Automatically Generating Data Exploration Sessions Using Deep Reinforcement Learning

Ori Bar El, Tova Milo, and Amit Somech
Tel Aviv University, Israel

ABSTRACT

Exploratory Data Analysis (EDA) is an essential yet highly demanding task. To get a head start before exploring a new dataset, data scientists often prefer to view existing *EDA notebooks* – illustrative, curated exploratory sessions, on the same dataset, that were created by fellow data scientists who shared them online. Unfortunately, such notebooks are not always available (e.g., if the dataset is new or confidential).

To address this, we present ATENA, a system that takes an input dataset and auto-generates a compelling exploratory session, presented in an EDA notebook. We shape EDA into a control problem, and devise a novel Deep Reinforcement Learning (DRL) architecture to effectively optimize the notebook generation. Though ATENA uses a limited set of EDA operations, our experiments show that it generates useful EDA notebooks, allowing users to gain actual insights.

ACM Reference Format:
Ori Bar El, Tova Milo, and Amit Somech. 2020. Automatically Generating Data Exploration Sessions Using Deep Reinforcement Learning. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD'20), June 14–19, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3318464.3389779

1 INTRODUCTION

Exploratory Data Analysis (EDA) is an essential component in the data science (DS) pipeline, performed by any data scientist when examining a new dataset – with the goal of better understanding its nature and characteristics.

EDA is widely known to be a cumbersome process, therefore, numerous systems have been devised to facilitate this process: simplified visual interfaces (e.g., [24], Tableau [2]), data-driven tools for surfacing interesting subparts of a data cube [37], interesting data visualizations [39, 45], as well as systems for recommending next exploratory steps [29], and automatic discovery of related datasets [10].

Yet, perhaps surprisingly, an increasingly popular method for data scientists to get a head start on their EDA process is to view *EDA notebooks* – curated, illustrative exploratory sessions prepared by other data scientists [23, 36]. Such EDA notebooks are presented in a *notebook interface*, a literate programming environment that allows users to easily document, and share, a sequence of programmatic operations and their results.

Existing EDA notebooks are typically available on data science (DS) or code sharing platforms such as Kaggle [1] and GitHub, in which users who already performed EDA on a specific dataset (hosted on the platform as well), summarize and curate it into a compelling EDA notebook to be shared with the community. Then, when other data scientists begin working on the same dataset, they first view its accompanied EDA notebooks and trace the exploratory steps in them, to learn how other users approached the dataset, what insights they already discovered, and to get ideas for interesting patterns to further examine when later performing EDA themselves [36]. Consider the following example:

EXAMPLE 1.1. *Data scientist Clarice wants to discover causes for flight delays, using the US flights dataset [32]. Before she begins exploring the dataset, she examines one of its accompanied EDA notebooks, containing a sequence of EDA operations and their results: The first operation is a group-by, showing the average flight delay per month of the year. Examining its results, she can immediately see that delays are longer in June, compared to the rest of the year. The second operation in the notebook is a filter operation, selecting only flights that took place in June. Sifting through the results does not yield immediate insights, so she scrolls down to examine the following operation in the sequence – a group-by over the June flights, depicting the average delay by origin airport. Clarice can quickly see that delays during June in LAX and ATL are considerably longer than in the rest of the airports.*

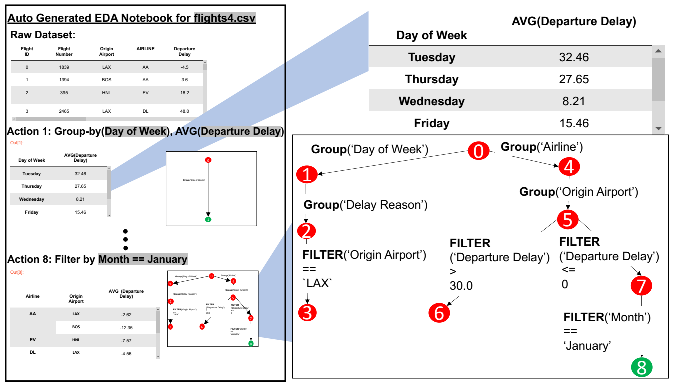
Viewing the notebook makes Clarice's own EDA process much easier: She already gained some insights on particular cases where delays are longer. Also, by tracing the user's exploratory steps in the notebook she can infer, to some degree, how they approached the dataset, and when relevant – reapply parts of their methodology for her own process.

SIGMOD'20, June 14–19, 2020, Portland, OR, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-6735-6/20/06…$15.00
https://doi.org/10.1145/3318464.3389779

While existing EDA notebooks are useful, unfortunately they are not always available (e.g., when a dataset is new, confidential, or has not yet been examined on the given DS platform). To address this, we present ATENA, a system for auto-generating EDA notebooks. ATENA takes as input a relational dataset, and automatically derives and performs an exploratory session – a meaningful, compelling sequence of EDA operations. The results of these operations are displayed to the users in a notebook interface (See Figure 1), guiding them through the dataset's highlights and important characteristics. Hereby, ATENA allows users to obtain preliminary insights and ideas for further exploration, even if notebooks made by real users are unavailable.

So what makes an EDA notebook helpful? Naturally, it should consist of *interesting* views of the data. But two other properties are equally as important [23, 36]: First, the EDA notebook should cover multiple, *diverse* aspects of the dataset, so that the user is able to learn about its different properties. Second, it should be *coherent* and easy to follow – the EDA operations need to be in a sensible order, in which subsequent operations are logically related (e.g., the second and third operation in the example above). ATENA is designed with these goals in mind. Our contributions are as follows:
i. Formulation of EDA as a control problem. We define the problem of generating EDA notebooks as a *control problem* using a Markov Decision Process (MDP) model. We define a *reward signal* that not only enforces that each EDA operation in the notebook is "interesting", but also that the *entire sequence* of operations is diverse and coherent, w.r.t. the input dataset. Additionally, ATENA can take as input a set of focal attributes, reflecting users' particular information needs (e.g., the *delay duration* in the example above) and consider them when generating the notebook.
ii. A Novel DRL architecture. As explained in the sequel, the EDA control problem is particularly challenging in our context, since the MDP contains an exceptionally large number of states and actions, and the reward signal is compound and non-differentiable. Therefore, to facilitate effective optimization for our goal we use a novel deep reinforcement learning (DRL) architecture, especially designed to tackle the large yet discrete action space (of possible EDA operations).
iii. Experimental evaluation and public benchmark. Our experimental evaluation, conducted on various input datasets, shows that the EDA notebooks generated by ATENA are insightful and easy to follow, allowing users to gain actual preliminary insights on their datasets. In addition to a comprehensive user study we also developed an automatic, reproducible benchmark [5] to facilitate the comparison of future models for auto-generating EDA notebooks.

Last, we note that a early, high-level descriptions of ATENA have been presented in [4, 28]. In comparison, this paper


Auto Generated EDA Notebook for flights4.csv

Raw Dataset:

Flight ID	Flight Number	Origin Airport	AIRLINE	Departure Delay
0	1839	LAX	AA	-4.5
1	1394	BOS	AA	3.6
2	395	HNL	EV	16.2
3	2465	LAX	DL	48.0

Action 1: Group-by(Day of Week), AVG(Departure Delay)

Out[1]:

Day of Week	AVG(Departure Delay)
Tuesday	32.46
Thursday	27.65
Wednesday	8.21
Friday	15.46

Action 8: Filter by Month == January

Out[8]:

Airline	Origin Airport	AVG (Departure Delay)
AA	LAX	-2.62
	BOS	-12.35
EV	HNL	-7.57
DL	LAX	-4.56

Group('Day of Week')
Group('Airline')
Group('Delay Reason')
Group('Origin Airport')
FILTER('Origin Airport') == `LAX`
FILTER ('Departure Delay') > 30.0
FILTER ('Departure Delay') <= 0
FILTER('Month') == 'January'
0
1
2
3
4
5
6
7
8

Figure 1: EDA Notebook Generated By ATENA

presents the laetest system in more detail, alongside a comprehensive experimental evaluation.

2 RELATED WORK

Assisting users in performing EDA and gaining insights from data has been the goal of many previous works in multiple research domains. We review selected relevant works.
Data tours and projection pursuit. Discovering interesting data "views" and presenting them in a coherent sequence for the purposes of data exploration dates back to the 1970's [16], where the focus was on generating 2-d projections (i.e., scatter plots) of multivariate, numeric data. *Projection Pursuit* [16] suggested an optimization based approach to find "interesting" 2-d projections of the data, while *data-tours* [3] focused on generating a coherent sequence of 2-d (arbitrary) projections, s.t. there is a low variation between subsequent views, and the sequence appears *continuous* to a human observer. While EDA notebooks may be regarded as an extension to the data "tour" experience, our work is fundamentally different in two parameters: (1) we assume that the dataset may contain attributes of heterogeneous (not necessarily numeric) types, including textual and categorical data, and (2) correspondingly, we focus on commonly used EDA operations such as: filtering, grouping, and aggregations (with extensions for visualizations, joins, etc.). Therefore, ATENA cannot rely on the particular optimization problem defined in [16] for 2-d projections, or the interpolation means of [3].
Data-driven generation of interesting views. More recent works suggest to auto-generating interesting views from an input dataset using different types of EDA operations, e.g. [12, 37] for interesting tuple-subsets, [21] for OLAP drill-downs, [45] for data-visualizations, and [41]. These works typically use heuristic measures of *interestingness* and search the space of all possible views, in order to return the one (or top-k) obtaining the highest interestingness scores.

ATENA also relies on a notion of interestingness, but uses additional menas in order to generate a compelling *sequence* of operations: (1) A compound reward signal that also considers the coherency and diversity of the EDA operations w.r.t. the previous operations and resulted data views in the sequence (2) An effective, novel DRL learning scheme to facilitate the optimization on the entire operations sequence.

Interactive EDA recommender systems. Numerous works produce next-step EDA recommendations for an ongoing exploratory session. While some use data-driven means as described above, other lines of work use external means such as a log of previous EDA operations [14, 29], and real-time feedback [11, 20] from the user in order to recommend possible exploratory steps at each point in the session. These works are shown to be useful for interactive EDA, and are complementary to ATENA, which focuses on the generation of EDA notebooks to be examined *before* one actively explores the data.

Complementary works for facilitating EDA. Several other aspects of assisting users in EDA have been considered in previous work. For example, systems like Vizier [8] facilitates the manual creation EDA notebooks, Data Polygamy [10] assist in finding related datasets, QUDE [47] ensures the safety of EDA automation tools by preventing false discoveries, and Northstar [24] enables non-programmers to efficiently perform interactive exploration in a compelling UI.

3 SYSTEM WORKFLOW

In a nutshell, ATENA works as follows: First, the user uploads a tabular dataset to the system, then is prompted to select, if desired, a set of focal attributes that she is particularly interested in. Next, an instance of an EDA control problem (i.e., an EDA environment and an objective function) is created w.r.t. the user's dataset and focal attributes – See Section 4 for more details. As explained, our EDA environment currently supports filter, group-by, and aggregation operations, yet can be extended to support, e.g., visualizations and joins (see Section 7 for a discussion).

Then a DRL learning scheme, as illustrated in Figure 2, is employed in order to generate an EDA notebook: First, the neural network of the DRL agent is initialized with random weights. Next, the agent "trains" on the dataset by self-interacting with the input dataset via the EDA environment: The environment (See Figure 2) allows the agent to first employ an EDA operation, then receive back an *observation-vector* that summarizes its results, as well as a positive/negative *reward* value, derived from the objective function (see Section 4.2). The goal of the DRL agent is to learn, by repeated interactions with the environment, how to perform a sequence of N (predefined) EDA operations which obtain a *maximal cumulative reward.* Once the learning phase is completed, the sequence of EDA operations which obtained

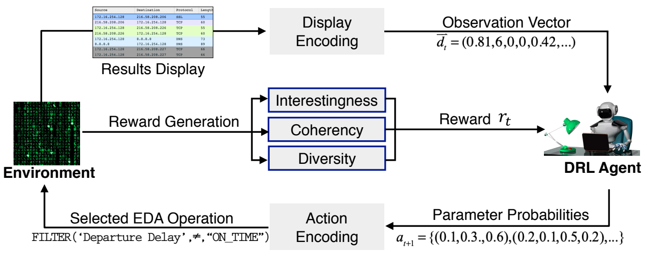


Figure 2: DRL Learning Scheme in ATENA

the highest cumulative reward is used to auto-generate the EDA notebook provided to the user.

User Experience. Figure 1 depicts an actual notebook auto-generated by ATENA on a *flights dataset* [32] (see Section 6.1 for a description) As shown in the figure, The EDA notebook lists the operations and their results in a chronological order (operations 2-7 are omitted for brevity). For improved readability, each operation is accompanied by a simple verbal description (e.g., *"filter by Month=='January'"*), and on the right-hand side of the notebook appears a dynamic tree-like illustration of the operations, allowing the user to easily follow the exploratory steps.

In what comes next, we present our MDP-based model for EDA, then explain why DRL is particularly suitable to solve it, and present a dedicated architecture we developed for it.

4 THE EDA CONTROL PROBLEM

We next explain how EDA is shaped into a control problem using an MDP model, then describe the reward signal.

4.1 MDP Model for EDA

Typically, in EDA, a user examines a dataset $\mathcal{D} = \langle Tup, Attr \rangle$, where Tup is a set of data tuples and $Attr$ is the attributes domain (We assume that the dataset includes attributes of different types, e.g. textual, numerical or categorical).

The user then executes a series of analysis operations (e.g., filter, aggregation, visualization, depending on the particular UI) $q_1, q_2, ..q_n$, s.t. each q_i generates a results *display*, denoted d_i. After examining the results display of operation q_i, the user decides if and which operation q_{i_1} to employ next.

We model an EDA process using an episodic MDP, which consists of a set of possible states, and a set of possible actions. Intuitively, in our case, the set of actions is the set of all possible (and supported) EDA operations, and the set of states correspond to their results displays. In a single *episode*, the agent explores a particular dataset $\mathcal{D}$ by preforming a predefined number of N actions. At each step, the agent obtains an observation vector describing its current state in the EDA session, then it is required to choose an action. According to the chosen action, the agent is granted a negative/positive reward then transits to a new state. The *utility* of an entire episode is defined as the cumulative reward, obtained for the actions in the current episode.

Next, we explain how actions and state observations are represented in our model, then present the reward signal.

EDA Action Space. Our model allows for composing *parameterized* EDA operations, in which the agent first chooses the operation type, then the adequate parameters. Each such operation takes some input parameters and operates, at time t, on the current display d_{t-1} (i.e., the results screen of the last performed operation at $t-1$). It then outputs a corresponding new results display d_t. Since ATENA is primarily a proof of concept, we use a limited set of analysis operations, to be extended in future work. The following EDA operations are included:

1. FILTER($attr, op, term$) is used to select data tuples that match a criteria. It takes an attribute, a comparison operator (e.g. $=, \geq, contains$) and a numerical/textual term, and results in a new display representing the corresponding data subset.
2. GROUP($g\_attr, agg\_func, agg\_attr$) groups and aggregates the data. It takes a single[1] attribute to be grouped by, an aggregation function (e.g. SUM, MAX, COUNT, AVG) and another attribute to employ the aggregation function on.
3. BACK() allows the agent to backtrack to the previous display (i.e, the results display of the action performed at $t-1$) in order to take an alternative exploration path.

Formally, the action space is defined as follows. Let OP be the set of *action types* (in our case, $OP = \{\texttt{FILTER}, \texttt{GROUP}, \texttt{BACK}\}$). Each action type $o \in OP$ has a corresponding set of *parameters* $P^o = \{p_1^o, p_2^o, \ldots\}$, and each parameter p has a finite value domain $V(p)$. An action is a tuple (o, v) where $o \in OP$ and v is a valid assignment for the parameters P^o (i.e., in the Cartesian product $v \in V(P^o)$, $V(P^o) := \bigtimes_{p \in P^o} V(p)$). The entire action space is thus defined by $\mathcal{A} = \bigcup_{o \in OP} \bigcup_{v \in V(P^o)} \{(o, v)\}$ for any valid parameters' assignment v. In a similar manner, our model can be extended with other operation types such as projection, visualization, join, etc.

The advantages of our action space are: (1) the actions are atomic and relatively easy to compose (e.g., there are no syntax difficulties). (2) complex displays are formed *gradually* (e.g., first employ a FILTER operation, then a GROUP by some column, then aggregate by another, etc.), as opposed, e.g., to SQL queries where the entire query is composed "at once". The latter allows fine-grained control over the agent, as each atomic action obtains its own *reward* (see Section 4.2).

State Observation Vectors. The second part of the MDP model is defining what information should be provided to the agent when reaching a new state in the ongoing episode.

[1]Complex group-by operations, comprising multiple attributes, are possible by employing consecutive group-by operations.

We use a simple vector representation based on the extraction of key descriptive features from the previous results displays obtained thus far. The current results display d_t is encoded to a numeric vector, denoted $\vec{d_t}$, that represents a compact, structural summary of d_t with the following features: (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 order to also record the broader context in which the current display was generated, we concatenate to $\vec{d_t}$ the display vectors of three most recent operations in the session, i.e., the final observation vector comprises $\vec{d}_{t-1}$ and $\vec{d}_{t-2}$, in concatenation with $\vec{d_t}$ (if d_{t-1} and d_{t-2} do not exist, a vector of zeros is provided instead).

4.2 Reward Signal

As discussed in Section 2, numerous data-driven systems (e.g. [12, 21, 37, 45]) use interestingness measures to assess the utility of analytical operations and present users with the ones obtaining top scores. In ATENA we also employ measures of interestingness to evaluate EDA operations performed by the agent, yet we use two additional signals, *diversity* and *coherency* in order to ensure that the *entire sequence* of operations is compelling and easy to follow.

The reward signal for a single EDA operation is a weighted sum of the following components:

(1) Interestingness Reward. In our system we implemented two different interestingness signals, one for group-by operations, and one for filter operations.

Interestingness reward for group-by operations: Our measure follows similar lines of *conciseness* measures [9, 17] which consider *compact* group-by results that covers many tuples as both informative and easy to understand. Our measure takes into account the number of groups, the number of attributes that are currently grouped-by, and the number of the underlying tuples, denoted g, a, and r (respectively). The reward signal is given by $\frac{h_1(g \cdot a)}{h_2(r)}$ where $h_1(\cdot)$ and $h_2(\cdot)$ are normalized sigmoid functions (see [26]) with a predefined width and center.

Interestingness reward for filter operations: To reward filter operations we follow common measures in the literature that assess the *exceptionality* of the filtered tuples, compared to a reference, larger set. Following [37, 44, 45], our interestingness reward favors filter operations whose result display d_t deviates significantly from the previous display d_{t-1}. To quantify such deviation, we use the Kullback-Leibler (KL) divergence measure, that determines how different one probability distribution is from another. In case d_t is not grouped, we define the value probability distribution P_A^t of an attribute $A \in Attr$ to be the relative frequency of its values (i.e., for each value v of attribute A in $d_t, p(v)$ is the probability to randomly choose v). The interestingness reward is then defined by: $h(\max_{A \in Attr} D_{KL}(P_A^{t-1}, P_A^t))$, where the sigmoid $h(\cdot)$ is used to obtain a more significant difference in values. In case d_t is grouped, the KL divergence is compared only w.r.t. currently aggregated attributes (rather than on all attributes, as above).

(2) Diversity Reward. We want to encourage the agent to choose actions that induce new observations and show different parts of the data than those examined thus far. This is done by further utilizing the numeric vector representation of each results display ($\vec{disp}$), namely, by calculating the minimal Euclidean distance between the observation vector $\vec{d_t}$ and the vectors of all previous displays obtained at time $< t$, i.e., $\min_{0 \leq t' < t} \delta(\vec{d_t}, \vec{d_{t'}})$.

(3) Coherency Reward. We rely on an external classifier in order to determine whether a given EDA operation is coherent at a certain point in the notebook. Our classifier is based on weak-supervision, using Snorkel [35] to construct a classification model from a set of heuristic classification rules. This solution allows us (1) to overcome the lack of training data containing annotated EDA operations, and (2) to easily fine-tune the classifier, if desired, w.r.t. the specific schema and the users' given set of focal attributes.

Each classification rule in our classifier takes as input a subsequence of EDA operations $q_1, q_2, \ldots q_t$ and their result displays, and outputs whether operation q_t is *coherent* in the context of the previous operations. We use two types of rules: *(i) general rules* – considering general properties of the operations sequence. For example: *"a group-by employed on more than four attributes is incoherent"*, and *a group-by on a continuous, numerical attribute is incoherent.* Such rules are applicable on all input datasets.

(ii) data-dependent rules – More rules can be (optionally) written particularly for the input dataset's semantics and w.r.t. users' predefined focal attributes. For example, in the flights dataset (as described in Example 1.1), aggregating on the column "flight-number" is largely incoherent, whereas, if the user focuses on flight delays, aggregating on the columns "departure-delay time" is preferred.

Snorkel is then employed over the set of classification rules to construct a generative model for predicting if a given EDA operation is coherent or not. The final coherency signal is calculated using the confidence level of the generative model (in $[0, 1]$). In Section 6.1 we report the types of rules used in our experiments.

5 DRL AGENT ARCHITECTURE

We first explain the difficulties in optimizing our MDP model for EDA, and why DRL is a sensible approach to solve it. We then discuss what challenges remain to be overcome, and present our novel DRL architecture for the problem.

Why use DRL for the EDA control problem? Optimizing our EDA model is challenging because of two main reasons: (1) the MDP model is exponentially large (w.r.t. the input dataset) and high-dimensional, as the number of states corresponds to the number of all intermediate results of any possible exploratory operation on the input dataset. Therefore the model cannot be fully materialized. (2) The reward signal is compound, non-differentiable, and is *cumulated* over N consecutive steps. Such settings make *analytical* optimizations (e.g.,linear programming, policy iterations) difficult to employ [18], as well as classic reinforcement-learning solutions [25]. In contrast, DRL has been shown to be extremely useful for solving high-dimensional, complex control problems previously thought to be intractable [25].

However, off-the-shelf DRL solutions are inefficient for the EDA problem, since in our MDP model, not only is the number of states large, but also the action-space is parameterized, very large, and discrete. Even in our prototype environment, which only supports filter, group-by and aggregate operations, the number of possible unique operations at each point exceeds $100K$. In comparison, the DRL environment of the very challenging game of Go allows for about 250 different discrete legal moves in every turn [40]. This setting is problematic for existing DRL architectures, since, (1) each distinct possible action is represented as a dedicated node in the output layer (see, e.g. [13, 25]), and (2) the large action space makes the already daunting *exploration/exploitation* problem even a bigger issue. As we empirically show in Section 6.4, when using existing DRL architectures, the agent's learning process to converge very slowly to a local maximum, far from the optimal.

Solution Overview. To that end, we use a novel solution that both decreases the size of the network and facilitates an efficient exploration/exploitation policy. Our solution can be easily injected into off-the-shelf DRL architectures and algorithms (e.g. DQN, Advantage Actor-Critic, etc.). It comprises three parts: (1) A "twofold output" layer architecture, that effectively utilize the parametric nature of the EDA action space, and allows the agent to choose an EDA operation type and a value for each parameter.(2) A dedicated "binning" method that further reduces the vast value-domain of the filter *term* parameter, which includes all available dataset tokens, and (3) an exploitation/exploration policy that leverages the gained experience not only to select the right action

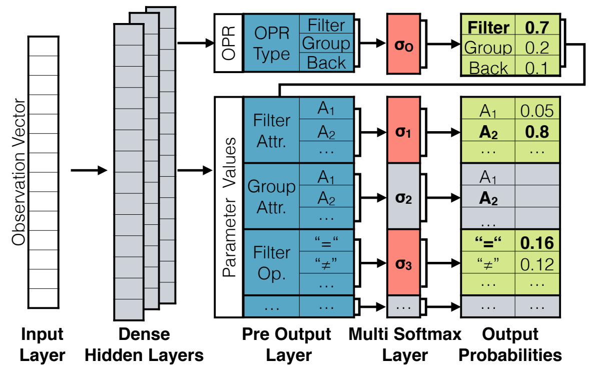


Figure 3: Actor Network Architecture

to employ at each state, but also for performing smart exploration choices. We explain each of these components next.

Twofold Output Layer Architecture. An illustration of our network architecture is provided in Figure 3. First, the observation vector goes through several dense hidden layers with a ReLU activation function (this is common practice in DRL architectures [19]). Then, instead of the standard, very large softmax output layer, we use two new layers that we design, to significantly reduce the network's size:
(1) The Pre-Output Layer. This layer (colored blue in Figure 3) contains a node for each EDA operation type, and a node for each of the parameters' values. Each node is connected to the previous hidden layer. The size of the Pre-Output Layer is equal to the size of the parameters' value domains and number of operation types, i.e., $|OP| + \sum_{o \in OP} \left|\bigcup_{p \in P_o} V(p)\right|$, which is significantly smaller than the standard output layer that contains a node for each distinct action, with a size of $\sum_{o \in OP} \left|\bigtimes_{p \in P_o} V(p)\right|$.
(2) The Multi-Softmax Layer. Often in DRL architecture a softmax calculation is applied on the output of the last hidden layer, in order to produce a probability value for each distinct action (that sums to 1). When the action space is large, not only is the computation time high, also the learning process is much slower and ineffective (as we show in Section 6.4). We therefore use a novel, Multi-Softmax layer, in which the softmax computation is segmented and performed for each operation type and parameter individually. Namely, Softmax Segment σ_{OP} is linked only to the Pre-Output nodes that correspond to operation types, then a separate Segment σ_p is defined for each parameter p, linked only to Pre-Output nodes corresponding to values in $V(p)$.

The Multi-Softmax Layer works as follows (See illustration in Figure 3). First, Segment σ_o is used to generate a probability distribution over the operation types OP, from which the chosen operation type $o \in OP$ is sampled (e.g., in the example workflow depicted in Figure 3 see that the chosen operation, obtaining the highest probability, is 'filter'). Next, the parameters of o are instantiated by activating only the corresponding segments σ_p $\forall p \in P^o$ (See the red colored segments of the Multi-Softmax layer in Figure 3).

Efficient selection of dataset values for the 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 the Pre-Output Layer, we use a simple yet effective binning solution that maps the individual tokens (i.e., dataset values or parts thereof) to a fixed size array of B bins, according to the *frequency of appearances* of each token in the current display. Then, instead of choosing a particular token, the agent chooses a frequency range $b_{[i,j]}$. In turn, an actual token whose frequency of appearance is within that range, is sampled uniformly at random. Following a common assumption that the frequency of tokens is exponentially distributed (according to, e.g., Zipf's Law), we particularly use logarithmic binning [31] (However, our solution is modular and any alternative binning division may be used).

Exploration/exploitation policy. Our architecture facilitates an effective exploration policy, based on Boltzmann exploration [22], in which, originally, an action is sampled according to the output probability distribution generated by the softmax layer. Using the two-fold output layer allows the agent to make an independent exploration/exploitation decision for each operation type and parameter, since each parameter has a dedicated softmax layer. This allows the agent to *exploit* its experience, when available, for some parameters, and *explores* the values of other parameters.

Also, to further encourage exploration, we use *entropy regularization* [30], which prevents the agent from prematurely converging to a local optimum. With entropy regularization the agent receives a reward bonus proportional to the entropy of the policy, i.e., the probability of choosing a distinct EDA operation (computed over the joint probability distribution of the Multi-Softmax Layer). The higher the entropy of the actions' probability distribution, the higher the bonus.

6 EXPERIMENTAL RESULTS

Our experiments are meant to answer three main questions:
(1) Can users derive actual insights from passively examining our auto-generated EDA notebooks?
(2) How good are the notebooks generated by ATENA compared to notebooks generated by other means?
(3) Is DRL, and specifically our architecture, necessary for generating high-quality EDA notebooks?

As common for generative models (e.g., image caption [46], machine translation [33]), we conduct two complementary quality evaluation experiments: Human evaluation (Section 6.2), in which participants manually reviewed the generated EDA notebooks, and an *automatic benchmark for EDA notebooks* (Section 6.3), which compares the generated notebooks to a set of gold-standard ones. The benchmark is fully open-source, available in [5].

Last, we performed additional experiments (Section 6.4), comparing the convergence of ATENA to alternative optimization architectures.

6.1 Experimental Setup

We next describe the experimental datasets and baselines, and provide implementation details.

Datasets. We use two collections of input datasets, each with a different schema and application domain, *cyber-security* and *flight-delays*. We chose these collections because they provide useful means to compare and evaluate the auto-generated notebooks. We next explain the characteristics of each collection and the purpose it serves in our experiments.
Cyber-Security Datasets. This collection comprises of 4 (completely unrelated) datasets, as detailed in Table 1. The datasets originate from 4 different cyber analytic challenges [43], in which participants are required to explore each dataset to reveal a specific underlying cyber attack conveyed in the dataset (See Table 1). We use this collection in our experiment for three reasons:

(1) Each dataset is provided with an official solution that contains all of the relevant insights regarding the underlying attack. We use these lists, which contain between 9 to 15 insights, to quantify the number of relevant insights users successfully derived from examining the EDA notebooks generated for the experimental evaluation (See Section 6.2).

(2) Each dataset has several "walk-through" documents manually written by expert cyber-security analysts, to demonstrate, step by step, the EDA operations they used in order to complete the challenge. We use these documents to generate "gold-standard" EDA notebooks, to serve as an upper bound in our quality evaluation experiments.

(3) Last, this collection is accompanied by traces of real-life EDA sessions made by experienced analysts exploring the datasets (publicly available in [42]). The EDA traces were recorded and collected by the authors of [29] for evaluating their proposed log-based EDA recommender system (recall from Section 2). We use the recorded traces (all performed with the same purpose of reveling the underlying cyber-attack hidden in each dataset), to generate corresponding EDA notebooks (which replay these sessions), and compare them to those generated by ATENA, as described below.

It is important to stress the difference between (2) and (3): The "gold-standard" notebooks, which originated from the walk-through tutorials, were manually created by experts, after completing their EDA process, with the goal of guiding readers through the dataset and allowing them to derive important highlights. In contrast, the notebooks generated from EDA traces simply capture the analysts' EDA process, and may not be comprehensible to other users.

Dataset	Size (rows)	Description
Cyber #1	8648	ICMP scan on IP range
Cyber #2	348	Remote code execution attack
Cyber #3	745	Web-based phishing attack
Cyber #4	13625	TCP port scan
Flights #1	5661	AA Flights on Sundays
Flights #2	8172	Flights departing from BOS
Flights #3	1082	From SFO to LAX
Flights #4	2175	Short, night-time flights

Table 1: Experimental Datasets

Flight-delays Datasets. To examine the output quality in another schema and application domain, we use an additional collection of datasets, derived from the popular Kaggle *2015 Flight Delays* [32] database. Each dataset contains records of past flights (with specific characteristics, as described in Table 1), with attributes such as origin/destination airport, flight duration, issuing airline, departure delay times, etc.

While these datasets are not provided with a list of relevant insights (as in the cyber-security collection), they all have a specific exploration goal – investigating flight delays. They also contain the following means to produce "gold-standard" and traces-based notebooks: (1) the Kaggle platform that hosts the database contains several EDA notebooks, manually created by fellow data scientists to demonstrate their EDA process in characterizing the flight delays. (2) To obtain EDA traces of real users exploring the datasets, we recruited 10 skilled analysts and asked them to perform EDA on the datasets, for the same exploration goal of investigating flight delays (while recording their traces).

Baselines. We compared the quality of the notebooks by ATENA to 4 different types of baseline notebooks.

First, the two different types of notebooks generated from human EDA processes: **1. "Gold-Standard": based on real EDA notebooks/tutorials.** As explained above, theses notebooks were created by expert users for the purpose of demonstrating their EDA process so that other users could passively examine and understand them, therefore forming a quality upper-bound to our auto-generated notebooks.

However, since the notebooks and tutorials contain textual captions as well EDA operations not yet supported in ATENA, we created equivalent EDA notebooks using the same filter, group-by and aggregate operations, in order to facilitate balanced quality comparison between all baselines.
2. "EDA-Traces": notebooks generated from logs of EDA sessions made by experienced analysts. For each recorded session we generated a corresponding EDA notebook. The traces largely contain the same EDA operations as supported

in ATENA, therefore little to no curation was required. Recall that all exploratory sessions, in each dataset collection were made in light of the same exploration goal: discovering the underlying attack (cyber-security datasets) and characterizing flight delays (flight-delays datasets).

We then used alternative auto-generated notebooks:
3. "Interestingness-Only": notebooks auto-generated based only on interestingness assessment. To examine the necessity in our compound reward signal we use baselines that optimize only on the interestingness of EDA operations, ignoring the diversity and coherency of the generated EDA operations. As mentioned above, this is a classic approach, commonly used in many data-driven assistance tools for EDA (e.g. [12, 21, 37]). We use two different methods to optimize on the interestingness of the entire sessions:**(3A) Interestingness-Only Greedy (Greedy-IO)** computes, at each step, the interestingness score (computed as described in Section 4.2) of all possible operations and greedily chooses (no machine learning), the operation obtaining maximal interestingness, and **(3B) Interestingness-Only ATENA (ATN-IO)** which uses our DRL architecture as described in Section 5 but with the goal of optimizing only on the interestingness signal.
4. "Alternative Optimization Architectures": notebooks auto-generated using different optimization architectures/techniques. To examine whether DRL and particularly our architecture described in Section 5 are necessary, we use three alternative optimization techniques for our compound reward signal (with all three components): First, we used two alternative DRL architectures to the one used in ATENA: **(4A) Off-the-Shelf DRL (OTS-DRL)** uses a standard DRL architecture with a softmax output layer containing a node for each distinct EDA operation[2]. Then to particularly examine the necessity of our Twofold Output Layer we also use **(4B) OTS-DRL With Binning (OTS-DRL-B)** which has the same standard DRL architecture as in (4A), but instead of employing explicit filter terms it uses the frequency-based binning solution, described in Section 5. Last, to examine whether DRL is indeed effective for our problem we used **(4A) Compound-Reward Greedy (Greedy-CR)** which does not uses DRL but a greedy, non-learned policy to select the operation inducing the highest reward.

Implementation. We implemented the EDA environment in Python 3, using the Pandas [27] library to perform EDA operations. As the overall reward is a weighted sum of three individual signals (recall from Section 4.2), we set the weight values to obtain learning balance between the reward components such that no component obtains below 10% of the total reward (However, different weight settings can be used to reflect different priorities of the reward components).

As for the coherency settings, for all the air-travel datasets, where the focus is on delays, we selected the attributes "departure_delay" and "arrival_delay" as focal attributes. For the cyber datasets, where the exploration goal is to reveal underlying network attacks, we set the focal attributes to be "source_ip" and "destination_ip".

The neural network agent uses the current DRL state-of-the-art Asynchronous Actor Critic (A3C) [30] enhanced with Proximal Policy Optimization (PPO) [38]. Our Twofold Output Layer is injected into the "Actor" network, replacing its softmax output layer. The algorithms and neural network are implemented in ChainerRL [34], a common DRL Python library. For training the agent network we used an Intel Xeon CPU based server with 24 cores and 96 GB of RAM.

6.2 Qualitative Human Evaluation

We performed a user study, with 40 volunteers, all computer science (BSc) graduates or graduate (MSc/PhD) students with some background in data analysis. We generated EDA notebooks using ATENA for each of the datasets listed in Table 1, as well as by a baseline from each type as described above.

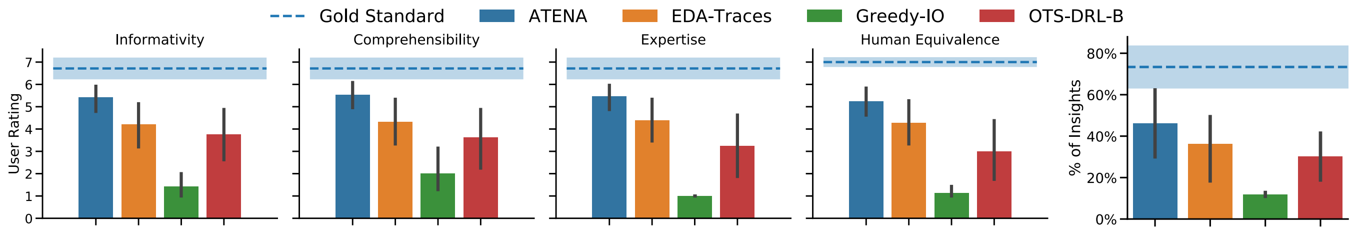
Each participant was presented with four EDA notebooks (each on a different dataset), one at a time, and asked to examine each notebook w.r.t. the same exploration goals as described above. After inspecting the notebook, we asked the participants to rate it according to several quality aspects, as well as to list the insights they gathered on the dataset just from examining the notebook.

We first discuss the overall rating for each type of EDA notebook, then compare the number of gathered insights.

Qualitative evaluation. After inspecting each notebook, users were asked to rate it, on a scale from 1 (lowest) to 7 (highest), according to the following criteria: **(1) Informativity** — How informative the notebook is and how well does it capture dataset highlights? **(2) Comprehensibility** — To what degree is the notebook comprehensible and easy to follow? **(3) Expertise** — What is the level of expertise of the notebook composer? **(4) Human Equivalence** — How closely the notebook resembles a human-generated session?

Figure 4a shows for each of the criteria above the average scores obtained by ATENA and the baseline notebooks: The human-generated baselines Gold-Standard and EDA traces, and a representative for each type of auto-generated notebooks – Greedy-IO and OTS-DRL-B (these baselines outperformed the other ones in their respective categories, as shown in Section 6.3). The scores differences between each two baselines were verified to be statistically significant (using a paired T-test), with p-values far below 0.00001.

[2]To allow convergence in a reasonable time, we restricted the number of filter terms to the ten most common tokens in each column (120 in total).



(a) User Ratings of Examined notebooks (on a scale of 1 to 7) (b) % of Gathered Insights

Figure 4: Qualitative Human Evaluation – Overall rating and insights gathered from viewing EDA notebooks

The gold-standard notebooks are marked by the dashed line (with an error band of ±1 standard deviation), obtaining, as expected, almost top scores (6.8/7 on average) in all criteria. As for the generated notebooks, first see that the alternative auto-generation approaches Greedy-IO and OTS-DRL-B obtain the lowest ranking, with an average score of 1.4/7 and 3.4/7. Then, see that notebooks generated from EDA traces, obtain an average rating of 4.3/7. The substantial difference in scores w.r.t. the gold-standard notebooks stems from the fact that the EDA-traces notebooks, although performed for the same EDA goal, were not generated for demonstrative purposes, and meant to be watched by other users. **In comparison, the notebooks of ATENA obtained an average score of** 5.4/7**, i.e., more than a full-grade better than the EDA-traces notebooks.** This happens since ATENA is specifically geared to generate notebooks that are coherent and easy to follow (using its compound reward signal).

Comparison of Gained insights. For each of the cyber-security datasets, we measured how many relevant insights were discovered by our users, just from viewing an EDA notebook, out of the total number of insights as described in the solution. (we disregarded "irrelevant" user insights that did not appear on the lists).

Figure 4b shows the (average) percentage of insights gathered by users from examining the notebooks generated by ATENA and the baselines as described above.

We can see that ATENA, in correspondence with the user ratings in all criteria, outperforms the alternative approaches. See that **just by passively examining the notebooks generated by ATENA, users successfully derived an average of 46% of the datasets' relevant insights.** However, see that there is still room for improvement, as the gold-standard notebooks, created by experts, allow users to gain a higher number of insights.

The variance in the number of gathered insights, observed for all baselines (including the gold-standard), stems from the challenges' varying difficulty level.

As a representative example, consider dataset "Cyber #1" which contains data regarding a complex network scan issued by an attacker (as listed in Table 1). Out of a total of 9 insights (such as the attacker IP address, victim's exposed addresses, network protocols used, etc.), most users discovered more than 5 insights by viewing the notebook generated by ATENA, which helped them to successfully detect the attacker IP address, the victim organization's IP range, and exposed addresses. This is potentially a great starting point to discover additional details about the attack, e.g., a smaller scale TCP scan performed by the attacker only on the victim's exposed IP addresses.

6.3 Automatic Benchmark

The user study clearly shows that ATENA significantly outperforms notebooks generated by other means. However, since user-studies are difficult to reproduce, we next describe a benchmark for auto-generated EDA notebooks (denoted A-EDA) that can be easily reproduced in other settings, hereby facilitating the comparison of future models and approaches.

A-EDA, similarly to benchmarks for other generative models (e.g., machine-translation [33] and auto-generated image captions [46]), assess the quality of a notebook based on its distance from a set of curated, ground-truth notebooks.

As ground-truth we use the gold-standard notebooks described above, which obtained close to perfect score in the human evaluation. For each dataset we used a set of $5-7$ gold-standard notebooks.We then used several metrics to evaluate the distance between the generated notebooks and the gold-standard ones, with different degrees of flexibility:
(1) Precision. This measure compares consider the EDA notebooks as sets of distinct views (ignoring their order), counting a "hit" if a view occurs in one of the gold-standard notebooks and a "miss" otherwise. It is calculated by $\frac{\text{hits}}{\text{hits+misses}}$.
(2) T-BLEU-1, (3) T-BLEU-2, (4) T-BLEU-3. These measures are based on the well known BLEU [33] score, used for comparing sentences in image captions and machine translations [7, 15] (in our case the "sentence" is the sequence of views in the notebook). T-BLEU is more strict than Precision, since it also considers the prevalence of each view in the gold-standard set, as well as their order, by comparing subsequences of size n (rather than single views). We use n between 1 to 3 for T-BLEU-1 through T-BLEU-3.
(5) EDA-Sim. Last, we use a dedicated distance metric devised in [29] to estimate the similarity for exploratory sessions (the source code is available in [42]). EDA-Sim also considers the order of views yet allows for a fine-grained

Baseline	Precision	T-BLEU-1	T-BLEU-2	T-BLEU-3	EDA-Sim
ATN-IO	0.10	0.10	0.05	0.03	0.22
Greedy-IO	0.12	0.11	0.07	0.04	0.23
OTS-DRL	0.26	0.16	0.12	0.06	0.23
Greedy-CR	0.27	0.21	0.16	0.07	0.23
OTS-DRL-B	0.33	0.24	0.21	0.16	0.27
EDA-Traces	0.45	0.30	0.27	0.22	0.40
ATENA	**0.45**	**0.45**	**0.41**	**0.31**	**0.46**

Table 2: Overall A-EDA Benchmark Results

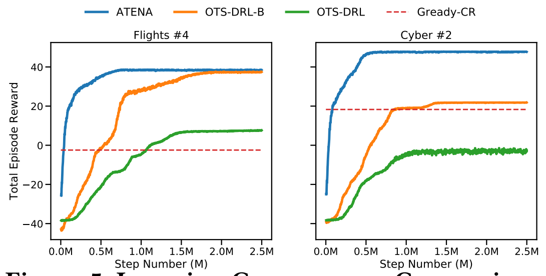


Figure 5: Learning Convergence Comparison

comparison of their content (i.e., almost identical views are considered "misses" in the above measures, yet EDA-Sim will evaluate them as highly similar, as described in [29]). For the final EDA-Sim score, we compare the generated notebook to each of the Gold-Standard notebooks and take the maximal EDA-Sim score obtained.

Results. Table 2 depicts the scores of all baselines, averaged across all 8 experimental datasets listed in Table 1. First, see that A-EDA scores closely correlate with the overall scores of the human evaluation and the gathered-insights comparison: All alternative auto-generation approaches obtain the lowest scores, outperformed by EDA-traces notebooks, which are then surpassed by ATENA w.r.t. each of the evaluation measures described above.

Next, examining the scores of all alternative auto-generation baselines, we can derive the following conclusions:
(1) Interestingness-Only baselines, which ignores the coherency and diversity of the EDA operations, obtain the lowest scores – whether using a simple greedy optimization (Greedy-IO) or when using DRL (ATN-IO). **This means that generating useful EDA notebooks requires a more elaborate reward signal than just the data-driven interestingness score**.
(2) While, indeed, Baselines 4A, 4B and 4C (which optimize on our compound reward signal) obtain better scores than the Interestingness-Only baselines – they are still significantly outperformed by ATENA. The off-the-shelf DRL architecture (OTS-DRL) is on par with the greedy-based optimization (Greedy-CR), but both are surpassed by OTS-DRL-B which uses our frequency-based binning solution. Yet all alternative optimizations are significantly outperformed by ATENA. **Hence, DRL, and particularly the novel architecture we use in ATENA, are highly effective for generating useful EDA notebooks**. We also derive an analogous conclusion from the learning-convergence perspective, as explained in Section 6.4 below.

6.4 Learning Convergence Comparison

Last, we compare the effectiveness of our solution with the performance obtained by alternative optimization architectures (baselines 4A-4C).

Figure 5 shows the mean episode reward (i.e., the cumulative, non-normalized reward) as a factor of the number of training steps obtained by ATENA, for two representative datasets (similar trends occur in the rest). First, as Greedy-CR uses a non-learning greedy policy, it is indifferent to the number of training steps, therefore depicted as the dashed horizontal line in each graph. See that Greedy-CR obtains a much lower reward than ATENA. Baseline OTS-DRL, which corresponds to standard DRL architectures with a softmax output layer, demonstrates inferior learning effectiveness, as it requires more than a million training steps to stabilize on a suboptimal reward (close to 0). The effectiveness of OTS-DRL-B, which uses the same architecture as OTS-DRL, refined with our frequency-based binning solution, is higher than of OTS-DRL, as it is able to converge, after more than a million steps (6-11 hours on our server), to a higher reward. Nevertheless, the full ATENA outperforms all three baselines – **it converges about 2-3X faster, to a significantly higher mean reward.** Furthermore, see that the convergence of ATENA is stable, i.e., reaching a high reward regardless of the explored dataset, while the performance of the other baselines varies.

7 CONCLUSION

We presented a system for auto-generating EDA notebooks. Our solution is based on a dedicated MDP model and a DRL architecture, used for generating notebooks that not only contain a set of interesting views, but, importantly [23], derive views that show diverse aspects of the dataset in a coherent narrative.Our experiments demonstrate the quality of the generated notebooks, and that users actually derive insights only from examining them.

Nevertheless, As the first of its kind, ATENA has several limitations that are left to be handled in future work, such as expanding its supported set of exploratory operations, facilitating the production of personalized sessions, and generalizing its learning process across datasets. We refer the reader to [6] for a discussion on what other components are required to produce customizable and more refined EDA sessions, in pursuit of a longer-run goal of reducing the manual effort in EDA, making it fully "hands-free".

Acknowledgments. This work has been partially funded by the Israel Innovation Authority (MDM), the Israel Science Foundation, and the Binational US-Israel Science Foundation.

REFERENCES

[1] Kaggle community. https://www.kaggle.com.

[2] Tableau software. https://tableau.com.

[3] D. Asimov. The grand tour: a tool for viewing multidimensional data. *SIAM journal on scientific and statistical computing*, 6(1):128–143, 1985.

[4] O. Bar El, T. Milo, and A. Somech. Atena: An autonomous system for data exploration based on deep reinforcement learning. In *CIKM*, 2019.

[5] O. Bar El, T. Milo, and A. Somech. A-eda: Automatic benchmark for auto-generated eda. https://github.com/TAU-DB/ATENA-A-EDA, 2020.

[6] O. Bar El, T. Milo, and A. Somech. Towards autonomous, hands-free data exploration. In *CIDR*, 2020.

[7] R. Bernardi, R. Cakici, D. Elliott, A. Erdem, E. Erdem, N. Ikizler-Cinbis, F. Keller, A. Muscat, and B. Plank. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *J. Artif. Int. Res.*, 55(1):409–442, Jan. 2016.

[8] M. Brachmann, C. Bautista, S. Castelo, S. Feng, J. Freire, B. Glavic, O. Kennedy, H. Müeller, R. Rampin, W. Spoth, et al. Data debugging and exploration with vizier. In *SIGMOD*, 2019.

[9] V. Chandola and V. Kumar. Summarization - compressing data into an informative representation. *KAIS*, 12(3), 2007.

[10] F. Chirigati, H. Doraiswamy, T. Damoulas, and J. Freire. Data polygamy: the many-many relationships among urban spatio-temporal data sets. In *SIGMOD*, 2016.

[11] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Aide: An active learning-based approach for interactive data exploration. *TKDE*, 2016.

[12] M. Drosou and E. Pitoura. Ymaldb: exploring relational databases via result-driven recommendations. *VLDBJ*, 22(6), 2013.

[13] 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.

[14] M. Eirinaki, S. Abraham, N. Polyzotis, and N. Shaikh. Querie: Collaborative database exploration. *TKDE*, 2014.

[15] EuroMatrix. Survey of machine translation evaluation, 2017. data retrieved from World Development Indicators, https://www.euromatrix.net/deliverables/Euromatrix_D1.3_Revised.pdf.

[16] J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on computers*, 100(9):881–890, 1974.

[17] L. Geng and H. J. Hamilton. Interestingness measures for data mining: A survey. *CSUR*, 2006.

[18] A. Gosavi. Reinforcement learning: A tutorial survey and recent advances. *INFORMS Journal on Computing*, 21(2):178–192, 2009.

[19] M. Hausknecht and P. Stone. Deep reinforcement learning in parameterized action space. *arXiv preprint arXiv:1511.04143*, 2015.

[20] E. Huang, L. Peng, L. D. Palma, A. Abdelkafi, A. Liu, and Y. Diao. Optimization for active learning-based interactive database exploration. *Proceedings of the VLDB Endowment*, 12(1):71–84, 2018.

[21] M. Joglekar, H. Garcia-Molina, and A. G. Parameswaran. Interactive data exploration with smart drill-down. In *ICDE*, 2016.

[22] L. P. Kaelbling, M. L. Littman, and A. P. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[23] M. B. Kery, M. Radensky, M. Arya, B. E. John, and B. A. Myers. The story in the notebook: Exploratory data science using a literate programming tool. In *CHI*, 2018.

[24] T. Kraska. Northstar: An interactive data science system. *PVLDB*, 11(12), 2018.

[25] Y. Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[26] L. Mason, J. Baxter, P. L. Bartlett, and M. R. Frean. Boosting algorithms as gradient descent. In *NeurIPS*, 2000.

[27] W. McKinney. Data structures for statistical computing in python. In S. van der Walt and J. Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[28] T. Milo and A. Somech. Deep reinforcement-learning framework for exploratory data analysis. In *aiDM*, 2018.

[29] T. Milo and A. Somech. Next-step suggestions for modern interactive data analysis platforms. In *KDD*, 2018.

[30] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.

[31] M. Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46(5):323–351, 2005.

[32] U. S. D. of Transportation. 2015 flight delays and cancellations. https://www.kaggle.com/usdot/flight-delays, 2015.

[33] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*. Association for Computational Linguistics, 2002.

[34] I. Preferred Networks. The deep reinforcement learning library chainerrl. https://github.com/chainer/chainerrl, 2017.

[35] 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.

[36] A. Rule, A. Tabard, and J. D. Hollan. Exploration and explanation in computational notebooks. In *CHI*, 2018.

[37] S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of olap data cubes. In *EDBT*, 1998.

[38] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[39] T. Siddiqui, A. Kim, J. Lee, K. Karahalios, and A. Parameswaran. Effortless data exploration with zenvisage: an expressive and interactive visual analytics system. *PVLDB*, 10(4), 2016.

[40] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[41] M. Singh, M. J. Cafarella, and H. Jagadish. Dbexplorer: Exploratory search in databases. *EDBT*, 2016.

[42] A. Somech and T. Milo. React: Interactive data analysis recommender system. https://github.com/TAU-DB/REACT-IDA-Recommendation-benchmark, 2018.

[43] L. Spitzner. The honeynet project: Trapping the hackers. *IEEE Security & Privacy*, 99(2), 2003.

[44] M. van Leeuwen. Maximal exceptions with minimal descriptions. *Data Mining and Knowledge Discovery*, 21(2):259–276, 2010.

[45] 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.

[46] R. Vedantam, C. Lawrence Zitnick, and D. Parikh. Cider: Consensus-based image description evaluation. In *CVPR*, 2015.

[47] Z. Zhao, L. De Stefani, E. Zgraggen, C. Binnig, E. Upfal, and T. Kraska. Controlling false discoveries during interactive data exploration. In *SIGMOD*. ACM, 2017.