ATENA: An Autonomous System for Data Exploration Based on Deep Reinforcement Learning

Ori Bar El, Tova Milo, and Amit Somech

Tel Aviv University

ABSTRACT

Exploratory Data Analysis (EDA), is an important yet challenging task, that requires profound analytical skills and familiarity with the data domain. While Deep Reinforcement Learning (DRL) is nowadays used to solve AI challenges previously considered to be intractable, to our knowledge such solutions have not yet been applied to EDA.

In this work we present ATENA, an autonomous system capable of exploring a given dataset by executing a meaningful *sequence* of EDA operations. ATENA uses a novel DRL architecture, and learns to perform EDA operations by independently interacting with the dataset, without any labeled data or human assistance. We demonstrate ATENA in the context of *cyber security* log analysis, where the audience is invited to partake in a data exploration challenge: explore real-life network logs, assisted by ATENA, in order to reveal underlying security attacks hidden in the data.

ACM Reference Format:
Ori Bar El, Tova Milo, and Amit Somech . 2019. ATENA: An Autonomous

System for Data Exploration Based on Deep Reinforcement Learning. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3357384.3357845

1 INTRODUCTION

Exploratory Data Analysis (EDA) is an important procedure in any data-driven discovery process. It is ubiquitously performed by data scientists and analysts in order to better understand the nature of their datasets and to find clues about their properties and underlying patterns.

EDA is known to be a difficult process, especially for non-expert users, since it requires profound analytical skills and familiarity with the data domain. Hence, multiple lines of previous work are aimed at facilitating the EDA process[1, 3, 5, 9, 12], suggesting solutions such as simplified EDA interfaces for non-programmers (e.g., Tableau[1], Splunk[2]), and analysis recommender-systems that assist users in formulating queries [5, 9] and in choosing data visualizations [12]. Still, EDA is predominantly a manual, non-trivial process that requires the undivided attention of the engaging user.

In recent years, artificial intelligence systems based on a DRL paradigm surpass human capabilities in a growing number of complex tasks, such as playing sophisticated board games, autonomous driving, and more [6]. In such solutions, an artificial agent, controlled by a deep neural-network, learns how to perform tasks merely by trial and error, without any human data or guidance (e.g. labeled samples, heuristics). While DRL is a promising, novel paradigm, to our knowledge such solutions have not yet been applied to EDA.

In this work, we showcase ATENA, a prototype EDA "assistant", based on a DRL architecture. The goal of ATENA is to be able to take a new dataset, and automatically "guide" the human analyst through it, by performing an entire *sequence* of analysis operations that highlight *interesting* aspects of the data. Importantly, ATENA learns to perform meaningful EDA operations by independently interacting with the dataset, and do not require training data or any human assistance or supervision.

Typically in DRL, an artificial *agent* operates within a specific predefined setting, referred to as an *environment*. The environment controls the input that the agent perceives and the actions it can perform: At each time t, the agent observes a *state*, and decides on an *action* to take. After performing an action, the agent obtains a positive or negative *reward* from the environment, either to encourage a successful move or discourage unwanted behavior. The goal of the agent is learning how to obtain a maximal cumulative reward, by leveraging the experience gained from repeatedly interacting with the environment.

In the context of autonomous EDA, using a DRL based system can be highly beneficial. For instance, as oppose to current solutions for EDA assistance/recommendations that are often heavily based on users' past activity [5, 9] or real-time feedback [3], a DRL-based solution has no such requirements since it trains merely from self-interactions. Also, since its training process occurs offline, a DRL-based system may be significantly more efficient in term of running times, compared to current solutions that rely on computing recommendations at interaction time [12].

However, designing a DRL solution for EDA poses theoretical and practical challenges that we tackle in this work:

(1) Devise a machine-readable encoding for EDA operations and result sets. Typically in EDA, datasets are often large and comprise values of different types and semantics. Also, EDA interfaces support a vast domain of possible analysis operations which may have a compound result sets, containing layers such as grouping and aggregations. To facilitate the computerized interaction with the dataset, we devise a DRL *environment* in which

[1]https://www.tableau.com
[2]https://www.splunk.com

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3357845

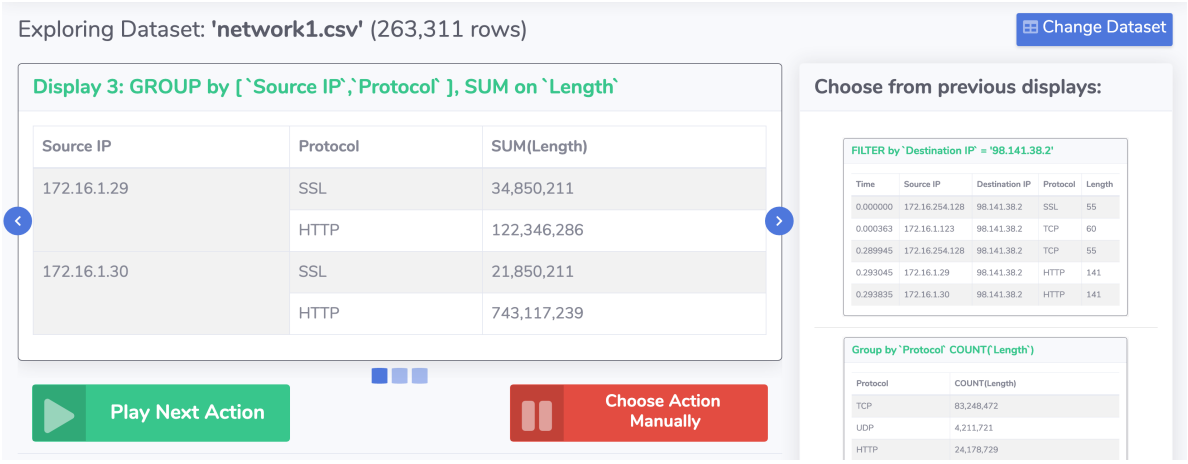

Figure 1: ATENA's User Interface

EDA operations are parameterized (rather then using, e.g., a free-form SQL interface) - the agent first chooses the operation type, then the adequate parameters. The operation is then performed on the dataset, and the agent receives a machine-readable encoded representation of its results.

(2) Formulate a reward system for EDA operations. A crucial component in any learning based system is an explicit and effective loss/reward function, which is used in the optimization process of the system's model. To our knowledge, there is no such explicit reward definition for EDA operations. We therefore design a compound reward signal, encouraging the agent to perform a sequence of analysis operations that are: (i) interesting - we employ a data-driven notion of *interestingness*; (ii) diverse from one another - we use a distance metric between actions' result sets, and (iii) human understandable - we utilize a weak-supervision based classifier [10] that employs a set of hand-crafted rules, and a small set of EDA operations made by human experts as an exemplar.

(3) Design and implement a DRL agent architecture that can handle thousands of different EDA actions. Typically in DRL, at each state the agent chooses from a finite, small set of possible actions. However, even in our simplified EDA environment there are over $100K$ possible distinct actions. Since off-the-shelf DRL architectures are inefficient in our case, we devise two novel solutions for handling the large, yet parameterized action space, and for choosing dataset tokens as action parameters.

A short paper describing our initial system design was recently published in [8]. Based on that design, we developted a working prototype of ATENA, presented in this work.

Demo Scenario. We demonstrate the ability of ATENA to effectively explore datasets in the context of *cyber security log analysis* (However, note that ATENA can explore datasets of any application domain). The audience will be invited to partake in a data exploration challenge: given a dataset of network traffic logs, the task is to effectively explore it in order to shed light on an underlying security attack that is hidden in the data (e.g. denial-of-service, port scan). Using ATENA UI (see Figure 1) the audience will examine the sequence of EDA operations performed by the artificial agent. We show that the EDA process guided by ATENA is more effective and faster, compared to a manual EDA interface.

2 ATENA SYSTEM DESCRIPTION

We first briefly overview basic concepts and notations for EDA and Reinforcement Learning (RL), then describe the architecture of ATENA w.r.t. the challenges and components described in the introduction: (1) the EDA environment, comprising a machine-readable encoding for analysis operations and result sets - Section 2.2, (2) a novel reward signal for EDA operations - Section 2.3, and (3) the DRL agent architecture with our solution for handling thousands of different EDA actions - Section 2.4.

2.1 Technical Background

To set the ground for our work, we recall basic concepts and notations for EDA and RL.

The EDA Process. A (human) EDA process begins when a user loads a particular dataset to an analysis UI. The dataset is denoted by $\mathcal{D} = \langle Tup, Attr \rangle$ where Tup is a set of data tuples and $Attr$ is the attributes domain. The user then executes a series of analysis operations (e.g., SQL queries) $q_1, q_2, ..q_n$, s.t. each q_i generates a results *display*, denoted d_i. The results display often contains the chosen subset of tuples and attributes of the examined dataset, and may also contain more complex features (supported by the particular analysis UI) such as grouping and aggregations, results of data mining operations, visualizations, etc.

Reinforcement Learning. Typically, DRL is concerned with an agent interacting with an environment. The process is often modeled as a Markov Decision Process (MDP), in which the agent transits between state by performing actions. At each step, the agent obtains an observation from the environment on its current state, then it is required to choose an action. According to the chosen action, the agent is granted a reward from the environment, then transits to a new state. We particularly use an episodic MDP model: For each episode, the agent starts at some initial state s0, then it continues to perform actions until reaching a terminus state. The utility of an episode is defined as the cumulative reward obtained for each action in the episode. The goal of a DRL agent is learning how to achieve the maximum expected utility.

2.2 RL-EDA Environment

We developed a simple yet extensible RL-EDA environment (See Figure 2), in which the actions are EDA operations and the states are their corresponding results sets.

Each environment contains a collection of datasets - all sharing the same schema, yet their instances are different (and independent). In each episode (i.e., exploration session) of a predefined length N, the agent is given a dataset $\mathcal{D}$, chosen uniformly at random, and is required to perform N consecutive EDA operations. However, in order for a DRL agent to intelligently choose an EDA operation, an effective, machine-readable encoding is required for EDA operations and result sets, as described next.

EDA Operations Representation. In general, analysis operations can be expressed using query languages (e.g. SQL), which is difficult for a machine agent to generate from scratch (to date). To simplify, we therefore use *parameterized* analysis operations, that allow the agent to first choose the operation type, then the adequate parameters. Each such operation takes some input parameters and a previous display d (i.e., the results screen of the previous operation), and outputs a corresponding new results display. For the prototype version of ATENA, we use a limited set of analysis operations (to be extended in future work):

FILTER($attr, op, term$) - used to select data tuples that matches a criteria. It takes a column header, a comparison operator (e.g. $=, \geq, contains$) and a numeric/textual term, and results in a new display representing the corresponding data subset (An example FILTER operation is given at the bottom of Figure 2).

GROUP($g\_attr, agg\_func, agg\_attr$) - groups and aggregates the data. It takes a column to be grouped by, an aggregation function (e.g. SUM, MAX, COUNT, AVG) and another column to employ the aggregation function on.

BACK() - allows the agent to backtrack to a previous display (i.e the results display of the action performed at $t-1$) in order to take an alternative exploration path.

In a similar manner, more EDA operations (e.g. visualizations, joins) can be integrated in ATENA.

Encoding Result Displays. Result displays are often compound, containing both textual and numerical data which may also be grouped or aggregated. However, a low-level numerical representation of the displays is required to maintain stability and reach learning convergence. In our model, the agent observes at time t a numeric vector, denoted $\vec{d_t}$, that represents a compact, structural summary of the results display d_t obtained after executing an analysis operation at $t-1$. $\vec{d_t}$ contains the following numerical values: (1) three descriptive features for each attribute: its values' entropy, number of distinct values, and the number of null values. (2) one feature per attribute stating whether it is currently grouped/aggregated, and three global features storing the number of groups and the groups' size mean and variance.

In the first state in an episode, denoted s_0, the agent observes the numeric representation of the initial display d_0, i.e., the dataset instance before any action was taken.

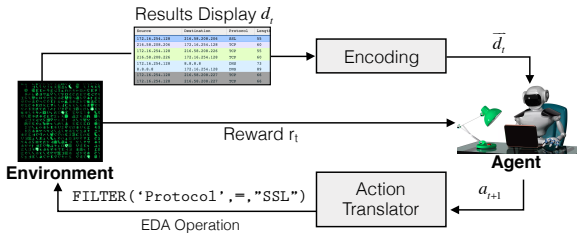

Figure 2: DRL Environment for EDA

2.3 Reward signal for EDA operations

We devise a reward signal for EDA actions with three goals in mind: (1) Actions inducing *interesting* results set should be encouraged. (2) Actions in the same session should yield *diverse* results describing different parts of the examined dataset, and (3) the actions should be *coherent*, i.e. understandable to humans. Correspondingly, our reward signal comprises the following components:

(1) Interestingness. In our prototype we employ the *Compaction-Gain* [2] method to rank GROUP actions (which favors actions yielding a small number of groups that cover a large number of tuples). To rank FILTER actions we use a relative, deviation-based measure (following [12]) that favors actions' results that demonstrate significantly different trends compared to the entire dataset. However, other measures from the literature can be used alternatively.

(2) Diversity. We use a simple method to encourage the agent to choose actions inducing new observations of different parts of the data than those examined thus far: We calculate the average of the Euclidean distances between the observation vector $\vec{d_t}$ (representing the current display) and the vectors of all previous displays obtained at time $< t$.

(3) Coherency. Ranking the coherency of an analysis operations is performed using an external classifier. Given the datasets' schema and application domain we use a set of heuristic classification-rules composed by domain experts (e.g. *"a group-by employed on more than four attributes is non-coherent"*), then employ Snorkel [10] to build a weak-supervision based classifier that lifts these heuristic rules to predict the coherency level of an analysis operation.

The overall reward is a weighted sum of the components above (Weights are hyper-parameters of ATENA).

2.4 DRL Agent Architecture

Different than most DRL settings, in our EDA environment, the action-space, comprising all possible EDA operations - is parameterized, very large, and discrete. Therefore, directly employing off-the-shelf DRL architectures is extremely inefficient since each distinct possible action is often represented as a dedicated node in the output layer (see, e.g. [4, 6]). We therefore use a novel, twofold solution to decrease the size of the network:

1. Efficient action selection. Our model utilizes the parametric nature of the action space, and allows the agent to choose an action type and a value for each parameter, rather than choosing a single action out of the entire action domain. This design significantly reduces the size of the output layer of the actor-network. To do so, we change the actor-network architecture as follows (See Figure 3): we add a "pre-output" layer, containing a node for each action type, and a node for each of the parameters' values. Then, by employing

a "multi-softmax" layer, we generate separate probability distributions, one for the action types and one for each parameters' values. Finally, the action selection is done according to the latter probability distributions, by first sampling from the distribution of the action types, then by sampling the values for each of its parameters.

2. Efficient selection of dataset values for FILTER "term" parameter. The parameter *term* of the FILTER operation may be prohibitively large, even when the selection is restricted only to tokens appearing in the current results display.

Therefore, to avoid having a dedicated node for each dataset token in our "pre-output" layer, we use a simple, effective solution that maps the individual tokens to a single yet *continuous* parameter. The mapping is done according to the *frequency of appearances* of each token in the current display. Finally, instantiating this parameter is done merely with two entries in our "pre-output" layer: a mean and a variance of a Gaussian (See Figure 3). A numeric value is then sampled according to this Gaussian, and translated back to an actual dataset token by taking the one having the closest frequency of appearance to the value outputted by the actor-network.

User Interface. ATENA features user interaction via a designated UI (See Figure 1). First, the user chooses a dataset to explore. ATENA then begins an automatic exploration process, allowing the user to examine the results display of each EDA operation. At any time, the user can pause the automatic exploration, and traverse back and forth through the different displays.

3 DEMONSTRATION

We demonstrate how ATENA effectively assist users in exploring datasets in the context of *cyber security log analysis*.

Demonstration datasets. We collected 4 datasets containing network traffic logs, obtained from online cyber-security challenges (e.g. the HoneyNet Project[3]). Each dataset contains a unique security attack hidden in the data, such as a denial-of-service attack on a Web server, port-scanning issued by an attacker on an internal network IP range, etc. All datasets share the same schema of 12 attributes and contains between 8K to 200K tuples.

Implementation and Training. We implemented the DRL-EDA environment as described in Section 2 and plugged the datasets collection in it. We fixed the number of EDA operations to 10 per episode. We then trained an ATENA agent, using the Advantage Actor Critic (AAC) basic architecture with the Proximal Policy Optimization algorithm [11]. The model hyper-parameters (including the reward weights) were tuned to obtain a maximal reward.

Engaging the Audience. We will begin with an overview of ATENA and essential concepts in cyber security, then invite the audience to partake in a data exploration challenge: volunteers will be asked to choose a dataset from our collection, and perform a data exploration process in order to shed light on the underlying security attack hidden in the data.

The participants will be split into two groups: one using ATENA, where the exploration is performed automatically, and the second will use a standard, manual EDA interface. Once the exploration is completed, the participants will be asked to describe the timeline of the security attack found in the data, and will be given a correctness score. Then, in order to gauge the effectiveness of ATENA we will compare the average correctness scores of the two groups, as well as the overall exploration time. Finally, we will present a "behind the scenes" view of the system, allowing the participants to examine the reward obtained for each EDA operation performed in their session, the machine-readable encoding of the result sets, information on the training process of the system, and more.

[3]https://www.honeynet.org

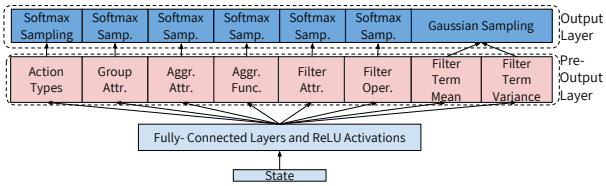

Figure 3: ATENA's Neural Network Architecture.

Related Work. As mentioned above, a battery of tools have been developed over the last years to assist analysts in interactive data exploration[1, 3, 5, 7, 9, 12], by e.g. suggesting adequate visualizations [12], SQL/OLAP query recommendations [1, 5], and recommendations of general exploratory steps [9]. Particularly, [3] presents a system that iteratively presents the user with interesting samples of the dataset, based on manual annotations of the tuples. Different from previous work, ATENA is a completely autonomous agent, capable of self-learning how to intelligently perform a meaningful *sequence* of EDA operations on a given dataset.

DRL is unanimously considered a breakthrough technology, used in solving a growing number of AI challenges previously considered to be intractable (See [6] for a survey). Our system is backed by state-of-the-art methods such as the Advantage Actor Critic (AAC) model [6] and the Proximal Policy Optimization (PPO) algorithm [11], used to optimize the training process.

ACKNOWLEDGEMENTS

This work has been partially funded by the Israel Innovation Authority, the Israel Science Foundation, the Binational US-Israel Science foundation, Len Blavatnik and the Blavatnik Family foundation.

REFERENCES

[1] J. Aligon, E. Gallinucci, M. Golfarelli, P. Marcel, and S. Rizzi. A collaborative filtering approach for recommending olap sessions. *ICDSST*, 2015.
[2] V. Chandola and V. Kumar. Summarization - compressing data into an informative representation. *KAIS*, 12(3), 2007.
[3] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Aide: An active learning-based approach for interactive data exploration. *TKDE*, 2016.
[4] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.
[5] M. Eirinaki, S. Abraham, N. Polyzotis, and N. Shaikh. Querie: Collaborative database exploration. *TKDE*, 2014.
[6] Y. Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.
[7] T. Milo, C. Ozeri, and A. Somech. Predicting "what is interesting" by mining interactive-data-analysis session logs. In *EDBT*, 2019.
[8] T. Milo and A. Somech. Deep reinforcement-learning framework for exploratory data analysis. In *aiDM*, 2018.
[9] T. Milo and A. Somech. Next-step suggestions for modern interactive data analysis platforms. In *KDD*, 2018.
[10] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré. Snorkel: Rapid training data creation with weak supervision. *PVLDB*, 11(3), 2017.
[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.
[12] M. Vartak, S. Rahman, S. Madden, A. Parameswaran, and N. Polyzotis. Seedb: efficient data-driven visualization recommendations to support visual analytics. *PVLDB*, 8(13), 2015.