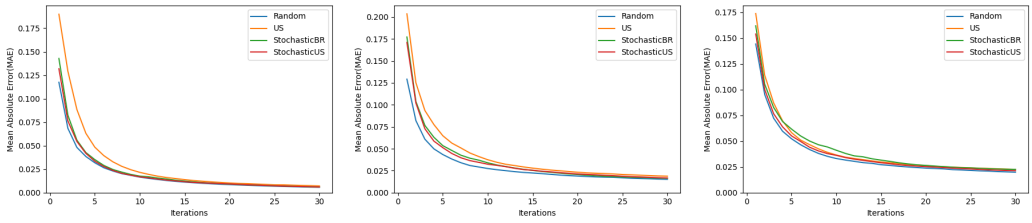
We conclude that, *Uncertainty Sampling* performs best when it is aware of the dataset at hand, which is the standard setting in practice. Otherwise, in the extreme case that the system does not have access to data, *Random Sampling* works best. Considering both scenarios, *Stochastic Best Response* and *Stochastic Uncertainty Sampling* work best on average compared to *Random Sampling* and *Uncertainty Sampling*.

Degree of Violation in Data. We evaluate how the performance of different methods change by increasing the degree of violation in the data. Figure 7 shows that if the model's priors are different, then the increment of violations exacerbates their performance. Our experiments for the cases



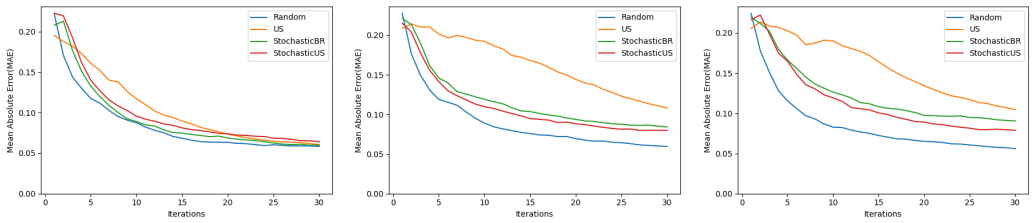
(a) Airport (b) Hospital (c) Tax

Fig. 5. MAE between Trainer and Learner Model, trainer's prior=Random, learner's prior=Data-estimate with ≈ 20% violation



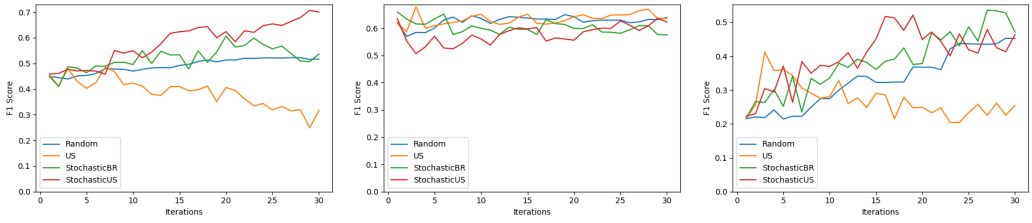
(a) Airport (b) Hospital (c) Tax

Fig. 6. MAE between Trainer and Learner Model, trainer's prior=Random, learner's prior=Uniform-0.9 with ≈ 20% violation



(a) Degree of violation ≈ 5% (b) Degree of violation ≈ 15% (c) Degree of violation ≈ 25%

Fig. 7. MAE between Trainer and Learner Model for *OMDB*, trainer's prior =Random, learner's prior=Uniform-0.9



(a) OMDB (b) Hospital (c) Tax

Fig. 8. Average F1 Score of the labeling of Learner model, trainer's prior=Random, learner's prior=Random with ≈ 20% violation

where the trainer and learner models are in agreement, not shown due to lack of space, indicate that

an increment in violations does not impact the results considerably. The results for other datasets demonstrate a similar trend and not shown due to the lack of space.

6.2.2 Accuracy. As explained in Section 6.1, we compare the error detection accuracy of our proposed methods, *Stochastic Best Response (StochasticBR)* and *Stochastic Uncertainty Sampling (StochasticUS)*, with the popular method of *Uncertainty Sampling (US)* in active learning and the baseline method of *Fixed Random Sampling (Random)*. Figures 8 shows the F1 scores of different methods over *OMDB*, *Hospital*, and *Tax* datasets for random priors of the learner and trainer. Overall, our proposed methods outperform or deliver the same level of F1 score as *Uncertainty Sampling* and *Fixed Random Sampling*. The relatively high values of F1 score for *Fixed Random Sampling* is mainly due to its high recall. Since it selects training examples uniformly at random, it provides the learner with an accurate overall understanding of the dataset. But, it generally delivers lower precision than other methods. This is because it does not provide the learner with sufficiently informative training examples in the reported interactions. *Uncertainty Sampling* delivers a lower recall and precision than other methods in most datasets. The drop in its recall is more significant than that of its precision compared to the proposed methods. This method assumes that trainer's belief is fixed, therefore, it picks samples from a subset of the data based on the early users' annotations. However, some of the early annotations might be erroneous. It cannot repair its early decisions on determining the subset of the data for selecting examples. Thus, its models are biased toward the early (inaccurate) decisions, which deliver low recall. But, the learner that uses this method still takes advantage of the correctly labeled examples in the later stages of interaction to improve the precision of its models. The results of using other priors for learner and trainer and degrees of violations show a similar trend and are not reported due to the lack of space.

7 RELATED WORK

Human-in-the-loop Data Analysis. There has been recent interest in developing human-centric data analysis and exploration methods [4, 14]. In particular, researchers have proposed data querying systems that take into account the characteristics of human users [38]. We, however, focus on considering human learning during active training that has a different setting and requires a different model and learning algorithms.

Learning From Imperfect Annotators. There is a stream of papers on active learning in challenging settings where some (weak) users can return incorrect labels [48, 55], are allowed to relabel examples [53], or even abstain from labeling. These approaches do not consider user learning during the annotation process. We believe combinations of these proposals and exploratory training should be explored in follow up work.

Discovering Functional Dependencies. In most practical settings, FDs are defined manually by domain experts after studying the data. Methods have been proposed to automatically discover FDs and integrity constraints (ICs) in general. The majority of such methods assume that a clean, but representative, sample of the data is provided and thus focus on efficiency [10, 12, 25]. Since gathering a large sample of clean data is hard, others drop the cleanliness assumption and target the discovery of approximate FDs [1, 29, 35, 42]. While these methods are efficient, they are not really useful in practice, as a large number of approximate FDs exist and discovery algorithms are sensitive to dirty data. In our application, we do not assume the presence of a sample of clean data and we do not expose the long list of FDs to the users. Some approaches claim that, by focusing on annotating examples only, this kind of systems can alleviate the human effort in discovering FDs [50].

Error Detection with Users. FDs and (ICs) have been used to identify violations and possible errors to be validated by humans [13, 36, 43]. However, methods have been proposed to let the users interact more closely with the cleaning systems. While most proposals focus on the repair step, i.e., show possible data repairs to the users to refine a ML module [52], a SQL update query [23], or a transformation [26], in this work we focus on systems to detect data errors [2]. Practical methods, where users are exposed to questions about cells, tuples and FDs, share the interactive detection approach with our proposal, but assume an oracle user, i.e., the answers from the users are always correct [24, 37, 50]. We remark that our framework for exploratory training is independent from the back-end of the error detection engine. While we adopt a simple system based on FDs to obtain clear beliefs from the users, more sophisticated solutions can be revised and extended based on our contributions.

Inter-active Visual Labelling. Researchers have extended active learning for visual data analytics by allowing users to select what data item to label or to directly manipulate the current learned model [28]. By offering additional methods of leveraging users' expertise, this method can reduce the number of interactions in active learning [7–9]. As opposed to our setting, these methods assume the user knowledge about the target model is fixed.

Uncertain and Weak Labeling. In many domains, there might not be sufficiently many correctly labeled training data. In these settings, some weak prior beliefs about the training data might be available, e.g., using negative examples or beliefs about subsets of training examples. Authors in [44] leverage such set of beliefs to learn accurate labels for training data using Bayesian reasoning. Similarly, [51] assumes that correct (hard) labels are available for a small subset of training examples and aims at using them to learn heuristics to label the rest of examples. In our setting, trainers might also start with weak prior beliefs, but their beliefs evolves over time. To reduce the number of interactions, an interesting future work is to use the ideas in the aforementioned line of work to update the model of the learner and its sampling strategy after sufficiently many (early) interactions. One could use the model in [44] to investigate the degree by which the converged belief generalizes to the data items not presented in the interaction.

Joint Data Cleaning and Modeling. Researchers have proposed methods of joint data cleaning and learning for models with convex loss functions, e.g., linear regression, by asking users to clean the tuples that are likely to impact the learning outcome [32]. As opposed to our setting, they assume that users do not make any mistake in cleaning and labeling. An interesting future work is to consider users' learning in both labeling and cleaning data items.

8 CONCLUSION & FUTURE WORK

We have carried out research on adaptation of interactive systems to human learning in the task of exploratory labelling of data. Given a new dataset, users can only rely on their prior knowledge when they start annotating it. Supported by a user study, we have modeled human learning, i.e., how their hypothesis over the data change with more labeling iterations. We exploit such model in novel algorithms that effectively select new data to expose to the users according to a formal collaborative learning model. Using empirical studies, we have shown that with our algorithms users improve their labelling process by converging to the correct hypothesis with a smaller number of interactions.

While this work focuses on an instance of exploratory training for error detection, we envision more applications of our framework, such as a more general labeling of tuples for classifiers. Moreover, user labels are influenced by other factors, in addition to users' belief about FD patterns. It would be valuable to extend the framework to consider all factors that might influence users'

labeling, such as typos and unwanted human mistakes. Finally, going beyond structured data, we plan to study our theoretical framework for any kind of data examples, such as images and text.

REFERENCES

[1] Ziawasch Abedjan, Cuneyt Gurcan Akcora, Mourad Ouzzani, Paolo Papotti, and Michael Stonebraker. 2015. Temporal Rules Discovery for Web Data Cleaning. *Proc. VLDB Endow.* 9, 4 (2015), 336–347. https://doi.org/10.14778/2856318.2856328

[2] Ziawasch Abedjan, Xu Chu, Dong Deng, Raul Castro Fernandez, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Michael Stonebraker, and Nan Tang. 2016. Detecting Data Errors: Where are we and what needs to be done? *Proc. VLDB Endow.* 9, 12 (2016), 993–1004.

[3] Serge Abiteboul, Richard Hull, and Victor Vianu. 1994. *Foundations of Databases: The Logical Level.* Addison-Wesley.

[4] Azza Abouzied, Dominik Moritz, and Michael J. Cafarella. 2022. HILDA'22: The SIGMOD 2022 Workshop on Human-in-the-Loop Data Analytics. In *Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 2552–2553. https://doi.org/10.1145/3514221.3524077

[5] Charu C. Aggarwal, Xiangnan Kong, Quanquan Gu, Jiawei Han, and Philip S. Yu. 2014. Active Learning: A Survey. In *Data Classification: Algorithms and Applications*, Charu C. Aggarwal (Ed.). CRC Press, 571–606. http://www.crcnetbase.com/doi/abs/10.1201/b17320-23

[6] Patricia C. Arocena, Boris Glavic, Giansalvatore Mecca, Renée J. Miller, Paolo Papotti, and Donatello Santoro. 2015. Messing up with BART: Error Generation for Evaluating Data-Cleaning Algorithms. *Proc. VLDB Endow.* 9, 2 (oct 2015), 36–47. https://doi.org/10.14778/2850578.2850579

[7] Jürgen Bernard, Marco Hutter, Matthias Zeppelzauer, Dieter Fellner, and Michael Sedlmair. 2018. Comparing Visual-Interactive Labeling with Active Learning: An Experimental Study. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2018), 298–308. https://doi.org/10.1109/TVCG.2017.2744818

[8] Jürgen Bernard, Matthias Zeppelzauer, Markus Lehmann, Martin Müller, and Michael Sedlmair. 2018. Towards User-Centered Active Learning Algorithms. *Computer Graphics Forum* 37, 3 (2018), 121–132. https://doi.org/10.1111/cgf.13406 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13406

[9] Jürgen Bernard, Matthias Zeppelzauer, Michael Sedlmair, and Wolfgang Aigner. 2018. VIAL: A Unified Process for Visual Interactive Labeling. *Vis. Comput.* 34, 9 (sep 2018), 1189–1207. https://doi.org/10.1007/s00371-018-1500-3

[10] Laure Berti-Équille, Hazar Harmouch, Felix Naumann, Noël Novelli, and Saravanan Thirumuruganathan. 2018. Discovery of Genuine Functional Dependencies from Relational Data with Missing Values. *Proc. VLDB Endow.* 11, 8 (2018), 880–892.

[11] Colin F. Camerer, Teck-Hua Ho, and Juin Kuan Chong. 2004. Behavioural Game Theory: Thinking, Learning and Teaching. In *Advances in understanding strategic behaviour : game theory, experiments, and bounded rationality*. Palgrave Macmillan, 120–180.

[12] Loredana Caruccio, Vincenzo Deufemia, Felix Naumann, and Giuseppe Polese. 2021. Discovering Relaxed Functional Dependencies Based on Multi-Attribute Dominance. *IEEE Trans. Knowl. Data Eng.* 33, 9 (2021), 3212–3228.

[13] Xu Chu, Ihab F. Ilyas, and Paolo Papotti. 2013. Holistic data cleaning: Putting violations into context. In *ICDE*. IEEE Computer Society, 458–469.

[14] Trevor Darrell, Xin Wang, Li Erran Li, Fisher Yu, Zeynep Akata, Wenwu Zhu, Pradeep Ravikumar, Shiji Zhou, Shanghang Zhang, and Kalesha Bullard. 2021. HILL'21: ICML Workshop on Human in the Loop Learning. In *Proceedings of the 2021 International Conference on Machine Learning (ICML '21)*.

[15] Kyriaki Dimitriadou, Olga Papaemmanouil, and Yanlei Diao. 2014. Explore-by-example: an automatic query steering framework for interactive data exploration. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, Curtis E. Dyreson, Feifei Li, and M. Tamer Özsu (Eds.). ACM, 517–528. https://doi.org/10.1145/2588555.2610523

[16] Wenfei Fan. 2008. Dependencies revisited for improving data quality. In *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2008, June 9-11, 2008, Vancouver, BC, Canada*, Maurizio Lenzerini and Domenico Lembo (Eds.). ACM, 159–170. https://doi.org/10.1145/1376916.1376940

[17] Wenfei Fan and Floris Geerts. 2012. *Foundations of Data Quality Management.* Morgan & Claypool Publishers. https://doi.org/10.2200/S00439ED1V01Y201207DTM030

[18] Daniel Fink. 1997. *A Compendium of Conjugate Priors.* Technical Report. https://www.johndcook.com/CompendiumOfConjugatePriors.pdf

[19] Drew Fudenberg and David Levine. 1998. *The Theory of Learning in Games.* MIT Press.

[20] Samuel J. Gershman, Eric J. Horvitz, and Joshua B. Tenenbaum. 2015. Computational rationality: A converging paradigm for intelligence in brains, minds, and machines. *Science* 349, 6245 (2015), 273–278. https://doi.org/10.1126/

science.aac6076 arXiv:https://science.sciencemag.org/content/349/6245/273.full.pdf
[21] Daniel Golovin, Andreas Krause, and Debajyoti Ray. 2010. Near-Optimal Bayesian Active Learning with Noisy Observations. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1* (Vancouver, British Columbia, Canada) *(NIPS'10)*. Curran Associates Inc., Red Hook, NY, USA, 766–774.
[22] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning*. Springer.
[23] Jian He, Enzo Veltri, Donatello Santoro, Guoliang Li, Giansalvatore Mecca, Paolo Papotti, and Nan Tang. 2016. Interactive and Deterministic Data Cleaning. In *SIGMOD*. ACM, 893–907.
[24] Alireza Heidari, Joshua McGrath, Ihab F. Ilyas, and Theodoros Rekatsinas. 2019. HoloDetect: Few-Shot Learning for Error Detection. In *SIGMOD*. ACM, 829–846.
[25] Arvid Heise, Jorge-Arnulfo Quiané-Ruiz, Ziawasch Abedjan, Anja Jentzsch, and Felix Naumann. 2013. Scalable Discovery of Unique Column Combinations. *Proc. VLDB Endow.* 7, 4 (2013), 301–312.
[26] Joseph M. Hellerstein, Jeffrey Heer, and Sean Kandel. 2018. Self-Service Data Preparation: Research to Practice. *IEEE Data Eng. Bull.* 41, 2 (2018), 23–34.
[27] Josef Hofbauer and William H. Sandholm. 2002. On the Global Convergence of Stochastic Fictitious Play. *Econometrica* 70, 6 (2002), 2265–2294. http://www.jstor.org/stable/3081987
[28] Benjamin Höferlin, Rudolf Netzel, Markus Höferlin, Daniel Weiskopf, and Gunther Heidemann. 2012. Inter-active learning of ad-hoc classifiers for video visual analytics. *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)* (2012), 23–32.
[29] Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. 1999. TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies. *Comput. J.* 42, 2 (1999), 100–111.
[30] Ihab F. Ilyas, Volker Markl, Peter Haas, Paul Brown, and Ashraf Aboulnaga. 2004. CORDS: Automatic Discovery of Correlations and Soft Functional Dependencies. In *SIGMOD*.
[31] Jyrki Kivinen and Heikki Mannila. 1992. Approximate Dependency Inference from Relations. In *ICDT (Lecture Notes in Computer Science, Vol. 646)*, Joachim Biskup and Richard Hull (Eds.). Springer, 86–98.
[32] Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J. Franklin, and Ken Goldberg. 2016. ActiveClean: Interactive Data Cleaning for Statistical Modeling. *Proc. VLDB Endow.* 9, 12 (aug 2016), 948–959. https://doi.org/10.14778/2994509.2994514
[33] Rui Li, Rui Guo, Zhenquan Xu, and Wei Feng. 2012. A prefetching model based on access popularity for geospatial data in a cluster-based caching system. *International Journal of Geographical Information Science* 26, 10 (2012), 1831–1844.
[34] Christopher H. Lin, Mausam, and Daniel S. Weld. 2016. Re-Active Learning: Active Learning with Relabeling. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (Phoenix, Arizona) *(AAAI'16)*. AAAI Press, 1845–1852.
[35] Ester Livshits, Alireza Heidari, Ihab F. Ilyas, and Benny Kimelfeld. 2020. Approximate Denial Constraints. *Proc. VLDB Endow.* 13, 10 (2020), 1682–1695.
[36] Ester Livshits, Benny Kimelfeld, and Sudeepa Roy. 2020. Computing Optimal Repairs for Functional Dependencies. *ACM Trans. Database Syst.* 45, 1, Article 4 (2020), 46 pages. https://doi.org/10.1145/3360904
[37] Mohammad Mahdavi, Ziawasch Abedjan, Raul Castro Fernandez, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2019. Raha: A Configuration-Free Error Detection System. In *SIGMOD*. ACM, 865–882.
[38] Ben McCamish, Vahid Ghadakchi, Arash Termehchy, Behrouz Touri, and Liang Huang. 2018. The Data Interaction Game. In *Proceedings of the 2018 International Conference on Management of Data* (Houston, TX, USA) *(SIGMOD '18)*. ACM, New York, NY, USA, 83–98. https://doi.org/10.1145/3183713.3196899
[39] Tom Mitchell. 1997. *Machine Learning*. McGraw-Hil.
[40] Yael Niv. 2009. The Neuroscience of Reinforcement Learning. In *ICML*.
[41] Y Niv. 2009. Reinforcement learning in the brain. *The Journal of Mathematical Psychology* 53, 3 (2009), 139–154.
[42] Eduardo H. M. Pena, Eduardo C. de Almeida, and Felix Naumann. 2019. Discovery of Approximate (and Exact) Denial Constraints. *Proc. VLDB Endow.* 13, 3 (2019), 266–278.
[43] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. HoloClean: Holistic Data Repairs with Probabilistic Inference. *Proc. VLDB Endow.* 10, 11 (2017), 1190–1201.
[44] Esther Rolf, Nikolay Malkin, Alexandros Graikos, Ana Jojic, Caleb Robinson, and Nebojsa Jojic. 2022. Resolving label uncertainty with implicit posterior models. In *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence (Proceedings of Machine Learning Research, Vol. 180)*, James Cussens and Kun Zhang (Eds.). PMLR, 1707–1717. https://proceedings.mlr.press/v180/rolf22a.html
[45] Alvin E Roth and Ido Erev. 1995. Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term. *Games and economic behavior* 8, 1 (1995), 164–212.
[46] Burr Settles. 2009. *Active Learning Literature Survey*. Computer Sciences Technical Report 1648. University of Wisconsin–Madison. http://axon.cs.byu.edu/~martinez/classes/778/Papers/settles.activelearning.pdf
[47] Burr Settles. 2012. *Active Learning*. Morgan & Claypool Publishers.

[48] Pannaga Shivaswamy and Thorsten Joachims. 2015. Coactive Learning. *J. Artif. Int. Res.* 53, 1 (may 2015), 1–40.
[49] Joshua B. Tenenbaum. 1999. Bayesian Modeling of Human Concept Learning. In *Advances in Neural Information Processing Systems 11*, M. J. Kearns, S. A. Solla, and D. A. Cohn (Eds.). MIT Press, 59–68. http://papers.nips.cc/paper/1542-bayesian-modeling-of-human-concept-learning.pdf
[50] Saravanan Thirumuruganathan, Laure Berti-Équille, Mourad Ouzzani, Jorge-Arnulfo Quiané-Ruiz, and Nan Tang. 2017. UGuide: User-Guided Discovery of FD-Detectable Errors. In *, SIGMOD*. ACM, 1385–1397. https://doi.org/10.1145/3035918.3064024
[51] Paroma Varma and Christopher Ré. 2018. Snuba: Automating Weak Supervision to Label Training Data. *Proc. VLDB Endow.* 12, 3 (nov 2018), 223–236. https://doi.org/10.14778/3291264.3291268
[52] Mohamed Yakout, Ahmed K. Elmagarmid, Jennifer Neville, Mourad Ouzzani, and Ihab F. Ilyas. 2011. Guided Data Repair. *Proc. VLDB Endow.* 4, 5 (Feb. 2011), 279?289. https://doi.org/10.14778/1952376.1952378
[53] Songbai Yan, Kamalika Chaudhuri, and Tara Javidi. 2016. Active Learning from Imperfect Labelers. In *NIPS*, Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (Eds.). 2128–2136.
[54] H Peyton Young. 2004. *Strategic learning and its limits.* OUP Oxford.
[55] Chicheng Zhang and Kamalika Chaudhuri. 2015. Active Learning from Weak and Strong Labelers. In *NIPS*, Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (Eds.). 703–711.
[56] Ugur Çetintemel, Mitch Cherniack, Justin DeBrabant, Yanlei Diao, Kyriaki Dimitriadou, Alexander Kalinin, Olga Papaemmanouil, and Stanley B. Zdonik. 2013. Query Steering for Interactive Data Exploration. In *CIDR*.

Received October 2022; revised January 2023; accepted February 2023