International Journal of Computing and Optimization
Vol. 9, 2022, no. 1, 1 - 12
HIKARI Ltd, www.m-hikari.com
https://doi.org/10.12988/ijco.2022.9830

Big Data Flow Errors Correction Model Based on Q-Learning

Sidi Mohamed Snineh

ENSET Mohammedia, Hassan II University, Casablanca, Morocco

Mohamed Youssfi

ENSET Mohammedia, Hassan II University, Casablanca, Morocco

Noureddine El Abid Amrani

ENSET Mohammedia, Hassan II University, Casablanca, Morocco

Omar Bouattane

ENSET Mohammedia, Hassan II University, Casablanca, Morocco

Abstract

In this article, we propose a model based on Q-learning to correct, in real-time, the frequent errors in the incoming flows in a company. When receiving a stream that contains structured data, and which may contain errors, it should be known what it is the succession of algorithms to be used to make a record go from a state where errors exist to a state final where the recording is clean. For this reason, we have used Q-learning to train agents to apply algorithms randomly until they learn to correct a type of structured data.

Keywords: Big Data, Reinforcement learning, Stream processing, Q-learning

1 Introduction

Data analysis in enterprises has been used since the emergence of data warehouses and data stores as well as analytical methods and techniques such as data mining,

text mining, images mining, etc. These methods were used for decision support and are applied to a more or less controllable quantity of data. Today, the era of Big Data is witnessing an explosion of data coming from multiple data sources of different types, such as social media, IOTs, sensors, and others. These data are widely used and analyzed, often in real-time, with algorithms and analysis methods to generate decision support models that can affect the future and competitiveness of companies.

This amount of data requires new strategies for data storage, processing, and analysis because over the past five years the world has created 90% of the data [1]. Processing and analyzing this huge amount of data in a timely manner is a priority for businesses in order to retain these customers and win over others. However, not all data of this size is immune to noise and errors that can distort decision-making. Indeed, several types of data with their sources are the cause of a wide variety of errors.

According to [2], insufficient or unreliable data causes companies to lose up to 14.2 million dollars, and the tools available in the market exhibit the characteristics such as profiling, analysis, cleaning, masking, data matching, and monitoring, but none of these features relate to data validation, especially in the era of Big Data which is the source of all decisions.

To clean up and correct these errors, we have proposed a model for cleaning, in real-time, errors in the structured data flow. This approach, which is based on a reinforcement learning technique called Q-learning, makes it possible to train agents to find a succession of cleaning algorithms for each record. This model brings several advantages:

- Agent training does not need data, only the environment created is sufficient to automate this task.
- This solution is applicable to other domains like robotic, self-driving cars, healthcare applications, engineering applications, games applications, etc.
- Our model is scalable and applicable to error correction of big data of different types: structured data, semi-structured data, and unstructured data.

2 Literature review

Q-learning (Watkins, 1989) is a means which enables agents to learn to act optimally in controlled Markovian domains [3]. This is one of the most widely used reinforcement learning approaches. Reinforcement learning is an artificial intelligence technique that allows an agent to learn by interacting with an environment, he undertakes different actions by failure and by success to accumulate experiences. It is based on a system of rewards and penalties to allow the agent to learn to solve a problem independently [4].

The figure 1 shows the main components of the standard model of reinforcement learning which are:

Agent: it is the learner and the decision maker. It interacts with everything that is outside of the agent: it is the environment.

Environment: In reinforcement learning, the "environment" is typically a set of states that the "agent" tries to influence through his choice of "actions" [5].
The **reward** (R), in an environment, indicates whether an agent makes the right or wrong choice (actions) and the agent according to these rewards tries to learn by maximizing the total rewards, which allows him to update his policy to arrive at the best and optimal policy [6].

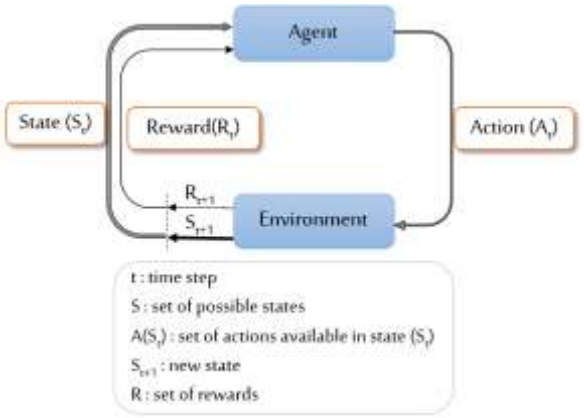

Fig. 1 Interaction agent-environment [6]

- **Mathematical formulation of RL problems**

Reinforcement learning is used in sequential decision-making problems where feedback is limited such as planning problems, game problems, robot control problems, etc. this type of problem can be formalized using Markov Decision Making (MDP). MDP is characterized by a quadruple (S, A, T, r) where:

- S : Set of finite states in the environment
- A : all actions in the environment
- T : S × A × S → [0,1] is the state transition function which describes the probability (1) of ending up in state s' when executing action "a" in state "s"

$$T(s,a,s') = P[S_{t+1} = s' | A_t = a, S_t = s] \quad (1)$$

- r : is a function of S → A in $\mathbb{R}$ which indicates the reward received by the system after each state transition, it is the expectation E of R_{t+1} knowing that $S_t = s, A_t = a, S_{t+1} = s'$

$$r(s,a,s') = E[R_{t+1}] \mid S_t = s, A_t = a, S_t = s'$$

In order to resolve an MDP, we will define a policy that characterizes the behaviour of the agent as well as a prediction of the reward obtained by this policy. This resolution consists in maximizing the return Gt that represents the sum of all the rewards while giving less importance to future rewards.

$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots with\ \gamma \epsilon [0,1]$ or $G_t = R_{t+1} + \gamma G_{t+1}$

Politics (π) is the way in which the agent, who perceives and acts in an environment, will behave in certain situations. The goal of reinforcement learning is to find the

optimal policy [6] which seeks to maximize the sum of the rewards. The measure of the rewards for an action a in a state s following a policy π is defined by the action value function (2):

$$Q_\pi(s,a) = E_\pi\{G_t|S_t = s, A_t = a\} \quad (2)$$

The goal of reinforcement learning is to choose actions that lead to maximum rewards which can be represented by the action value function $Q_\pi(s,a)$. The function optimal action value is the maximum action value function over all policies denoted by (3):

$$Q*(s,a) = max_\pi\big(Q*(s,a)\big) \quad (3)$$

Q-learning is a reinforcement learning algorithm without a model, it makes it possible to find an optimal policy in the sense of maximizing the expected value of the total reward on all successive stages, starting from the current state. Q-Learning assesses the quality of an action taken to move to a state rather than determining the possible value of the state to which it is moved. The heart of the Q-learning algorithm is a Bellman equation in the form of a simple value iteration update, using the weighted average of the old value and the new information (4):[7]

$$Q(s,a)_{new} = Q(s,a)_{current} + \alpha * (R(s,a) + \gamma * Max(next\ state, all\ actions)_{MRPA} - Q(s,a)_{current}) \quad (4)$$

- γ: defines how much we will give importance to future rewards, it can have a value in 0 and 1. The closer it is to 1 the program will treat future rewards almost the same way, on the contrary, more Gamma is less than 1 importance to diminished future rewards. In our model, we have chosen a value close to 1.
- MRPA: Maximum future reward expected given the new state and all possible choices in that new state.
- α : learning rate

The Q-learning algorithm uses a Q-table (Q-value) of action-state values. In Q-table, the rows represent states and the columns represent actions. The intersection between each row and each column is a cell that contains the estimated Q-value for the corresponding state-action pair. Q-learning finds the optimal policy by learning the optimal Q-values for each state-action pair.

3 Proposed model

The importance of data reliability leads us to further rethink error management, for this reason, that we propose in this article a model that corrects, in real-time, errors in Big Data flows based on Q-learning. The idea is to deal with structured data flows where each record contains a set of fields, each field has a structured type, and possibly these data flows may contain errors. We start with the principle that we have a record that is in an initial state belonging to a state graph, like Figure 2, which passes from one state to another state by applying a succession of cleaning algorithms until that it reaches the final state where the record is cleaned.

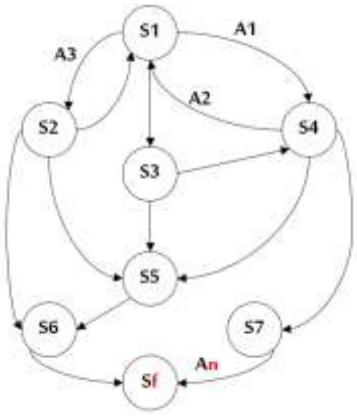

Fig. 2 Example of states graph

This approach consists of taking advantage of the company repository [8] which stores, among other things, information on frequent errors detected in company data and cleaning algorithms specific to each error.

To find a succession of error cleaning algorithms in a structured data flow, our approach follows a multi-step process:

a) Definition and creation of the environment.
b) Construction and initialization of the rewards table.
c) Agent initialization.
d) Training of the agent in the created environment.
e) Construction of the Q-table, which is the agent's learning table.

Step a) consists in creating, according to the number of cleaning algorithms stored in the repository, an environment with several states (Figure 3 and Table 1). The state of the color white has a reward of 0, the state of the color yellow has a negative reward -1 and the state of the color green is the final state with a maximum reward of 100. Then the color red does not represent any state, that is, there is no action that causes a record to go into this red state unless there are new algorithms subsequently that can change that state.

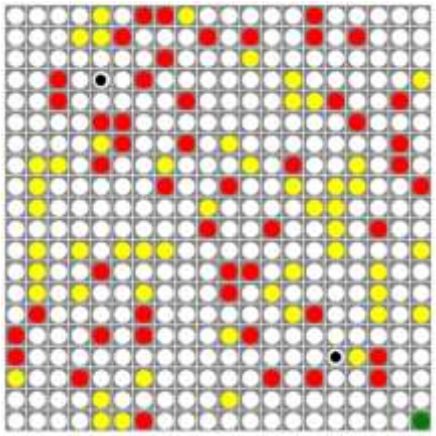

Fig. 3 Environment of our model

Table 1. Environment legend

States	Description
	Reward positive Reward negative Final state (record is clean) Watch state (S64 and S335)

The different states of the environment depend on the errors detected in the data flows, while the set of actions represents the set of algorithms proposed by the company to clean up frequent errors that can affect the quality of the data. Each action moves a record to another data state.

In this environment, agents are developed to train them and learn them to choose a succession of cleaning algorithms according to the initial state of each record.

Once the environment has been created, we go to step (b), the environment also defines a reward system represented by a square matrix of dimension the number of cleaning algorithms according to the values cited above. This matrix defines, for the choice of each algorithm, the possible reward for going to the next algorithm in order to correct the next error.

In step (c) the agent is initialized to an initial state according to the error detected in a record. The choice of this initial state is explained, below, in section (4.2. Choice of the initial state after the agent's training).

The following steps (d and e) consist of training the agent and creating the Q-table. The objective is to train the agent to evolve in the environment created, according to the initial state of the recording, by selecting the appropriate succession of cleaning algorithms. During the learning phase, the agent will try to explore the environment to correct a series of errors detected in several iterations, which represent the number of learning epochs.

At each iteration, the agent tries to repeat several choices, from an initial state, to act on the environment by randomly choosing one algorithm among several. The environment provides him with a reaction containing a reward allowing the agent to switch to another algorithm. This allows him to update his Q-Table learning table allowing him to have visibility on the choice of other cleaning algorithms for each environment. The Q-Table is a simple lookup table where we update the maximum expected future rewards to help the agent choose the best cleaning algorithm in each state. The Q-table is calculated using the Bellman equation (4).

Using this equation, each reward in the Q-table will be the maximum expected future reward the agent will receive if they choose a one-state error-cleaning algorithm. This is an iterative process, which allows us to improve the Q-Table with each iteration.

The final objective, therefore, is to maximize the Q-value function to help the agent choose the appropriate succession of algorithms for errors in a structured record.

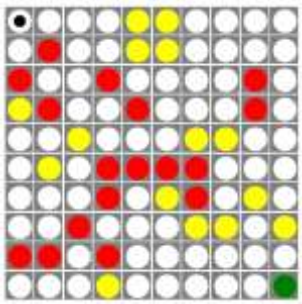

Fig. 4 Our model can work with different environments (100 states)

4 Implementation

4.1 Training and test

To validate our model, we carried out a project, which follows the steps of Q-learning, with the possibility of choosing a variable number of states (example Figure 3 and Figure 4). The results of training and testing the model obtained in this article are based on an environment of 400 states (0 to 399) (Figure 3) and a number of epochs equal to 1000. Before starting the training of our model, we have chosen to monitor two states, "watch states" to see the evolution of the values of the Q-value of the actions to be taken for each state and to have at the end the optimal value. The selected states are "S64" and "S335", one state far from the end state and another state close to the end state.

During training, the Q-value values of the two monitored states change from minimum values to maximum values. The figure 5, whose X-axis represents the iterations, that is to say, the epochs from 1 to 1000 and the Y-axis represents the values of Q-value, shows after the stop of the evolution of Q- value, the values of each iteration.

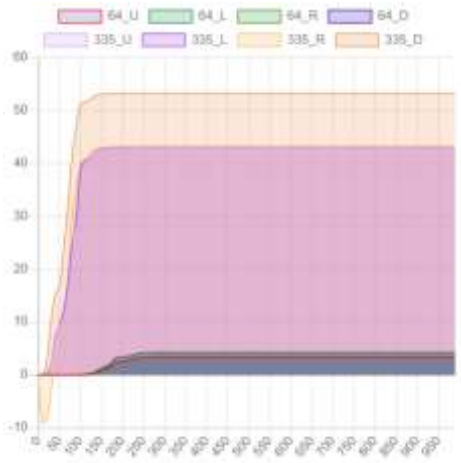

Fig. 5 This figure shows the evolution of the Q-values of the two watch states" "S64" and "S335"

Indeed, for each iteration and for each period of learning, we try to represent the Q-value of each action. For example, if we are monitoring a state that it has four possible actions to take, each of the actions in the Q-table has a calculated value that is updated on each iteration. As the algorithm advances, the system converges towards the maximum values.
In our project and as shown in the figure 5, the evolution of the values stopped at 1000th iterations, but as shown in the figure 6 and figure 7 we could stop the learning towards iteration 260 since maximum values are reached at the iterations 150 and 260 respectively for the watch states "S335" and "S64".

Fig. 6 The monitored state "S335" only reaches the maximum Q-value after about 150 iterations and shows that the maximum value is assigned to the "Down" action. This information can help us choose an adequate number of epochs during agent training.

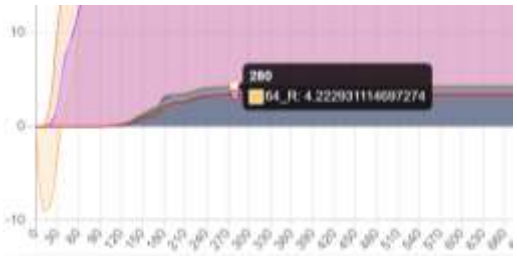

Fig. 7 This figure shows that the monitored state "S64" starts to reach the maximum value from iteration 260 and that almost all actions (U, D, L, R) have the same priorities.

After training the model, we generated the «Optimal Policy» table (figure 9) and we performed several tests to validate our model. For this reason, we started by testing the monitored states; according to figure 8, we notice that if we are in state "S335" the following action is «Down» 335_D which corresponds to the action generated in the table «Optimal Policy».

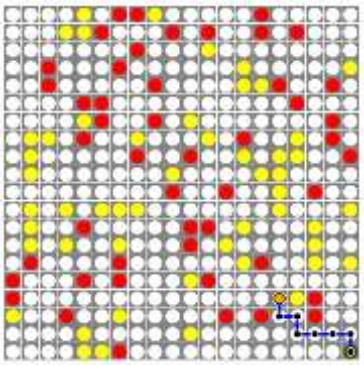

Fig. 8 This figure shows the result of the test if we start from the state «S335» which corresponds to the values of the «Optimal Policy» table.

We notice the same thing for the state "S64" whose next action is «Right» in figure 9. The Figure 10 shows the actions chosen to go to the end state.

Optimal Policy

State	Action.Label	Action.deltaX	Action.deltaY	Reward	Destination State
0	R	1	0	0	1
1	R	1	0	0	2
2	D	0	1	0	22
...					
63	R	1	0	0	64
64	R	1	0	0	65
65	D	0	1	0	85
67	R	1	0	0	68
...					
334	R	1	0	0	335
335	D	0	1	0	355
336	D	0	1	0	356
...					

Fig. 9 Extract from the «Optimal Policy» table generated after training the model

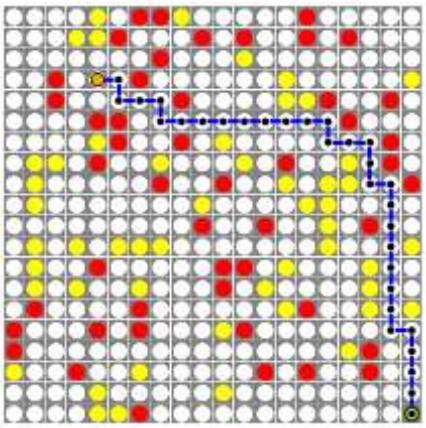

Fig. 10 This figure shows the Right action chosen by the agent after training from the watch state «S64» which corresponds to the values of the «Optimal Policy» table

Then we have chosen different initial states in figure 11, and we see that our model is well trained and arrives at the final state by choosing a succession of algorithms, which have the maximum Q-values.

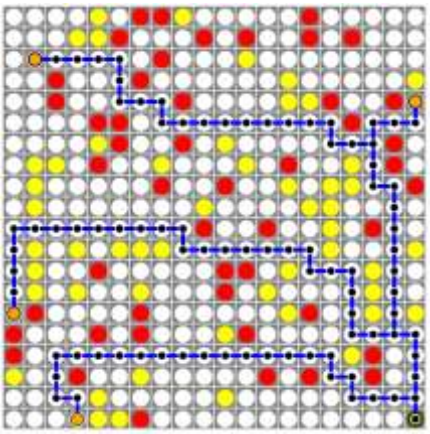

Fig. 11 This figure shows several tests of our model. Which shows that the model is well trained.

In order to do even more testing and verify that the model is able to find the best sequence of algorithms from any state. We started the test in a random way from state "S81" in figure 12 and when it arrived at state "S208" we forced it to jump to state "S91" then quickly it found a new succession of algorithms to the final state. Then, on reaching state "S318", it is made to jump again to state "S214", and with the same flexibility, it continues to the final state by choosing yet another succession of algorithms.

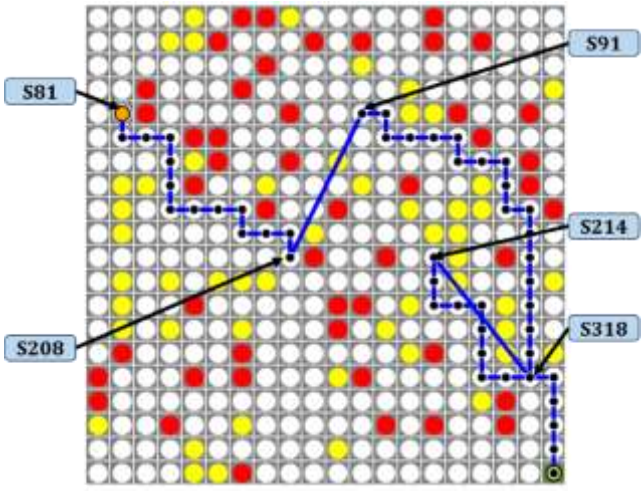

Fig. 12 This figure shows that the model is able to find the best succession of algorithms regardless of the initial state of a recording.

4.2 Choice of the initial state after the agent's training

After training the model and before starting to correct errors, in real time, in a data flow, the model must know the initial state of the current record in order to be able

to apply the corresponding succession of algorithms. For this reason, we have planned to use a neural network whose layers are fully connected (figure 13), which will have as input the fields of the record in question and as output the state of this same record as a function of the detected error. For this reason, we will adapt the solution that we proposed in the work [9] and which is based on a collection of micro-agents where each micro-agent is trained to detect a specific type of error using an atomic neural network based on a multi-layered perceptron sample. This solution will be the continuation of this work in the context of a new article using Deep Q-learning. Indeed, the basic working step for Deep Q-Learning is the initial state which is introduced into a neural network and returns the Q-value of all possible actions.

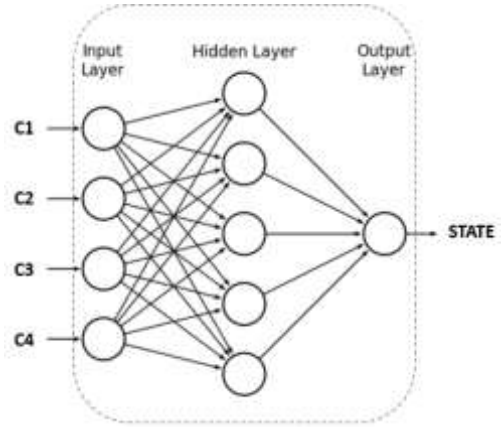

Fig. 13 Example of a fully connected neural network to determine the initial state of a record

5 Conclusion

In this article, we have proposed an approach that improves the quality of business data. The idea is to deal with structured data flows where each record contains a set of fields; each field has a structured type.

The approach consists to training agents using the Q-learning method to choose a succession of error cleaning algorithms in data flows.

Our model can use variable environments according to the errors detected in each company and a set of actions representing the cleaning algorithms proposed by the company. After training and upon arrival of data flows, the agent must use its Q-table to apply the best succession of algorithms to arrive at the final state where each record will be cleaned.

Our solution is extensible because we can train other agents in new situations and it is exportable because it is easy to integrate it into other companies since the agents are very small units easy to set up and manage.

The results obtained from our implementation are very satisfactory. In perspective, we project to use deep Q-learning by combining a neural network fully connected to find the initial state and our approach to improving this work.

References

[1] M. Chen, S. Mao, and Y. Liu, Big data: A survey, *Mob. Networks Appl.*, **19**, no. 2 (2014), 171–209. https://doi.org/10.1007/s11036-013-0489-0

[2] K. Sharma and V. Attar, Generalized Big Data Test Framework for ETL migration, *Int. Conf. Comput. Anal. Secur. Trends, CAST 2016*, no. April 2015, 528–532, 2017. https://doi.org/10.1109/CAST.2016.7915025

[3] C. J. C. H. Watkins and P. Dayan, Q-learning, *Mach. Learn.*, **8**, no. 3–4 (1992), 279–292. https://doi.org/10.1007/bf00992698

[4] H. nan Wang *et al.*, Deep reinforcement learning: a survey, *Front. Inf. Technol. Electron. Eng.*, **21**, no. 12 (2020), 1726–1744. https://doi.org/ 10.1631/FITEE.1900533

[5] L. Berti-Equille, Reinforcement learning for data cleaning and data preparation, *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2019.

[6] E. F. Morales and J. H. Zaragoza, An introduction to reinforcement learning, *Decis. Theory Model. Appl. Artif. Intell. Concepts Solut.*, (2011). 63–80. https://doi.org/10.4018/978-1-60960-165-2.ch004

[7] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, Q-Learning Algorithms: A Comprehensive Classification and Applications, *IEEE Access*, **7** (2019), 133653–133667. https://doi.org/10.1109/ACCESS.2019.2941229

[8] S. M. Snineh, M. Youssfi, O. Bouattane, A. Daaif, and O. E. K. Abra, Real-Time management model for frequent Big Data errors : Automatic Clean Repository for Big Data (ACR), *Int. Conf. Multimed. Comput. Syst. - Proceedings*, **2018** May (2018), 1–6. https://doi.org/10.1109/ICMCS.2018.8525920.

[9] S. M. Snineh, M. Youssfi, A. Daaif, and O. Bouattane, Micro agent and neural network based model for data error detection in a real time data stream, *Int. J. Adv. Comput. Sci. Appl.*, **10**, no. 7 (2019), 171–177. https://doi.org/10.14569/ijacsa.2019.0100725

Received: December 14, 2021; Published: January 3, 2022