Reinforcement Learning for Data Preparation with Active Reward Learning

Laure Berti-Equille[1,2][0000−0002−8046−0570]

[1] ESPACE-DEV/IRD, UMR 228, IRD/UM/UG/UR,
Montpellier, France
[2] Aix Marseille Université, Université de Toulon, CNRS, LIS,
DIAMS, Marseille, France
laure.berti@ird.fr

Abstract. Data cleaning and data preparation have been long-standing challenges in data science to avoid incorrect results, biases, and misleading conclusions obtained from "dirty" data. For a given dataset and data analytics task, a plethora of data preprocessing techniques and alternative data cleaning strategies are available, but they may lead to dramatically different outputs with unequal result quality performances. For adequate data preparation, the users generally do not know how to start with or which methods to use. Most current work can be classified into two categories: 1) they propose new data cleaning algorithms specific to certain types of data anomalies usually considered in isolation and without a "pipeline vision" of the entire data preprocessing strategy; 2) they develop automated machine learning approaches (AutoML) that can optimize the hyper-parameters of a considered ML model with a list of by-default preprocessing methods. We argue that more efforts should be devoted to proposing a principled and adaptive data preparation approach to help and learn from the user for selecting the optimal sequence of data preparation tasks to obtain the best quality performance of the final result. In this paper, we extend Learn2Clean, a method based on Q-Learning, a model-free reinforcement learning technique that selects, for a given dataset, a given ML model, and a preselected quality performance metric, the optimal sequence of tasks for preprocessing the data such that the quality metric is maximized. We will discuss some new results of Learn2Clean for semi-automating data preparation with "the human in the loop" using active reward learning and Q-learning.

Keywords: Reinforcement learning · Data preparation · Q-learning · Data preprocessing.

1 Motivations

In a 2017 survey conducted by the data science community Kaggle, "dirty data" is the common answer for 49.4% members responding to the questionnaire, when asked about the biggest barriers faced in data science[3]. As mitigation, data preparation often accounts for about 80% of the work of data scientists, but it is perceived as the least

[3] https://www.kaggle.com/surveys/2017

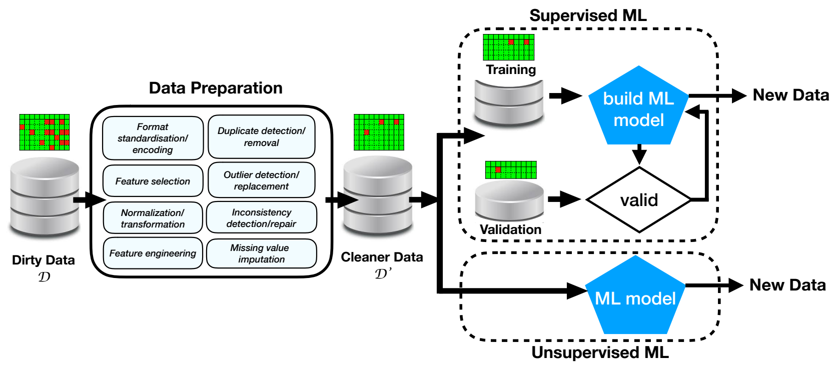

Fig. 1. A Generic ML workflow

enjoyable part of the data scientists job. As scientists, they would rather be deriving new knowledge and insights. The paradox is that without principled and adequate data preparation, those new insights are suspect at best.

Data preparation and data cleaning are known to be very challenging tasks that require detection and elimination of a variety of complex data quality problems, such as duplicate, inconsistent, missing, and outlying values. A wide range of methods for statistical analysis [13], constraint mining and checking [2], entity matching [9, 5], and machine learning [16, 11, 7] are used nowadays for data quality checks, data cleaning, and data repairing [4]. However, most existing systems suffer from significant limitations.

First, data preprocessing includes many steps (e.g., normalization, transformation, encoding, discretization, imputation, deduplication, pattern or rule enforcement, feature selection, and engineering, etc.) but current systems generally provide support for a limited number of steps in the pipeline represented in Fig. 1.

Second, for a given data preprocessing step, current systems generally rely on a few by-default methods. The users have to try and test the methods to select the most appropriate ones in a tedious and time-consuming process. To extend the system with another method eventually more adequate, the users will have to write code or include other libraries, e.g., imputing missing values can be based on median, mean, most frequent values as default methods, but multiple imputations by chained equations (MICE) or K-nearest neighbors may be more accurate, but they are generally not included in the set of by-default methods provided by the systems.

Third, the systems do not recommend neither the most adequate preprocessing methods for a given dataset and a given ML task, nor the complete strategy as the full execution sequence of the methods: e.g., imputation of missing values can be done after or before deduplication, and these two different orderings may lead to a dramatically different preprocessed dataset. While some AutoML systems [3] can find the best model and configuration of hyper-parameters, the optimization has not been yet applied to the entire data preprocessing pipeline which remains fixed and based on by-default methods in a fixed (potentially nonoptimal) ordering.

Finally, the users may not know which preprocessing methods can be applied to optimize the final results downstream. This would require executing all possible methods for each task of preprocessing, as well as all the possible combinations of the methods with different orderings.

We argue that more efforts could be devoted to proposing a principled and adaptive data preparation solution to help the users in selecting the optimal sequence of data preprocessing tasks to obtain the best quality performance of the final result. As the first step in this direction, we have proposed Learn2Clean[4] in [1], a method based on Q-Learning, a model-free reinforcement learning technique that selects, for a given dataset, a given ML model, and a quality performance metric, the optimal sequence of tasks for preprocessing the data such that the quality metric of the final result is maximized. To integrate the "human in the loop", leverage existing knowledge and user's feedback during execution, we present a novel contribution enhancing Learn2Clean exploration strategy with active reward learning.

The paper is organized as follows. Section 2 presents a literature review and positions our contribution with respect to the literature. Section 3 presents Learn2Clean computational model based on Q-learning for data preprocessing and gives our notations. Section 4 discusses how the model is enhanced with active reward learning to integrate the "human-in-the-loop" and leverage the user's feedback. Finally, Section 5 provides some experimental results and insights in favor of our approach of adapting Q-learning with active reward learning for data preparation.

2 Related Work

Machine Learning-based Data Cleaning. Several recent works that use machine learning to improve the efficiency and reliability of data cleaning and data repairing have been proposed lately [16, 11, 6, 12, 7]. In [16], Yakout et al. train ML models and evaluate the likelihood of recommended replacement value to fix erroneous or missing values. In [11], Rekatsinas et al. propose a framework for data repairing based on probabilistic inference to handle integrity constraints or external data sources seamlessly, with quantitative and statistical data repairing methods. Human input is often expensive and impractical to apply to entire large datasets. Machine learning can advantageously infer rules from a small set of examples cleaned by a human as shown in entity matching and deduplication by [6]. This approach can be coupled with active learning [12] to learn an accurate deduplication model with the fewest possible number of examples. Krishnan et al. in [7] propose a method for data cleaning optimization for user-defined ML-based analytics. Their approach selects an ensemble of methods (statistical and logic rules) for error detection and repair combinations using statistical boosting. However, to the best of our knowledge, our approach is the first one that leverages reinforcement learning for data cleaning and data preparation.

Reinforcement learning. Reinforcement learning (RL) [14] has the potential to learn from interaction with the environment adaptively. It consists in discovering which actions produce the greatest reward by experience and estimating how good it is to take

[4] `https://github.com/LaureBerti/Learn2Clean`

certain action in a given state, with the overall goal to maximize cumulated reward. Q-learning [15] is widely used because of its computational simplicity. In Q-learning, one does not require a model of transition functions and reward functions but learns directly from observed experience. This is an advantage as learning a model often requires exhaustive exploration that is not suitable for a large state space. Thus, Q-learning seems more suitable for exploring data preparation space. Furthermore, data preparation is a somewhat new application of reinforcement learning, and it raises new exciting challenges related to the dynamic update of the reward matrix at runtime. Typically, when the algorithm starts with a data preprocessing strategy, it picks and executes an action that will update the original dataset, and the set of the next valid actions and the corresponding reward matrix also change dynamically. Integrating the human in the loop in this context via active reward learning for automating data curation is again a novel contribution.

3 Learn2Clean Computational Model

Reinforcement learning methods can continuously learn from interaction with a dynamic environment [14]. Initially, with no prior knowledge of which action has to be taken in a given state, a conventional RL method takes an action in a starting state, receives a reward, moves to some next state, and repeats this procedure. It discovers which actions produce the greatest reward by trial-and-error experiences and estimates how good it is to take certain action in a given state to maximize the cumulated reward. Reinforcement learning methods are usually classified as :

– Model-based or model-free methods: model-based methods iteratively collect data samples randomly to learn the dynamics model, plan, recollect training data, and back-propagate the changes into a policy; model-free methods improve the value function directly from observed experience and do not rely on any predefined transition and reward models;
– On- or off-policy approaches: RL methods have to learn the optimal policy while behaving non-optimally, i.e., by exploring all actions. On-policy methods learn the best policy while using it to make decisions, whereas off-policy methods learn a policy different from what currently generates their behavior and actions.

3.1 Q-Learning for data preprocessing

Learn2Clean is based on Q-learning [15] which is a model-free, off-policy reinforcement method that learns the transition probability $T(s_{t+1}|(s_t, a))$ from the pair of current state s_t and action a to the next state s_{t+1}. It is a temporal-difference (TD) method that does not require prior knowledge about the environment's dynamics.

In our context of data preparation, the transition probabilities and the dynamics of the system are not given a priori because we do not know precisely which data preprocessing strategy would be the most adequate for a given dirty dataset and ML-based analytics tasks at hand. Furthermore, the space of possible states and actions can be very large if we consider all possible preprocessing methods with the range of their

Table 1. Notations

Notation	**Description**
$M = \langle S, A, T, R, \gamma \rangle$	Markov Decision Process (MDP) with states S, actions A, transition function $T : S \times A \rightarrow Pr[S]$, reward function R, and discount factor γ
γ	The discount factor $\gamma \rightarrow [0, 1)$
π	A policy $\pi : S \rightarrow Pr[A]$ defined in Eq.(1) as the probability of selecting an action in a state
$Q^\pi(s, a)$	The Q-function represents the expected long-term reward of taking action a in state s, and following policy π for a given MDP as defined in Eq. (2) (also noted $Q(s, a)$ for simplification)
α	The learning rate $\alpha \rightarrow (0, 1]$ used in Eq.(2)
$\mathcal{S}$	Sequence of pairs (state, action), $\mathcal{S} = \langle (s_1, a_1), (s_2, a_2), \ldots, (s_p, a_p) \rangle$
$q(\mathcal{S})$	Quality performance metric for sequence of actions $\mathcal{S}$ for a given ML goal, e.g., accuracy for classification, silhouette for clustering, MSE for regression
$R^d(s, a)$	The deterministic reward function per (state, action) pair $R : S \times A \rightarrow [R_{min}, R_{max}]$
$R^u(\mathcal{S})$	The user-defined reward per sequence $R^u : (S \times A) \times \cdots \times (S \times A) \rightarrow [-\delta, +\delta]$
$R(\mathcal{S})$	The global reward defined in Eq.(3) also noted R for simplification
$\mathcal{D}$	The training dataset $\mathcal{D} = \{(\mathbf{q}_{1:n}, \mathbf{s}_{1:n}, \mathbf{R}_{1:n})\}$ where the quality metric value q_i for a sequence of actions s_i leads to a certain reward value R_i
$Pr(R\|\mathbf{g}, \mathcal{D})$	The probabilistic reward model given the goal $\mathbf{g}$ as the quality metric for a given ML model and the training data $\mathcal{D}$ as a sequence of data preparation tasks

possible parameter configurations. Moreover, since we may not have generic solutions from previous datasets and tasks to learn how to clean, we believe that model-based reinforcement learning methods are not adapted.

Learn2Clean learns through experience in an unsupervised way. It explores from state to state until it reaches the goal (i.e., to maximize the user-defined quality metric).

Each exploration is equivalent to one training session in which the system explores the data preparation graph possibilities, receives the reward (if any) until it reaches the goal state. The purpose of the training is to enhance the decision of the system to get more training results. In other words, it has the choice between exploiting its knowledge by choosing the action with the highest estimated reward value and exploring its environment by taking any other action. Learn2Clean determines the next action (as the next preprocessing method to execute) without waiting until the end of the episode (i.e., the entire cleaning pipeline). This makes the method agile, computationally simple, and very performant compared to other data preparation approaches.

In [1], we propose a Q-learning reinforcement learning algorithm that can learn optimal actions for data preprocessing using heuristics for exploration of the action space. For exploration in RL, we apply the policy of Softmax action selection: the actions are weighted according to their respective Q-value estimates and sampled from the resulting distribution in the form of the Boltzmann distribution such as:

$$\pi = P(a \mid s) = \frac{e^{Q(s,a)/k}}{\sum_j e^{Q(s,a_j)/k}} \tag{1}$$

where k is a positive parameter such that a higher k value results in a more uniform distribution while a lower k value results in greedy action selection. Our notations are given in Table 1. The estimated value of taking action a in state s with policy π, denoted $Q^{\pi}(s,a)$ (or $Q(s,a)$ for simplification) is updated using reward $R^d(s,a)$, and the observed next state s' as:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(R^d(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)\big) \tag{2}$$

with α is a positive fraction such that $0 < \alpha \leq 1$, the step-size parameter that influences the rate of learning. When $\alpha = 1$, the system considers only the most recent information for learning. If α is properly reduced over time, the function converges [14]. The discount rate γ ($0 \leq \gamma < 1$) determines the present value of future rewards. If $\gamma = 0$, the method is only concerned with the immediate reward. The action influences only the current reward. If γ approaches 1, the method considers future rewards more strongly. To maximize total reward, the system must select the action with the highest value (exploitation), but to discover such action, it has to try actions not selected before (exploration).

4 Exploration with Human-In-the-Loop via Active Reward Learning

To enable the exploration phase to experience other actions not taken before and increase the greater total reward in the long run in discovering better actions and also actions suggested by the human expert, we propose a new probabilistic reward model $Pr(R|\mathbf{g},\mathcal{D})$, instead of a deterministic reward function $R^d(s,a)$ in order to modify Eq.(2). This allows us to integrate the human analyst in the loop and leverage the information about the certainty of the estimate to control the amount of human interaction required, as we only need to interact with the human expert if our model of the reward is not certain enough. The new architecture is presented in Figure 2.

Reward Design. Both intermediate rewards and final rewards are used to guide the behavior of the system. Given a sequence $\mathcal{S}$, we define the final rewards as a sum over sequence-specific rewards and the user-defined rewards collected on-line for $\mathcal{S}$.

– The sequence-specific deterministic rewards R^d consist of the sum of deterministic scores for the sequence where invalid consecutive actions are penalized, e.g., normalization before imputation is penalized by -1, whereas normalization after imputation is valid with a score augmentation of +1. Valid sequences of actions are defined initially (as valid moves) in the initial reward matrix.
– The intermediate rewards R^u include step-wise validity rewards learned from the user. A small positive reward is assigned if the sequence of tasks does not violate the user's suggestions. Otherwise, a small negative reward is assigned.

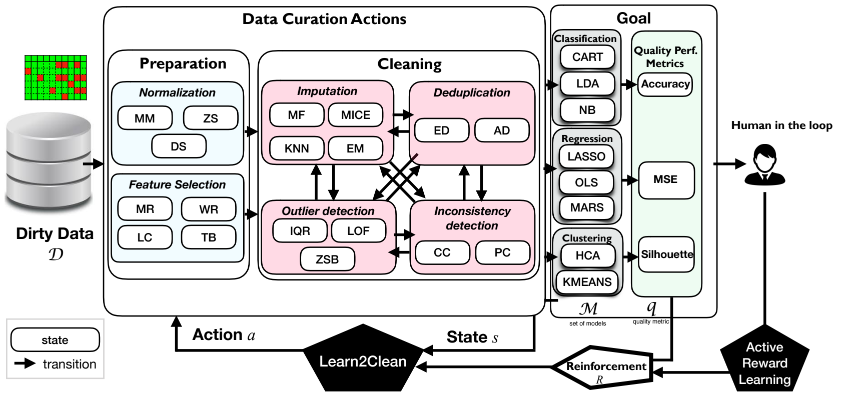

Fig. 2. Learn2Clean Extension with Active Reward Learning

The final reward score for a given sequence $\mathcal{S}$ is

$$R(\mathcal{S}) = \sum_{\forall(s,a)\in\mathcal{S}} R^d(s,a) + R^u(\mathcal{S}) \tag{3}$$

Active Reward Learning. Our goal is to find a model $Pr(R|\mathbf{q},\mathcal{D})$ that predicts the global reward given an observed quality metric outcome **q** and training data $\mathcal{D}$, which is partly obtained from the human expert annotating a set of sequences of actions to reach a certain ML goal in terms of quality performance. When modeling the reward, we have to take into account that the expert can give noisy samples of his implicit reward function and we also have to model the observation noise. Thus, we need to solve the regression problem:

$$R(\mathbf{q}) = f(\mathbf{q}) + \eta, \eta \sim \mathcal{N}(0,\beta), \tag{4}$$

where we assume zero mean Gaussian noise. Such a regression problem can, for example, be solved with Gaussian Process (GP) regression: $f(\mathbf{q}) \sim GP(m(\mathbf{q}), Cov(\mathbf{q},\mathbf{q}'))$, where $m(\mathbf{q})$ is the mean function and $Cov(\mathbf{q},\mathbf{q}')$ is the covariance function of the GP. Similarly as in [8], we use the standard squared exponential covariance function

$$Cov(\mathbf{q},\mathbf{q}') = \theta_0^2 \exp\Big(-\frac{||\mathbf{q}-\mathbf{q}'||^2}{2\theta_1^2}\Big). \tag{5}$$

The set of hyper parameters $\boldsymbol{\theta} = \{\theta_0, \theta_1, \beta\}$ is found through Bayesian optimization [10]. We collect observations from the human analyst in previous iterations and add them to the training set $\mathcal{D} = \{(\mathbf{q}_{1:n}, \mathbf{s}_{1:n}, \mathbf{R}_{1:n})\}$ where the quality metric value q_i for a sequence of actions s_i leads to a certain reward value R_i[5]. We use Bayesian optimization to decide what reward R^{new} should be considered next. By the properties of GPs, $\mathbf{R}_{1:n}$ and R^{new} are jointly Gaussian and using the Sherman-Morrison-Woodbury

[5] We use subscripts to denote the sequences of actions annotated by the human analyst, i.e., $\mathbf{q}_{1:n} = \{q(\mathcal{S}_1),\ldots,q(\mathcal{S}_n)\}$) with related reward $\mathbf{R}_{1:n} = \{R(\mathcal{S}_1),\ldots,R(\mathcal{S}_n)\}$).

formula (see [10] for details), the predictive posterior reward $Pr(R^{new}|\mathbf{q},\mathcal{D})$ of a new goal $\mathbf{q}^{new}$ is then given by a Gaussian with mean $\mu(\mathbf{q}^{new})$ and variance $\sigma^2(\mathbf{q}^{new})$ as

$$Pr(R^{new}|\mathbf{q},\mathcal{D}) \sim \mathcal{N}\Big(\mu(\mathbf{q}^{new}), \sigma^2(\mathbf{q}^{new})\Big). \quad (6)$$

To optimize the model of the reward function, we adapt the Expected Improvement (EI) from [8] that balances the greedy optimization with an exploration-exploitation trade-off parameter ξ such as:

$$EI(\mathbf{q}) = \Phi\left(\frac{\mu(\mathbf{q}) - \mu^* - \xi}{\sigma^2(\mathbf{q})}\right) \text{ with } \mu^* = \max_{i=1..B} \mu(q_i), \quad (7)$$

the best sample in the training set $\mathcal{D}$ over the budget B of queries submitted to the user and $\Phi(.)$ is the normal cumulative density function. The exploration-exploitation trade-off parameter ξ is chosen manually as it determines the amount of exploration during optimization and higher values lead to more exploration. A suggested value by [8] is $\xi = 0.01$.

Algorithm 1 Bayesian Optimization for Active Reward Leaning in Learn2Clean

```
1. INPUT:
2.        $B$ budget of queries to the human expert
3.        Top-$k$ list of pairs (quality metric, sequence of actions) ordered by quality metrics
              $\mathcal{L} = [(q_1, s_1), \ldots, (q_k, s_k)]$
4. FOR $i = 1$ to $B$ (with $B \leq k$)
5.    REPEAT
6.        Find the next sequence $s_i$ in $\mathcal{L}$ by optimizing the expected improvement
            of quality $EI$ over the GP: $s_i = \arg\max_s\Big(EI(q_s \mid \mathcal{D}_{1:i-1})\Big)$
7.        Obtain a possibly noisy reward sample $R(q_i) = f(q_i) + \eta$ from the objective
            function $f$
8.        Add the sample to previous samples $\mathcal{D}_{1:i} = \{\mathcal{D}_{1:i-1}, (q_i, s_i, R_i)\}$ and update the GP.
```

5 Experiments

In these experiments, we study the impact of the parameters such as the step-size parameter α, the discount rate γ, and the training budget B, for a fixed exploration-exploitation trade-off parameter $\xi = 0.01$ on the global reward for selecting the data preparation strategy before a considered ML task. The code and datasets are available at: `https://github.com/LaureBerti/Learn2Clean`.

We evaluate how these parameters influence the learning performance of Q-learning in both cases: with versus without active reward learning. We used the House Price

Table 2. Learn2Clean results on House Price dataset with regular Q-learning

ML task	ML Model	Pred. Var.	Sequence of Actions for Data Preparation	Metric Name	Metric Value	Time (s)
Reg.	MARS	SalePrice	$IQR \rightarrow MICE \rightarrow MARS$	MSE	0.231	134.5
	LASSO	SalePrice	$KNN \rightarrow ZSB \rightarrow LASSO$	MSE	1.44E-12	24.2
	OLS	SalePrice	$KNN \rightarrow ZSB \rightarrow OLS$	MSE	2.20E-25	45.1
Classif.	NB	SaleCondition	$LC \rightarrow MF \rightarrow NB$	Accuracy	0.518	20.1
	CART	SaleCondition	$MM \rightarrow AD \rightarrow CART$	Accuracy	0.4489	58.7
Clust.	HCA	//	$MM \rightarrow ED \rightarrow HCA$	Silhouette	0.844	32.2
	KMEANS	//	$MM \rightarrow MR \rightarrow LOF \rightarrow ED \rightarrow KMEANS$	Silhouette	0.679	29.3

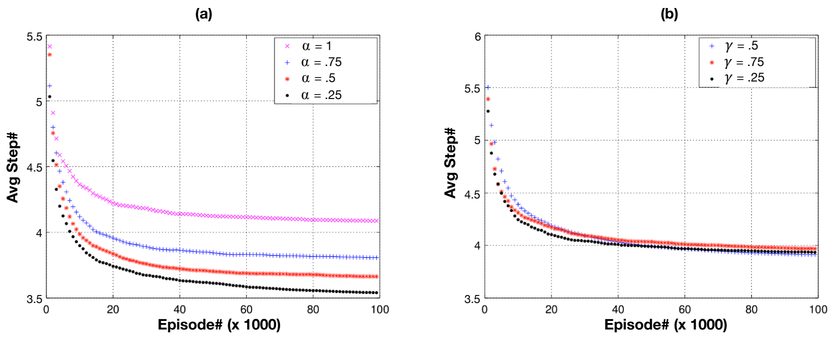

Fig. 3. Regular Q-learning: Average number of steps with varying α with fixed $\gamma = .5$ in (a) and varying γ with fixed in $\alpha = .25$ (b) for $B = 0$ and $\xi = .01$.

dataset from Kaggle[6] with 81 variables and 1.46k observations for clustering, regression, and classification. We ran Learn2Clean and obtained the results presented in Table 2 for various methods, namely: LASSO, OLS, and MARS for regression, KMEANS and HCA for clustering, and Naive Bayes and CART for classification. The quality metric, execution time (in seconds), and data preparation strategy are reported in the table. Data preparation strategies include various tasks such as K-nearest neighbors (KNN) or MICE imputation, outlier detection using InterQuartile Range (IQR), Z-score-based (ZSB), or LOF method, exact duplicate removal (ED), MinMax normalization (MM), and linearly correlated (LC) feature selection as illustrated in Figure 2 (see [1] for more details about the preprocessing methods).

We compare regular Q-learning and active reward learning with the average number of steps. The average steps are calculated at every 100 episodes by dividing the total steps from the first episode to the last by the total number of episodes. Figure 3(a) shows the average number of steps with different step-size parameter α values for regular Q-learning. We compare $\alpha = .25, .5, .75, 1$ with a fixed $\gamma = .5$ and $B = 0$. As the α is smaller, the average number of steps also decreases. Lower step-size values perform better. This indicates that accumulated experience affects value estimation more significantly than recent experience.

For the discount rate γ experiment, we fixed $\alpha = 0.25$ and vary $\gamma = .25, .5, .75$ for $B = 0$ (regular Q-learning). The average number of steps with different γ values are

[6] `https://www.kaggle.com/datasets`

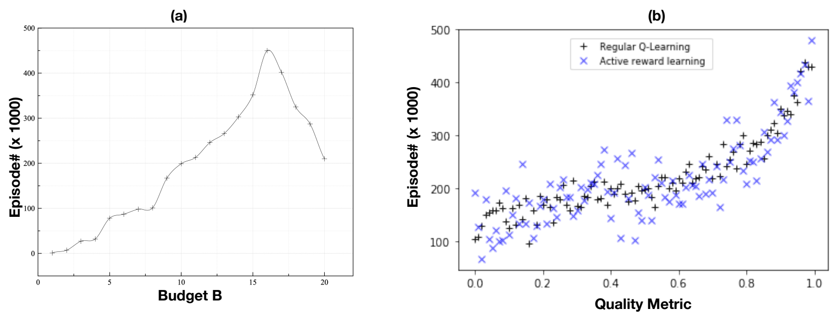

Fig. 4. Active Reward Learning: Average number of episodes with varying B from 1 to 20 with fixed $\alpha = .25$, $\gamma = .25$, and $\xi = .01$ in (a) and quality performance in (b).

shown in Figure 3(b). The lowest $\gamma = .25$ performs better. This means that immediate rewards are more important than future rewards with regular Q-learning. As episodes continue, a higher $\gamma = .75$ is slightly better. Relatively more steps are needed to achieve the same goal. In this case, future rewards are more significant than current rewards.

We then compare regular Q-learning (with no human in the loop) with active reward learning involving the user. For experiments, we select the parameter values shown previously. The step-size parameter α is set to 0.25 because low learning rate seems to be the most appropriate to our problem. The discount rate γ is set to .25 as well.

Figure 4(a) shows the average number of episodes with different B values from 1 to 20, given $\alpha = .25$ and $\gamma = 0.25$. The average number of episodes first increases dramatically with B and then decreases after $B = 16$. In the beginning, exploration is relatively inexpensive but turns out to be less effective when the user tends to systematically disagree with the strategy recommended by the system via Q-learning. The more disagreements (as negative samples in the online training), the more steps are needed to converge and reach for a sub-optimal solution yet maximizing the quality performances. Eventually, when sufficient consensual knowledge is accumulated, exploitation is worthy. As immediate future work, we plan to investigate further the impact of adversarial user inputs on the Q-learning strategy to mitigate the problem of noisy inputs.

6 Conclusions

We explored the use of reinforcement learning for data preparation with integrating human in the loop via active reward learning. The evaluation results provide empirical support for the effectiveness of our approach for the task at hand. We are planning more extensive studies with datasets from various application domains to experience a wider range of data preprocessing scenarios. We are also currently extending this work to consider a more dynamic environment to scale the approach to larger and more complex spatio-temporal datasets with more complex data preparation strategies.

References

1. Berti-Équille, L.: Learn2Clean: Optimizing the Sequence of Tasks for Web Data Preparation. In: Proc. of the The Web Conf 2019 (2019)
2. Fan, W.: Data quality: From theory to practice. SIGMOD Record **44**(3), 7–18 (2015)
3. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In: Advances in Neural Information Processing Systems 28, pp. 2962–2970 (2015)
4. Ilyas, I.F., Chu, X.: Trends in cleaning relational data: Consistency and deduplication. Foundations and Trends in Databases **5**(4), 281–393 (2015)
5. Konda, P., Das, S., C., P.S.G., Doan, A., Ardalan, A., Ballard, J.R., Li, H., Panahi, F., Zhang, H., Naughton, J.F., Prasad, S., Krishnan, G., Deep, R., Raghavendra, V.: Magellan: Toward building entity matching management systems. PVLDB **9**(12), 1197–1208 (2016)
6. Konda, P., Das, S., Suganthan G. C., P., Doan, A., Ardalan, A., Ballard, J.R., Li, H., Panahi, F., Zhang, H., Naughton, J., Prasad, S., Krishnan, G., Deep, R., Raghavendra, V.: Magellan: Toward building entity matching management systems. Proc. VLDB Endow. **9**(12), 1197–1208 (Aug 2016)
7. Krishnan, S., Franklin, M.J., Goldberg, K., Wu, E.: Boostclean: Automated error detection and repair for machine learning. CoRR **abs/1711.01299** (2017)
8. Matthew Hoffman, E.B., de Freitas, N.: Portfolio allocation for Bayesian optimization. In: 27th Conference on Uncertainty in Artificial Intelligence (UAI2011) (2011)
9. Navarro, G.: Approximate string matching. In: Encyclopedia of Algorithms, pp. 102–106 (2016)
10. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press (2005)
11. Rekatsinas, T., Chu, X., Ilyas, I.F., Ré, C.: Holoclean: Holistic data repairs with probabilistic inference. PVLDB **10**(11), 1190–1201 (2017)
12. Sarawagi, S., Bhamidipaty, A.: Interactive deduplication using active learning. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 269–278. KDD ’02, ACM, New York, NY, USA (2002)
13. Schafer, J.: Analysis of incomplete multivariate data. Chapman & Hall (1997)
14. Sutton, R.S., Barto, A.G.: Introduction to Reinforcement Learning. MIT Press (1998)
15. Watkins, C.J.C.H., Dayan, P.: Technical note: Q-learning. Mach. Learn. **8**((3-4)), 1190–1201 (1992)
16. Yakout, M., Berti-Équille, L., Elmagarmid, A.K.: Don’t be scared: use scalable automatic repairing with maximal likelihood and bounded changes. In: Proc. of the ACM SIGMOD. pp. 553–564 (2013)