Towards Autonomous, Hands-Free Data Exploration

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

ABSTRACT

Exploratory Data Analysis (EDA) is an important yet difficult task, currently performed by expert users, as it requires deep understanding of the data domain as well as profound analytical skills. In this work we make the case for the Hands-Free EDA (HFE) paradigm, in which the exploratory process is automatically conducted, requiring little or no human input - as in watching a "video" presenting selected highlights of the dataset.

To that end, we suggest an end-to-end visionary system architecture, coupled with a prototype implementation. Our preliminary experimental results demonstrate that HFE is achievable, and leads the way for improvement and optimization research.

1. INTRODUCTION

Exploratory Data Analysis (EDA) is a core task in the majority of data-driven processes. It is ubiquitously done by data scientists and analysts who perform "hands-on" interaction with a dataset, by iteratively applying analysis actions (e.g. filtering, aggregations, visualizations) and manually examining the results. This is primarily done to understand the nature of the data and extract knowledge from it, yet is also fundamental for particular data scientific tasks such as data wrangling and cleaning, feature selection and engineering, as well as for explaining predictive models.

However, 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 [15, 34, 41, 10, 23], suggesting solutions such as simplified, modern EDA interfaces for non-programmers (e.g., [23], Tableau[1]), explore-by-example systems [10, 19], and dedicated recommender-systems that assist users in formulating queries [15], exploratory operations [34] and in choosing data visualizations [41, 43] (Refer to Section 5 for a more elaborate discussion). While these works greatly facilitate the exploration process in terms of response times and ease-of-use, the user remains in the "driver's seat", having to decide which exploratory operation to employ. The latter means that the entry bar for EDA is still high, and the task, even with a simplified UI, still requires data-oriented skills and the undivided attention of the user.

In this work we bring to discussion the Hands-Free EDA (HFE) paradigm, in which the exploratory process is automatically built. Think of it as watching an interactive video that leads the user through the dataset's highlights, and is customized according to her preferences and commands. The HFE paradigm may greatly reduce the manual effort devoted to EDA, allowing the users to understand their datasets and reach insights quickly and effectively.

Example Use Case. Clarice, a business analyst working for Beta Airlines (a major airline in the US), is assigned to examine causes for flight delays in Q3, 2019. She loads the data to the HFE system, selects the columns "Is Delayed" and "Delay Time" as focal attributes, and asks the HFE system to explore the dataset. After a while, she receives a message that her session is ready. She puts the HFE system on full screen and begins watching as the system performs exploratory operations. First, the system performs a "group-by" operation, presenting the total number of delayed Beta flights for each *origin airport*. Clarice skims through the results, realizing that most delays originate from only four airports. The HFE system continues the exploration by performing a "filter" operation, showing Beta flights where the origin airport is KGL International Airport (which turns out to be the one with the highest number of delayed Beta flights). Next, the system presents a line chart visualization, plotting the number of delayed flights for each week between June and August 2019. Clarice notices that there are substantially more delays on weekends. Clarice then issues a voice command "Add the average flight delay to the current display". After complying, the HFE system continues its automatic exploration by employing an additional group-by on "Aircraft Model". Examining the results screen, Clarice understands that many delays occur when the aircraft model is "878". The automatic session continues, providing Clarice with additional, informative displays that allow her to understand the "storyline" within the data, and quickly gain some preliminary insights regarding what may affect flight delays.

We propose a visionary system architecture, showing how HFE, as illustrated in the example above, can be accom-

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

10th Annual Conference on Innovative Data Systems Research (CIDR '20) January 12-15, 2020, Amsterdam, Netherlands.

plished. At the core of our architecture is a Deep Reinforcement Learning (DRL) mechanism, enriched with complementary components from existing lines of research, e.g., interestingness evaluation, EDA recommender systems, weak supervision, and representation learning.

In what follows, we present the HFE paradigm and its key challenges in Section 2. We detail the system architecture in Section 3, and present our prototype implementation and early experiments (Section 4). We overview related work in Section 5, then conclude and give future remarks in Section 6.

2. THE HANDS-FREE EDA PROBLEM

We first present the common, human-guided EDA process, then discuss the HFE paradigm and its key challenges.

The EDA Process. A (human-guided) EDA process begins when a user loads a particular dataset to an analysis UI. The user explores a dataset $\mathcal{D}$ by executing a series of analysis operations $q_1, q_2, ..q_n$ (e.g. filter, group-by, OLAP drill-down/rollup, and visualizations, depending on the particular analysis interface). The operations are executed in an interactive manner: each operation q_i generates a results *display*, denoted d_i. After examining the results display d_i the user decides if and which operation to perform next.

Typically, the goal of the EDA process is to obtain a general understanding of the content of the examined dataset, its characteristics, as well as to discover interesting aspects of it. However, users may have different prior knowledge, and points of interest. For instance, even on the same dataset of flights (as presented in the motivational example), an exploratory session of an analyst interested in *flight delays*, may be different than the one made by a different analyst interested in *sales and revenue*. Also, as the exploratory session progresses, different users often gain different insights, therefore navigating the continuation of their session to expose different aspects of the data. In other words, even on the same dataset, there is no singular "recipe" for a useful EDA process.

Autonomous, Hands-Free EDA. The ultimate goal of HFE, is to be able to automatically explore any given dataset, and gradually build a "storyline" for the particular data and user. The generated exploratory session not only should reveal highlighting, interesting aspects of the data, but it should also do it in a coherent, understandable manner, allowing the user to quickly gain insights on the examined data. Last, while autonomously exploring the data, the HFE system should be, at the same time, adaptive and responsive to the users' preferences and commands.

As is often the case with autonomous systems, accomplishing HFE requires formulating (and efficiently solving) a *control* problem. Namely, given some *controller* that allows the system to perform actions, the goal is to perform actions while optimizing an objective function (e.g., in the "driverless car" settings - navigate from A to B while minimizing transport costs and avoiding collisions). In our case, given a dataset and a controller that allows the system to perform exploratory operations, the goal of the HFE system will be to perform a sequence of *exploratory operations* that maximizes an EDA objective notion.

We next overview the key challenges in designing such a system:

(1) Designing an AI-enabled EDA interface: How to devise machine-readable encoding for EDA operations and their result displays? The first challenge is facilitating EDA for an AI-based autonomous system, namely, defining (a) what kind of operations it can employ and (b) what information it receives about its current state. For instance, should we support an expressive, flexible interface such as free-form SQL, or a more restricted set of primitive operators? Also, EDA operations may have compound result sets, with values from different types and semantic fields, as well as additional "layers" such as grouping and aggregations. How can this information be captured in a compact, machine readable way? The goal is thus to devise a numeric vector representation of EDA operations and result displays that supports effective decision making for the autonomous system.

(2) Defining the EDA objective function: How to evaluate EDA sessions? Autonomous systems require a computable objective function in order to correctly make decisions. However, to our knowledge, there is no such explicit objective function for EDA operations, let alone entire EDA sessions.

Ideally, we want the HFE system to generate exploratory sessions that (i) highlight *interesting* aspect of the data, (ii) are *coherent*, i.e., understandable and easy to follow, and (iii) contain *diverse* displays that cover multiple facets of the data.

However, implementing such an objective function is a challenge. As for interestingness, while a multitude of measures have been devised to capture the interestingness of data analysis/mining operations (e.g. [17]), each measure often captures a different facet of the elusive concept. Particularly in EDA, as we showed in [31], the notion of interestingness (and correspondingly, the measure used to capture it) changes dynamically even in the same EDA session. Deciding which measure to use, at each point of the session, is a challenge. As for coherency/understandability of EDA actions, we are not aware of any previous work dealing with this issue.

(3) Effective decision making: Handling a vast number of distinct exploratory operations requires more than a good planning algorithm. In many existing autonomous system settings the number of actions that the system can perform at a given moment is rather low (e.g., playing the game of Go requires choosing between 100 legal moves in every turn). However, this is not the case with autonomous data exploration. Even in our prototype implementation (See Section 4), which only supports filter, group-by and aggregate operations, the number of possible unique operations at each point exceeds $100K$, due to the parameters' value domain size (for example, choosing a *token* to filter by). This poses a substantial difficulty in employing existing planning/learning solutions, as well as aggravates the *exploration/exploitation* problem.

(4) Providing a personalized exploration experience: How to adjust the exploratory session according to users' preferences? Last, the autonomous system should be reactive and responsive to the user's input preferences and commands. It is thus challenging to formulate a personalized objective according to user preferences, as well as to facilitate the envisioned voice commands, after which the system should readjust its exploration path.

3. SYSTEM ARCHITECTURE

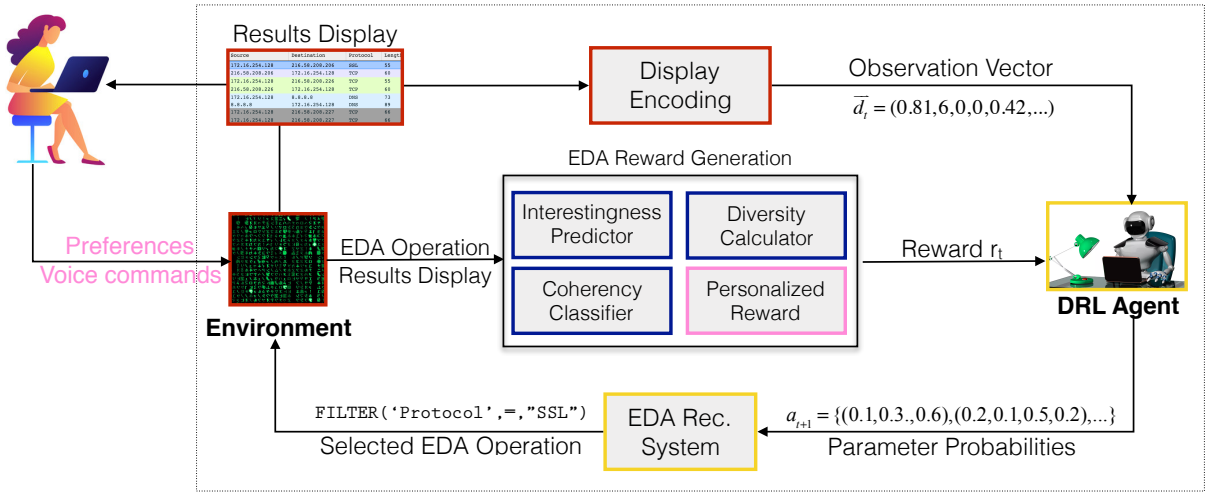

Figure 1: HFE System Architecture

From the multitude of algorithms and techniques for solving *control problems*, we decided to base the core of our system on Deep Reinforcement Learning (DRL). DRL is a highly promising paradigm that utilizes deep neural network models in classic (and novel) reinforcement learning algorithms. In recent years, it is being used at the core of many artificial intelligence-based systems, and shown to surpass human capabilities in a growing number of complex tasks, such as playing sophisticated board games, robotics, traffic light control, and more [27].

In a nutshell, DRL is concerned with a deep neural network *agent* interacting with an *environment*. The process is often modeled as a Markov Decision Process (MDP), in which the agent transits between *states* by performing *actions* from a predefined *action space*. At each step, the agent obtains an *observation-vector* from the environment, containing numeric information 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. The goal of the DRL agent is to learn how to obtain a high cumulative reward (over the course of its actions).

System Overview. An illustration of our DRL-based HFE architecture is provided in Figure 1. Its main components are: (1) The *EDA Environment* (which includes the boxes marked in red in Figure 1) is the interface that allows the DRL agent to employ operations and encodes the results to a machine-readable vector. (2) The *Reward Signal* (blue boxes) is used to evaluate the agent's operations in terms of interestingness, coherency, and diversity. (3) The *DRL Agent* (yellow boxes) composes EDA operations and incorporates an EDA recommender system to reduce the exploration space. Last, (4) *Session Customization* (pink box) is obtained by generating a personalized reward signal based on user preferences and commands.

Sections 3.1-3.4 detail the system architecture components (1)-(4) listed above.

3.1 AI-Enabled EDA Environment: What does the agent "see" and what can it "do"?

As mentioned, the first challenge is designing an AI-enabled EDA interface, namely the EDA *environment*. The crux of environment design, from our perspective, is twofold: (1) How to represent and control what the agent can "do"? (2) How to properly encode what the agent is "seeing"? i.e., how to devise a machine-readable representation of result displays.

EDA Action Space. Traditional EDA interfaces often use query languages (SQL, MDX, DSL, etc.), assuming the query composer is human. However, artificially generating queries is a known difficult problem, currently in the spotlight of active research areas such as *question answering* and *natural language database-interfaces* [26]. Such solutions attempt to translate a single NL request into a structured query, whereas we are interested in allowing an agent to compose a *sequence* of EDA operations from scratch.

Therefore, to facilitate autonomous composition as well to support a variety of EDA operation types (e.g., filter, group, OLAP/OLTP, visualizations, mining, etc.,) we suggest using *parameterized* representation of analysis operations for the EDA action space, that allow the agent to first choose the operation type, then the adequate parameters, such as the a particular aggregation function, kind of plot, filter term, etc. (similar representation is used in modern EDA systems such as Tableau's VisQL.)

Each such operation takes some input parameters (e.g., filter condition, aggregation function, type of chart, etc.) and a previous display d (i.e., the results of the previous operation), and outputs a corresponding new results display. The advantages of this EDA operations representation are that (1) actions are atomic and relatively easy to compose (e.g., there are no syntax difficulties). (2) queries are formed *gradually* (e.g., first employ a `FILTER` operation, then a `GROUP` by some column, then aggregate by another, etc.), as opposed to SQL queries where the entire query is composed "at once". The latter allows fine-grained control over the system's output, since each atomic action obtains its own *reward* (as explained below).

EDA Observation-Space. The agent decides which action to perform next mostly based on the *observation-vector* it obtains from the environment at each state. Intuitively, the observation should primarily represent the results display of the last EDA operation performed by the agent. However, (1) result displays are often compound, containing both textual and numerical data which may also be grouped, aggregated, or visualized and (2) other session information (e.g., the results of previous operations, data tuples covered thus far) may also be required for the agent to properly de-

cide on the next action.

The main challenges in designing the observation-vector are thus (i) to devise a uniform, machine-readable representation for result displays, and (ii) to identify what information is necessary for the agent to maintain stability and reach learning convergence.

We suggest two possible techniques, one that relies on extracting descriptive features, and the second on a more complex representation learning method. *(1) Descriptive Summaries.* Given a results display, we can extract a set of numeric features to form a compact, structural summary. Example features are the attributes value entropy and number of distinct values, and grouping information such as the mean and variance of the groups' sizes. To include session information, we suggest adding the encodings of the last n (configurable) result displays to the observation vector. *(2) Distributed representation learning.* Recent works [3, 14] demonstrate the benefits of using embedding technique for dataset tokens and tuples (in the spirit of word embedding techniques such as Word2Vec [30]), compared to manual feature extraction. The idea is to take a collection of datasets and employ a shallow neural network that learns a numeric vector representation for each token, s.t. the resulted vector convey contextual information (from other dataset tokens that frequently reside in the same tuple). Then, an entire dataset (or a subset thereof) can be represented by aggregating the vectors of its tokens (e.g., in NLP this is often done using weighted average [25]).

While the first approach is easier to implement, it mainly encodes structural and statistical features. Representation learning is more complex and requires further research, yet it holds the promise of also capturing the semantics conveyed in result displays.

3.2 The EDA Reward Signal: How to evaluate exploratory operations?

The EDA environment described above allows the DRL agent to interact with a dataset by employing EDA operations and obtaining machine-readable observations on their results. However, to effectively learn to compose meaningful operations, the agent also requires feedback (i.e., a positive/negative reward) for the operations it chooses.

In the absence of an explicit, known method for evaluating the quality of exploratory sessions, we propose a reward signal for EDA actions with three goals in mind: (1) Actions inducing *interesting* result sets should be encouraged. Also, (2) the actions should be *coherent*, i.e. understandable to humans and easy to follow. Last, (3) actions in the same session should yield *diverse* results describing different parts of the examined dataset (A fourth consideration is the relevance to the user needs and preferences, an issue that we defer to Section 3.4). The cumulative reward is defined as the *weighted sum* of these individual components.

Interestingness. As mentioned before, many measures are devised in previous work to capture the interestingness in individual data analysis or mining operations (such as dispersion, peculiarity, diversity, etc. [17]). While one may simply select a specific interestingness measure for each operation type, the problem is that different measures capture different facets of interestingness (even for the same operation type [17, 31]). Therefore, operations that are ranked as interesting by one measure, may be ranked as not interesting by another. To that end, we propose employing an external *interestingness measure predictor*, that, given a set of interestingness measures, predicts the most appropriate one to use at the current point of an ongoing exploratory session (In [31] we detail an example design and implementation for such a predictor). The agent then receives an interestingness reward on its chosen operation, according to the score it obtained from the particular measure selected by the interestingness measure predictor.

Coherency. Encouraging *coherent* EDA operations is rather unique in the field of RL and optimal control. For example, when playing a board game such as chess or Go, the artificial agent's objective is solely to win the game, rather than to perform moves that make sense to human players. Yet, in the case of EDA, the sequence of operations performed by the agent *must* be understandable to the user, and easy to follow. To that end, we suggest using an external classifier to evaluate the coherency of each EDA operation. However, since a relevant training dataset that contains annotated EDA operations does not exist, we propose employing a *weak-supervision* solution, namely, to build a set of heuristic classification-rules, e.g. *"a group-by employed on more than four attributes is non-coherent"* (enough such rules can be produced by several knowledgeable analysts in a few hours of work), then lift these rules into an efficient classifier by using systems such as Snorkel [35]. The coherency reward component can then be calculated based on the coherency prediction of the operation selected by the DRL agent.

Diversity. 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 can be done by further utilizing the numeric vector representation of the result displays, e.g. by aggregating the Euclidean distances of the current display vector and the vectors of the previous displays in the session thus far.

3.3 DRL Agent Design: Effectively learning how to choose exploratory actions

Typically in DRL, the agent neural network is composed of an input layer of the size of the observation space, several fully connected layers, and an output softmax layer of the size of the action-space. Action probabilities are then calculated from the network's output, and the one that obtained the highest probability is often the selected action of the agent. However, as opposed to most DRL settings, in our EDA environment the action-space is parameterized, very large, and discrete. Hence, two considerable challenges surface from the EDA problem setting: (1) directly employing off-the-shelf DRL architectures is extremely inefficient since each distinct possible action is represented as a dedicated node in the output layer (see, e.g. [13, 27]). (2) Due to the vast number of distinct EDA operations, the known *exploration/exploitation* trade-off (i.e., whether the agent should explore and try "new" operations rather than performing operations that it already knows) becomes a bigger issue, as the number of possible operations is very large. The latter may cause the agent's learning process to converge very slowly, as well as converging to some local maximum, far from the optimal. Our proposed architecture addresses these challenges as follows.

(1) Agent Network Architecture. To solve the challenge of the agent's network design we propose a flexible solution that can be easily injected to off-the-shelf DRL architectures and algorithms (the current state-of-the-art is

rapidly changing). Our solution hooks on the given architecture (e.g. DQN, Actor-Critic, etc.) with 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 action types and one for each parameter's 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 associated parameters.

In our prototype implementation (Section 4) we use this suggested network architecture (see Figure 2 for an illustration), and obtain a successful, converging learning process.

Nevertheless, other, more complex approaches may be explored in order to further improve the decision making process of the agent, such as: (1) Using a *hierarchical* network model [24], where one neural network learns to select an operation type, then a series of other networks are trained, one for each type of operation, to select adequate parameters. (2) Using representation learning for encoding EDA operations and parameters (as recently suggested in [29] for SQL operators), then utilizing a *continuous* action space. Both these directions require future research.

(2) Reducing the exploration space using an EDA recommender system. As mentioned above, the vast action space creates a large exploration space hence worsening the exploration/exploitation problem. To this end, previous work suggested incorporating external demonstrations and guidance when solving difficult exploration problems in DRL [38, 18]. Luckily, in the field of EDA, dedicated *recommender systems* (e.g., such as the one we proposed in [34, 32]) have been shown to effectively utilize a small collection of previous sessions (i.e., "demonstrations") in order to generate next-step recommendations. Such recommendations can be used to guide the DRL agent, and narrow its vast exploration space to a subspace that contains more promising EDA operations.

3.4 User Customization & Interactivity

An additional important feature in the HFE paradigm is the ability of the system to adjust itself according to user preferences as well as to receive and understand user commands in interaction time. We briefly discuss what is needed for this component and point to existing works that can be incorporated in its implementation.

User Preferences. The idea is to incorporate a new, *personalized* reward component (See the pink part in Figure 1), that will be formulated according to the user's preferences, such as focal attributes and data subsets, prior beliefs and hypotheses, etc. Implementing such a reward signal can be done using e.g., dedicated measures for *subjective interestingness*. Such measures consider prior information about the user and provide a personalized interestingness assessment, such as *surprisingness* and *actionability*, generated w.r.t. the data at hand and the user's prior beliefs. Also, it is possible to use harvest users' interestingness feedback and use learning-based solutions as suggested in [10] (using active learning), and [28] (using a learning-to-rank method).

Interactivity by Voice Commands. As the EDA session progresses, the user may want to give further specifications to the system or to redirect the exploration path elsewhere, e.g. "Go back to the previous display", or "Add the sum of transactions to the current display". Such voice commands

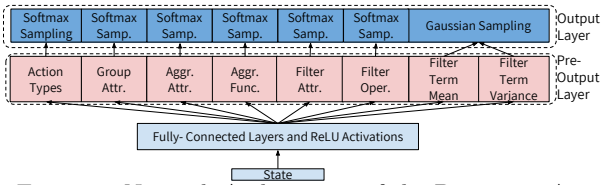

Figure 2: Network Architecture of the Prototype Agent

can be used to refine the EDA session and generate manual EDA operations, still "hands-free". After issuing such commands, the DRL agent readjusts its exploration path, as it decides on the next action based on the last observation that now also includes the external directions.

The implementation of this component can build on ideas from natural language interfaces for data access [26] and the conversational paradigm for data science, as introduced in [20].

4. PROOF OF CONCEPT IMPLEMENTATION

In light of the HFE architecture discussed in Section 3, we have built ATENA, a limited, proof-of-concept implementation (envisioned in [33] and demonstrated in [2]). ATENA employs a small set of EDA operations (filter, group-by and aggregation) to autonomously explore a given dataset. We next sketch the implementation details, and review some early experimental results, showing that ATENA's DRL agent converges to a high reward, and is able to conduct meaningful EDA sessions on small-medium datasets. Nevertheless, this is only a step on the way to fully achieve the HFE vision. In Section 6 we discuss the remaining components necessary to accomplish full fledged HFE.

Datasets. We use a dataset schema of network traffic logs containing 12 attributes, and 4 completely different data instances containing between 8K to 200K tuples. These datasets are publicly available[2], and were used in the experimental study of [34]. We particularly chose these datasets since they are accompanied by a collection of human EDA sessions. This allows us to evaluate and compare our auto-generated EDA sessions with the ones made by the human users, as described below.

EDA Environment. The ATENA environment allows the following operations:

`FILTER`$(attr, op, term)$ - used to select data tuples that match 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.

`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 in order to take an alternative exploration path.

As for the observation space, we used the descriptive summary observation vectors. Each *display vector* contains 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 (3)

[2]https://github.com/TAU-DB/REACT-IDA-Recommendation-benchmark

three global features storing the number of groups and the groups' size mean and variance. To provide context information, the *observation* vector is formed by concatenating the last 3 display vectors.

Reward Implementation. The reward signal implementation contains three components (as described in Section 3.2): interestingness, coherency, and diversity).

For simplicity, our restricted implementation does not include yet the measure prediction component (as described in Section 3.2), and uses, instead, a single measure per operation type: the *Compaction-Gain* [5] method is used to rank group-by actions (which favors group-by results in which a small number of groups covers a large number of tuples). To rank filter actions we use a relative, deviation-based measure (following [41]) that favors result sets that demonstrate significantly different trends compared to the entire dataset. The coherency reward relies on an implementation of the coherency classifier, using Snorkel [35] to create a labeled dataset from hand-crafted rules, and the diversity reward is implemented by calculating of the distances of the last observation vector and all the previous ones, as described in Section 3.2.

Agent Architecture. We used a current state-of-the art architecture: Advantage Actor Critic [27] with the Proximal Policy Optimization algorithm [36]. We modified the actor network by adding the "pre-output" layer as described in Section 3.3.

However, as the parameter *term* of the filter action can take any token that currently appear in the dataset, this still results in a very large output layer. To tackle this issue we used 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 2). 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.

Learning Convergence. Our DRL agent successfully obtains a high reward in the EDA environment. The learning curves (depicted in Figure 3), show that each reward component (and their weighted sum) steadily converges after about 0.5 million steps.

Early experimental results. To gauge the usability and quality of the EDA sessions generated with our prototype system, we conducted a small scale user study (10 student volunteers with some experience in EDA). Each participant was presented with two exploratory sessions on two different datasets, s.t. one was generated by ATENA, and the other generated by a human analyst. Without knowing their origin (i.e., ATENA/human) we asked the participants to watch each session via a user friendly interface. After watching each session, we asked the participants to provide a high-level description of the content of the dataset explored in the session, as well as to rank the quality of the session from several perspectives on a scale from 1 to 7. Aggregating the results for the human and auto-generated sessions, our findings are:

(1) 7/10 participants thought that the auto-generated sessions are **highly informative** (score $\geq$6), whereas only 4/10 ranked the human analysts' sessions as such.

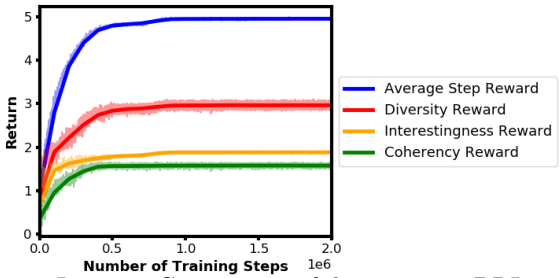

Figure 3: Learning Convergence of the prototype DRL agent

(2) 8/10 participants found the auto-generated sessions **easy to follow** (score $\geq$6), compared to 4/10 for the human analysts.

(3) 6/10 participants believed that the **auto generated sessions were made by an expert analyst, knowledgeable in the data domain**, whereas, surprisingly, none of the participants believed so regarding the human analysts' sessions. This is because the auto-generated sessions were easier to follow (due to the coherency reward), whereas the human analysts' sessions were conducted for personal use, rather than to serve as a demonstration.

5. RELATED WORK

Outside the scope of automating EDA, the database community has been making a considerable effort in facilitating the EDA process, along several important facets: First, simplified EDA interfaces (e.g., [23], Tableau, Pandas, Splunk) allow non-programmers to effectively explore datasets without knowing scripting languages or SQL. Second, numerous solutions (e.g., [6, 21]) were devised to improve the interactivity of EDA, by reducing the running times of exploratory operations, and displaying preliminary sketches of their results. Last, several more systems simplify the query formulation for non-expert users: Works e.g. [22, 16] suggest systems for SQL query auto-completion, and query-by-example systems (e.g. as in [8, 37]), allow non-technical users to specify their information needs by selecting a set of example tuples.

Closer to our work, another line of research has been dedicated particularly to automate the exploratory process, whether partially or fully. These works can be roughly divided into three categories:

1. EDA Recommender Systems. This special type of recommender system is dedicated to EDA, providing users with suggestions for specific, high-utility exploratory operations, or interesting views of the dataset. Within EDA recommender systems, we can differentiate between two major types of systems: (1) *data-driven* (also known as discovery-driven) systems, which use heuristic notions of *interestingness* and employ them, e.g., to find data subsets conveying interesting patterns ([11]), data visualizations [41], and data summaries [39]. (2) *Log-based* systems [15, 1, 44, 12] leverage a log of former exploratory operations, performed by the same or different users, in order to generate more personalized EDA recommendations. Last, hybrid methods such as [32, 34] allow effectively utilizing both the log and the dataset currently being explored.

Our envisioned HFE framework, as mentioned in Section 5, benefits from incorporating such systems: First, a data-driven, "interestingness" component is included in the HFE framework's reward signal (See Section 3.2). Second,

we rely on EDA recommender systems such as [34] to assist the construction of the automated session (as explained in Section 3.3) by reducing the exploration space to a subspace that contains more promising EDA operations.

2. Modeling users' interests, Explore-by-Example. As interestingness is often subjective [7] and dynamically changing, even in the same exploratory session [40], previous work attempted at modeling users' interestingness preferences. For example, [10, 19] suggest "explore-by-example" systems by taking an *active learning* approach, i.e., they harvest feedback on presented tuples ("interesting" or "not interesting") and use it to construct a model for an individual user's interest and gradually improve its accuracy as more feedback is collected. Another example is [28], in which the authors use a *learning-to-rank* model to assess the quality of data visualizations.

As mentioned above (see Section 3.4), we rely on such systems in order to construct a *personalized* reward component, and improve the interactivity of the system.

3. Automated Exploration. The idea of of fully automating EDA has been proposed in visionary short-papers e.g., [4, 42], however these works do not sketch a system architecture or present a prototype implementation. In [9], the authors describe a system for auto-generating data visualizations based on a supervised learning model of *sequence-to-sequence recurrent neural network*. In comparison, our envisioned system primarily relies on unsupervised, deep reinforcement learning, as high-quality labeled data may not always be available.

6. CONCLUSION

In this work we make the case for autonomous, hands-free EDA. With the exponential growth in data collected worldwide, and the corresponding demand for data oriented insights, we believe that a successful implementation of the HFE paradigm may greatly reduce the human effort devoted to EDA, hereby decreasing the need for highly-skilled data analysts.

Looking further ahead, while our PoC implementation demonstrates the potential of auto-generating meaningful and informative EDA sessions, the road to full-fledged HFE is still long. First, our prototype only partially covers the HFE architecture, using rather simple implementation for each component. Therefore, many of our architecture components can benefit from future research, most notably, the session customization and personalized reward. Also, establishing effective methods to reduce session generation time is a necessary future work, as well as devising more sophisticated methods for representing a larger number of EDA operations and their results.

Acknowledgments

This work has been partially funded by the Israel Innovation Authority - MDM, the Israel Science Foundation, Len Blavatnik and the Blavatnik Familiy, and Intel® AI DevCloud.

7. REFERENCES

[1] J. Aligon, E. Gallinucci, M. Golfarelli, P. Marcel, and S. Rizzi. A collaborative filtering approach for recommending olap sessions. *ICDSST*, 2015.

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

[3] R. Bordawekar and O. Shmueli. Exploiting latent information in relational databases via word embedding and application to degrees of disclosure. In *CIDR*, 2019.

[4] U. Cetintemel, M. Cherniack, J. DeBrabant, Y. Diao, K. Dimitriadou, A. Kalinin, O. Papaemmanouil, and S. B. Zdonik. Query steering for interactive data exploration. In *CIDR*, 2013.

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

[6] A. Crotty, A. Galakatos, E. Zgraggen, C. Binnig, and T. Kraska. The case for interactive data exploration accelerators (ideas). In *HILDA*. ACM, 2016.

[7] T. De Bie. Subjective interestingness in exploratory data mining. In *Advances in Intelligent Data Analysis XII*, pages 19–31. Springer, 2013.

[8] D. Deutch and A. Gilad. Qplain: Query by explanation. In *ICDE*. IEEE, 2016.

[9] V. Dibia and Ç. Demiralp. Data2vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks. *IEEE CG&A*, 39(5):33–46, 2019.

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

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

[12] K. Drushku, J. Aligon, N. Labroche, P. Marcel, and V. Peralta. Interest-based recommendations for business intelligence users. *Information Systems*, 86:79–93, 2019.

[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. Ebraheem, S. Thirumuruganathan, S. Joty, M. Ouzzani, and N. Tang. Distributed representations of tuples for entity resolution. *PVLDB*, 11(11), 2018.

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

[16] J. Fan, G. Li, and L. Zhou. Interactive sql query suggestion: Making databases user-friendly. In *ICDE*. IEEE, 2011.

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

[18] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, et al. Deep q-learning from demonstrations. In *AAAI*, 2018.

[19] E. Huang, L. Peng, L. D. Palma, A. Abdelkafi, A. Liu, and Y. Diao. Optimization for active learning-based interactive database exploration. *PVLDB*, 12(1), 2018.

[20] R. J. L. John, N. Potti, and J. M. Patel. Ava: From data to insights through conversations. In *CIDR*, 2017.

[21] N. Kamat, P. Jayachandran, K. Tunga, and A. Nandi. Distributed and interactive cube exploration. In *ICDE*. IEEE, 2014.

[22] N. Khoussainova, Y. Kwon, M. Balazinska, and D. Suciu. Snipsuggest: Context-aware autocompletion for sql. *PVLDB*, 4(1), 2010.

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

[24] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *NeurIPS*, 2016.

[25] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *ICML*, 2014.

[26] F. Li and H. Jagadish. Constructing an interactive natural language interface for relational databases. *PVLDB*, 8(1), 2014.

[27] Y. Li. Deep reinforcement learning: An overview. *arXiv*

preprint arXiv:1701.07274, 2017.

[28] Y. Luo, X. Qin, N. Tang, and G. Li. Deepeye: Towards automatic data visualization. ICDE, 2018.

[29] R. Marcus and O. Papaemmanouil. Flexible operator embeddings via deep learning. *arXiv preprint arXiv:1901.09090*, 2019.

[30] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, 2013.

[31] T. Milo, C. Ozeri, and A. Somech. Predicting "what is interesting" by mining interactive-data-analysis session logs. In *EDBT*, 2019.

[32] T. Milo and A. Somech. React: Context-sensitive recommendations for data analysis. In *SIGMOD*, 2016.

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

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

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

[37] T. Sellam and M. Kersten. Cluster-driven navigation of the query space. *TKDE*, 28(5), 2016.

[38] 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.

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

[40] A. Somech, T. Milo, and C. Ozeri. Predicting "what is interesting" by mining interactive-data-analysis session logs. In *EDBT*, 2019.

[41] 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.

[42] A. Wasay, M. Athanassoulis, and S. Idreos. Queriosity: Automated data exploration. In *2015 IEEE International Congress on Big Data*, pages 716–719. IEEE, 2015.

[43] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *TVCG*, 2016.

[44] X. Yang, C. M. Procopiuc, and D. Srivastava. Recommending join queries via query log analysis. In *ICDE*, 2009.