Guided Exploration of User Groups

Mariia Seleznova[1]*, Behrooz Omidvar-Tehrani[2], Sihem Amer-Yahia[3]†, Eric Simon[4]
[1]TU Berlin, [2]NAVER LABS Europe, [3]CNRS, University of Grenoble Alpes, [4]SAP Paris
[1]seleznova@tu-berlin.de, [2]behrooz.omidvar-tehrani@naverlabs.com,
[3]sihem.amer-yahia@univ-grenoble-alpes.fr, [4]eric.simon@sap.com

ABSTRACT

Finding a set of users of interest serves several applications in behavioral analytics. Often times, identifying users requires to explore the data and gradually choose potential targets. This is a special case of Exploratory Data Analysis (EDA), an iterative and tedious process. In this paper, we formalize and solve the problem of guided exploration of user groups whose purpose is to find target users. We model exploration as an iterative decision-making process, where an agent is shown a set of groups, chooses users from those groups, and selects the best action to move to the next step. To solve our problem, we apply reinforcement learning to discover an efficient exploration strategy from a simulated agent experience, and propose to use the learned strategy to recommend an exploration policy that can be applied to the same task for any dataset. Our framework accepts a wide class of exploration actions and does not need to gather exploration logs. Our experiments show that the agent naturally captures manual exploration by human analysts, and succeeds to learn an interpretable and transferable exploration policy.

PVLDB Reference Format:
Mariia Seleznova, Behrooz Omidvar-Tehrani, Sihem Amer-Yahia, Eric Simon. Guided Exploration of User Groups. *PVLDB*, 13(9): 1469-1482, 2020.
DOI: https://doi.org/10.14778/3397230.3397242

1. INTRODUCTION

User data is widely available in various domains and is characterized by a combination of demographics such as age and location, and actions such as rating a movie, providing advice on a product, or recording one's blood pressure [1]. Many companies address the very fast growing market of user data analysis by proposing dedicated platforms to collect and analyze such data in a variety of business segments, and start to tightly couple user data management with enterprise operational data management solutions [2, 3].

A common way of understanding user data is *user group analysis* whose purpose is to breakdown users into groups to gain a more focused understanding of their behavior or to identify a target group of users satisfying an information need. User group analysis has many applications in domains such as social sciences, product design, product marketing campaigns, and customer services [1]. In this paper, we provide examples on different user datasets: DBLP [4], a dataset of researchers, and MovieLens [5], a collaborative rating dataset. The first example task is about a PC chair who builds a program committee [6], starting with any seed group of researchers, and iteratively looking for users in groups that match properties expected of a PC (geographic distribution, gender and topic balance, etc). The other example task concerns about a movie festival committee seeking to form a diverse set of critics at the first screening of Drama and Comedy movies, by gathering a mix of reviewers from the two genres with different demographics [7].

Due to the iterative nature of the task at hand, user group analysis can be viewed as an instance of Exploratory Data Analysis (EDA). In this setting, the exploratory analysis of user groups is an iterative decision-making process, whereby an analyst is shown a set of user groups labeled with user and item attribute values (e.g., groups of researchers who published in VLDB, groups of movie critics in the 35-44 age range who review Drama movies), chooses target users from those groups, and selects the best exploration action to move to the next iteration (e.g., select a group whose label is [prolific, female, published in VLDB] that has researchers with well distributed geographical location, and apply an exploration action to return k diverse groups that overlap with the selected group). Despite a large body of work on the recommendation of data exploration actions, several shortcomings exist. First, most works recommend SQL/OLAP queries, which is not adapted to analysts with no IT expertise, while other works focus on specific types of actions that are too limited for user group analysis. Second, existing solutions either rely on tedious manual exploration [8], or on a log of exploration sessions on the same dataset [9]. This assumption is not valid in our case since many analysis tasks are performed on different datasets. Finally, recent work on automating the exploration process does not generate interpretable exploration policies [10, 11].

We propose the first framework for user group exploration that learns an exploration policy *without requiring prior collection of exploration logs.* The learned policy is used to rec-

*Work completed when author was an intern at SAP.

†Work funded by the Horizon 2020 research and innovation programme under grant agreement No 863410.

Proceedings of the VLDB Endowment, Vol. 13, No. 9
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3397230.3397242

ommend an end-to-end interpretable exploration session on a given dataset. Our framework supports exploration sessions consisting of a sequence of exploration actions of various types and returns recommendations for the exploration of new datasets, provided they are similar (with respect to domain-specific features) to previously analyzed datasets. Our framework is *independent* from the approach used to generate user groups from raw data. The exploration actions operate on groups that can be generated with any method ranging from SQL aggregate queries to k-means, graph-based algorithms (e.g., community detection), and graph representation learning, to name a few. In an offline phase, we learn an exploration policy from a simulated agent experience in such a way that no human intervention is needed. The lack of real exploration logs from a variety of users rules out the use of approaches that require large amounts of exploration traces such as sequence mining.

Our first contribution is to formalize the guided EDA problem applied to user groups as a Markov decision process where a state displays several groups, a transition is the application of an exploration action to a chosen group, and the reward of an action is a function of the number of target users discovered by that action. An exploration session is hence a sequence of exploration actions, and an exploration policy is a function that maps a state to an action. A common difficulty is the design of state features to capture the application of an exploration action to a group. To reflect a human agent, we propose semantics that unifies previously introduced actions (explore-around [8], explore-within [8], by-facet [12, 13], by-distribution [14], and by-topic [15]). State features are carefully designed to capture the effect of an action on a state. The problem of guided EDA on user groups becomes that of finding a policy that maximizes utility, i.e., that discovers as many target users as possible regardless of the dataset and the seed group.

Our second contribution is the use of Reinforcement Learning (RL) to learn an exploration policy from a simulated agent experience, i.e., an RL agent that acts as an analyst who chooses exploration actions, and recommends a policy. A notable advantage of RL methods is that, unlike supervised methods, they do not require to gather labeled data. Instead, the agent learns from rewards computed by the environment during the interaction [16]. This naturally fits our context since we can use, for instance, the PC of WebDB 2017 (or any previous PC of the same or different venue) to learn an exploration policy offline, and apply it online to build the PC of WebDB 2018. Additionally, the use of RL for EDA enables to model our problem as an interactive stochastic process, a nascent research area in databases, that can greatly benefit from research experience in data modeling [9]. Our work differs from recent proposals on RL-based EDA [9, 17]. First, we address a well-defined user data analysis task (gathering target users) and the exploration actions are chosen accordingly. Additionally, in our case, running a large number of manual explorations for each new dataset and new analysis task is impractical. Moreover, our focus on producing interpretable exploration sessions for a human analyst warrants the use of classical RL methods instead of deep RL or contextual bandits, a special case of RL where the agent's action does not determine the next state of the environment [10, 11, 18, 19, 20].

Our third contribution is an extensive set of experiments that validate the use of RL to solve the guided EDA problem for user groups. While we do not focus on theoretical convergence results, we empirically study various cases: users to be found are scattered in groups, exploration with different starting groups, and how our learned policies perform with respect to random ones. We also examine the utility of learned policies where they are learned on the same dataset or transferred between datasets. We also validate the appropriateness of the state features we crafted. Our experiments show that the simulated agent succeeds to learn an interpretable policy that naturally captures human analysts (as in [8]) and can hence be used to automate guided EDA.

The rest of the paper is organized as follows. First, we review related work in Section 2. We then introduce the user data model and define the guided EDA problem for user groups in Section 3. Then we present our solution based on reinforcement learning in Section 4. In Section 5, we provide an extensive set of experiments to validate the effectiveness of our approach for EDA. We conclude in Section 6.

2. RELATED WORK

Our work lies at the intersection of data management and machine learning. We present related work on user group exploration, recommendation of exploration actions, and reinforcement learning for interactive exploration.

2.1 User group exploration frameworks

User group exploration is a recent and dynamic research area in data management [1]. The traditional way of exploring user groups is based on previous frameworks [21, 22]. Most notably, `by-query` can be used to select user groups that satisfy predicates. However, to formulate a query, the analyst needs to know the database schema, the query language, and the underlying data distributions. We discard this type of exploration in our model because we want to target analysts that have no programming expertise.

Higher-level exploration actions not requiring much expertise have been introduced in recent work. by-facet exploration returns groups that have the same value for some attribute, e.g., gender [23, 12, 13]. `by-example` exploration [24, 8] finds groups that are similar to/different from an input group. In [8], `by-example-around` returns k diverse groups that overlap with an input group, and `by-example-within` returns a set of k subgroups that maximize the coverage of the input group. `by-analytics` selects groups based on some desired statistics. Some actions admit as input a set of distributions and find groups with similar rating distributions [14]. Others are a variant of `by-example` and look for groups similar to/different from a given group in terms of their distributions [25]. `by-text` exploration actions [15] leverage textual information such as tags and reviews to find groups that exhibit similar/dissimilar tags or reviews.

Our framework accommodates the above actions and is sufficiently generic to capture other actions. There exist other types of actions that we did not consider. For instance, `by-evolution` actions [26, 27, 28, 29] are designed for timestamped datasets. They can be used to select groups based on similarity/dissimilarity in their evolution over time, or those that exhibit some pattern.

2.2 Recommendation of exploration actions

A number of works recommend queries for the interactive exploration of databases [30, 31]. One approach based on collaborative filtering uses previously collected query logs of a dataset (SQL queries in [32, 33], and OLAP queries in [34, 35]) to recommend queries on the same dataset. This approach is not applicable to our problem because we cannot rely on a large number of previous exploration sessions for the same data analysis task, and we assume that the same analysis task can be performed on different datasets. Finally, these approaches focus on the recommendation of SQL/OLAP queries, while we are interested in more abstract operations that do not require technical expertise.

Another mode of interactive exploration, sometimes called data-driven approach [30, 36, 37, 38, 39, 40], recommends a single type of exploration action whose result is expected to optimize a measure of "interestingness" with respect to the current analysis context of a user on a given dataset. For instance, [38] suggests different drill-down operations on a given table (a case of `by-facet`), each producing a different set of tuples. With a data-driven approach, the notion of context is predefined (e.g., user profile [36], sequence of actions within the same session [38, 40], or data returned by a query [37, 39]). These works do not apply to our problem firstly because they only consider one type of exploration action. Secondly, the proposed measures of "interestingness" are predefined and cannot be adapted to different analysis tasks. The closest work to ours is the recommendation of various types of high-level exploration actions (e.g., filter, roll-up, cluster-by) for interactive data analysis over different datasets [9]. In the same spirit as the data-driven approach, an exploration action is recommended based on different user's analysis contexts modeled as trees in which previous actions are edges and their output dataset, called "display", are nodes. Given a context, the system looks for k similar contexts in a log of previous exploration sessions, and retrieves a candidate next action. A suggested action can be an *abstract action* (e.g., filter-by) that is generalized from a concrete action (e.g., filter-by protocol="SSL") by leaving out data-specific and context-specific attributes. This framework is however not suited for our problem. The proposed approach assumes the prior collection of many exploration logs for a given task. Additionally, in our case, the choice of a best next action is driven by the improvement of a utility function computed over all the states seen so far, rather than similarity with collected logs.

Another direction of research, called user intent identification, aims to understand how a user's exploration goal evolves during interaction. The approach developed in [17] considers prediction of "interestingness" measures relevant to the next step of interactive exploration. These measures characterize some properties of interactive exploration displays. Examples of such measures are: diversity (favors differences in values), dispersion (favors similar values), peculiarity (favors values different from average), conciseness (favors short displays). Other works propose predictive models to determine the topic of the next query [41] or the relevance of content items [42] based on the behavioral features of the analyst. Although the notion of interestingness proposed there is closer to our needs, these approaches are based on the clustering of real behavioral patterns using large collections of past user interactions.

3. USER DATA ANALYSIS MODEL

In this section, we define our EDA environment for user group analysis. Section 3.1 describes the user data model and the notion of user groups. Section 3.2 presents a unified semantics for the exploration actions used in our framework and defines the notions of relevance and quality of an action. Last, Section 3.3 defines our guided EDA problem which consists of recommending an exploration policy.

3.1 User datasets and user groups

We model user data as a set of users $\mathcal{U}$, a set of items $\mathcal{I}$, and a set $\mathcal{D} = \{\langle u, i, s, x\rangle\}$ where $u \in \mathcal{U}$, $i \in \mathcal{I}$, s is an integer score, and x is a text. A tuple $\langle u, i, s, x\rangle$ represents the action of user u (e.g., publishing, reviewing) on item i with an optional score s (e.g., recency of research publications, movie rating) and an optional text x (e.g., publication titles/abstracts, movie reviews).

Each user $u \in \mathcal{U}$ is described with a set of attribute-value pairs $u = \{\langle a, v\rangle\}$, where $a \in \mathcal{A}_U$ is a demographic attribute (e.g., age, gender, occupation), and v is a value in a's domain, i.e., $v \in dom(a)$. Similarly, each item $i \in \mathcal{I}$ is described with a set of attribute-value pairs $i = \{\langle a, v\rangle\}$ where $a \in \mathcal{A}_I$ is an item attribute (e.g., publication topic, movie genre). The set of all demographic and item attributes is denoted $\mathcal{A} = \mathcal{A}_U \cup \mathcal{A}_I$.

A user group g is a subset of $\mathcal{U}$ to which is associated attribute-value pairs $\langle a, v\rangle$ that define $label(g)$. The set of groups is denoted $\mathcal{G}$. $items(g)$ is the set of items i for which there exists a tuple $\langle u, i, s, x\rangle$ in $\mathcal{D}$ associated to a user u in g. Every user u in g must satisfy $label(g)$. For instance, $label(g) = \{\langle gender, female\rangle, \langle location, CA\rangle, \langle venue, VLDBJ\rangle, \langle venue, SIGMOD\rangle\}$ represents a group of female researchers in California who published a VLDB Journal and a SIGMOD paper at least once.

Since our focus is on exploration, our framework can rely on any group generation method, ranging from SQL aggregate queries to k-means, graph-based algorithms (e.g., community detection), graph representation learning, etc. Resulting groups may or may not overlap. For more details, the reader is referred to a survey on the stages of user group analytics [1].

Example 1. *As our running example, we consider Martin, a PC chair, who wants to build the WebDB 2014 PC by gathering geographically distributed male and female researchers with different topics of interest, seniority and expertise levels.*

3.2 Group exploration actions

The exploration actions to consider for our framework must be expressive enough to reflect how an analyst explores the space of groups to satisfy her needs in terms of reaching target users. Intuitively, an action should provide the ability to look into a group, expand it, or to compare group members. In [1], different group exploration actions are surveyed and categorized according to how they capture users' needs. In this paper, we cover those exploration actions and unify their semantics to make them composable.

We use E to denote a set of exploration actions. To represent the effect of an action, we define a generic function $explore(g, k, e)$ that takes as input a group $g \in \mathcal{G}$, an integer k, and an exploration action $e \in E$, and returns k other groups $\mathcal{G}_k \subseteq \mathcal{G} \setminus g$ that represent new exploration options.

We unify the semantics of the actions by defining two conditions on the set of k groups returned by *explore*(.) as follows (g_{in} is an input group and g a candidate group to be returned):

- $\forall g \in \mathcal{G}_k, relevance(g_{in}, g, e) \geq \sigma$;
- $quality(\mathcal{G}_k, e)$ is maximized.

Both functions return a value between 0 and 1. Their exact definitions depend on the exploration action e. Our design is inspired by the approach developed in [17] to predict "interestingness" measures that are relevant to the next exploration iteration: $relevance(g_{in}, g, e)$ forces the resulting groups $g \in \mathcal{G}_k$ to share similarities with the input group g_{in}, measured by functions such as Jaccard, Cosine, or Earth Mover's Distance. This ensures continuity in the exploration. Maximizing quality imposes that $\mathcal{G}_k$ brings added value to the exploration, measured by functions such as diversity and coverage.

Exploration action explore-around. This action returns k diverse groups that overlap with an input group g_{in} [8]. Jaccard similarity is used to implement relevance, and diversity is used for quality:

$$relevance(g_{in}, g, \textsf{explore-around}) = Jaccard(g_{in}, g) = \frac{|g_{in} \cap g|}{|g_{in} \cup g|}$$

$$quality(\mathcal{G}_k, \textsf{explore-around}) = diversity(\mathcal{G}_k) = \frac{|\bigcup_{g \in \mathcal{G}_k} g|}{\sum_{g \in \mathcal{G}_k} |g|}$$

Exploration action explore-within. This action returns k subgroups that maximize the coverage of the input group g_{in} to ensure that all exploration options within g_{in} are still available [8]. This is akin to subgroup discovery [43]:

$$relevance(g_{in}, g, \textsf{explore-within}) = \mathbb{1}[g \subseteq g_{in}]$$

The function $relevance(g_{in}, g, \textsf{explore-within})$ returns either 0 or 1. Hence in this case, σ is always set to 1. To ensure a range of exploration options, coverage is used to define quality as follows:

$$quality(g_{in}, \mathcal{G}_k, \textsf{explore-within}) = coverage(g_{in}, \mathcal{G}_k) = |\cup_{g \in \mathcal{G}_k} (g \cap g_{in})| / |g_{in}|$$

Exploration action by-facet. This action is a realization of faceted search in the group space [44]. Output groups result from splitting an input group g_{in} on a given facet (e.g., split a group by gender into males and females), i.e., $\langle a, v \rangle$, is added to the label of g_{in}, where $a \in \mathcal{A}$ and $v \in dom(a)$. Given g_{in} and an attribute a, relevance and quality are defined as follows:

$$relevance(g_{in}, g, \textsf{by-facet}) = \mathbb{1}[label(g) \setminus label(g_{in}) = \langle a, v \rangle]$$

The function $relevance(g_{in}, g, \textsf{by-facet})$ returns either 0 or 1. Hence in this case, σ is always set to 1.

$$quality(g_{in}, \mathcal{G}_k, \textsf{by-facet}) = \frac{|\{g \in \mathcal{G}_k, label(g) \setminus label(g_{in}) = \langle a, v \rangle\}|}{|dom(a)|}$$

Exploration action by-distribution. A score distribution $\tilde{g}$ can be built for each group g using the score component s in $\langle u, i, s, x \rangle \in \mathcal{D}$ (e.g., publication years of papers, ratings of products), as follows:

$$\tilde{g} = \{\langle s, count(s) \rangle : \forall u \in g, \forall i \in items(g), \exists \langle u, i, s, x \rangle \in \mathcal{D}\}$$

by-distribution finds groups with score distributions similar to an input group g_{in} [14]. In this case $relevance(g, \textsf{by-distribution})$ of a group $g \in \mathcal{G}_k$ can be expressed with a distribution comparison function (for instance Earth Mover's Distance or Kendall Tau), and $quality(\mathcal{G}_k, \textsf{by-distribution})$ is expressed using diversity:

$$relevance(g_{in}, g, \textsf{by-distribution}) = EMD(\tilde{g}, \tilde{g_{in}}) = \frac{min(work(\tilde{g}, \tilde{g_{in}}))}{min(\|\tilde{g}\|_2, \|\tilde{g_{in}}\|_2)}$$

where the function $work(\tilde{g}, \tilde{g_{in}})$ computes the matching weight between the score distributions $\tilde{g}$ and $\tilde{g_{in}}$, and $\|.\|_2$ denotes the Euclidean norm.

$$quality(\mathcal{G}_k, \textsf{by-distribution}) = diversity(\mathcal{G}_k) = \frac{|\bigcup_{g \in \mathcal{G}_k} g|}{\sum_{g \in \mathcal{G}_k} |g|}$$

Exploration action by-topic. This action operates on the text component x of $\langle u, i, s, x \rangle$ in $\mathcal{D}$. Several methods, including Latent Dirichlet Allocation (LDA) and tf-idf, can be used to process those x's and associate a topic vector $\vec{x}$ to each tuple $\langle u, i, s, x \rangle \in \mathcal{D}$ [45, 46]. In our work we use LDA where the corpus is the set of x's in all tuples of $\mathcal{D}$. Given a group g of users u, its topic vector $\vec{g}$ is obtained by combining (e.g., using sum) the topic vectors of all its associated tuples $\langle u, i, s, x \rangle$.

In by-topic, relevant groups are determined by their Cosine similarity to g_{in}'s topic vector. The quality is defined as the diversity of labels of returned groups.

$$relevance(g_{in}, g, \textsf{by-topic}) = Cosine(\vec{g}, \vec{g_{in}})$$

$$quality(\mathcal{G}_k, \textsf{by-topic}) = \frac{|\bigcup_{g \in \mathcal{G}_k} \langle a, v \rangle \in label(g)|}{\sum_{g \in \mathcal{G}_k} |\langle a, v \rangle \in label(g)|}$$

Example 2. *We now revisit our PC formation example. Assume that Martin starts with an input group containing 2 junior researchers, Sebastian Michel and Xiaokui Xiao to be included in the final committee. He applies* explore-around *to broaden his search. He discovers* 3 *groups of researchers out of which he chooses a group whose label is [prolific, SIGMOD] that contains 29 geographically-distributed and gender-distributed researchers. Martin identifies Lucian Popa, An-Hai Doan, Sihem Amer-Yahia and Michael Benedikt in that group, and applies another* explore-around *exploration to explore relevant groups. Among the 3 returned*

groups, he examines one containing 119 *highly senior researchers, and requests* explore-within *to delve into that large group. He finds the group labeled [highly senior, very productive, VLDB, ICDE] containing 26 senior researchers out of which Francesco Bonchi, Kaushik Chakrabarti, Piero Fraternali and Felix Naumann are of interest. Martin then applies a* by-topic *exploration to find other groups whose research areas are similar to those 26 researchers.* by-topic *expands the explored topics to identify a new group of 38 researchers who cover "stream processing" and "data integration", the main topic of WebDB in 2014. Martin decides to apply a* by-facet *on that group to separate males and females and balance his PC. At this stage and after 5 steps only, Martin has covered 80% of the WebDB 2014 PC.*

Example 2 illustrates the need for an iterative process to find users of interest, since the analyst does not necessarily have a complete specification of the set of target users at the beginning, and a full understanding of the underlying data distributions. An effective end-to-end exploration depends on the seed group (the set of users with which the analyst starts the exploration) as well as the series of exploration actions and their utility in helping the analyst to locate target users.

3.3 Exploration states and policies

In our EDA environment, the analyst goes through multiple steps where each step is the application of an exploration action to an input group g_{in} to obtain a set of k groups, $\mathcal{G}_k$, that constitute further exploration options. Each step generates an exploration state $s_i = \langle g_i, \mathcal{G}_{ki} \rangle$ that represents the result of applying an action e_{i-1} to group g_{i-1} (from the previous state). An exploration session $\mathcal{S}$, starting at state s_1, of length n, is a sequence of exploration states and actions of the form:

$$\mathcal{S}_{s_1} = [(g_1, \mathcal{G}_{k1}, e_1), \ldots, (g_n, \mathcal{G}_{kn}, e_n)]$$

where $g_i \in \mathcal{G}_{ki} \subseteq \mathcal{G}$ and $e_i \in E$. For instance, the exploration session of length 3 in Example 2 can be represented as: $[\langle g_1, \{g_1\}, \text{explore-around} \rangle, \langle g_2, \mathcal{G}_{k2}, \text{explore-within} \rangle, \langle g_3, \mathcal{G}_{k3}, \text{by-topic} \rangle]$, where g_1 is a starting group in $\mathcal{G}$, $\mathcal{G}_{k2}$ is the result of applying explore-around to g_1, etc.

We define a policy π as a function that takes an exploration state $s_i = \langle g_i, \mathcal{G}_{ki} \rangle$ and returns an action e_i, i.e., $\pi(s_i) = e_i$. Then, we can rewrite the definition of a session $\mathcal{S}$ starting at state s_1 and generated by a policy π as:

$$\mathcal{S}^{\pi}_{s_1} = [(s_1, \pi(s_1)), \ldots, (s_n, \pi(s_n))]$$

Utility of groups. The aim of a task is to gather a set of target users $\mathcal{U}_t \subseteq \mathcal{U}$ that are scattered in many groups in $\mathcal{G}$. Therefore, we need to measure the utility of a group g returned by an *explore*(.) function with respect to $\mathcal{U}_t$. A straightforward definition of utility would be $|g \cap \mathcal{U}_t|$. However, this definition results in higher utility for larger groups which is not desired since the analyst would need to scan many users to identify targets. For example, when Martin (the PC chair) wants to find "Asian female researchers working on machine learning", the group of "all female researchers" contains many target users, but also many more irrelevant users. We hence introduce a concentration parameter $c \in [0, 1]$ to reflect the distribution of target users in a user group. We define group utility:

$$g\_utility(g, \mathcal{U}_t) = |g \cap \mathcal{U}_t| * \mathbb{1}[g \in \mathcal{G}^c_t]$$

$$\mathcal{G}^c_t = \{g \in \mathcal{G} : \frac{|g \cap \mathcal{U}_t|}{|g|} > c\}$$

where $\mathcal{G}^c_t \subseteq \mathcal{G}$ is a set of target groups. While $\mathcal{U}_t$ characterizes the exploration goal, $\mathcal{G}^c_t$ is the set of all groups where users from $\mathcal{U}_t$ are concentrated.

Utility of a policy. The utility of an exploration session measures the total number of unique target users identified in target groups during this session discounted by the number of steps in that session with parameter $\gamma \in [0, 1]$:

$$s\_utility(\mathcal{S}, \mathcal{U}_t) = \sum_{(g_i, \mathcal{G}_{ki}, e_i) \in \mathcal{S}} \gamma^i g\_utility(g_i, \mathcal{U}_t \setminus \bigcup_{j<i} \{g_j \in \mathcal{G}^c_t\}) \quad (1)$$

Given a seed state $s_1 = \langle g_1, \mathcal{G}_{k1} \rangle$, we can now define the utility of a policy π for a target set of users $\mathcal{U}_t$:

$$p\_utility(\pi, s_1, \mathcal{U}_t) = s\_utility(\mathcal{S}^{\pi}_{s_1}, \mathcal{U}_t)$$

3.4 Guided EDA problem

We are now ready to state our problem: given the task of finding a set of target users $\mathcal{U}_t$, the guided EDA problem is formulated as finding a policy π^* with the highest utility. More formally,

$$\pi^* = argmax_{\pi} p\_utility(\pi, s_1, \mathcal{U}_t), \forall s_1 = \langle g_1, \mathcal{G}_{k1} \rangle$$

The guided EDA problem poses two major challenges: (*i*) how to simulate a human agent in such a way that we learn a policy that is applicable to any dataset and any input groups? (*ii*) how to characterize exploration states in such a way that they are independent from the underlying data, and that the decision of which action to apply next maximizes overall utility?

4. RL-BASED APPROACH

We describe our solution to the guided EDA problem using an RL-based approach. We first describe the general architecture of our approach (Section 4.1). Then, Section 4.2 presents our modeling using a Markov Decision Process (MDP), following which we reformulate our guided EDA problem for user groups in section 4.3. Last, Section 4.4 describes the RL framework and algorithms to learn and apply the best exploration policy.

4.1 Architecture of the RL-based approach

The general architecture for our RL-based approach is depicted in Figure 1. The offline phase addresses our first challenge: simulate a human experience in such a way that we learn a policy that is applicable to any dataset and any initial group. During the offline phase, an agent simulating a human analyst is trained to learn a policy that maximizes utility, e.g., for the PC of WebDB 2017. The policy is updated as the agent interacts with user groups via exploration actions. The outcome is final exploration policy that can be leveraged in an online phase to find any set of target users. For instance, the policy will apply explore-within at a given step i in case a group at step $i-1$ includes some target users but is too large and hence requires further splitting. If

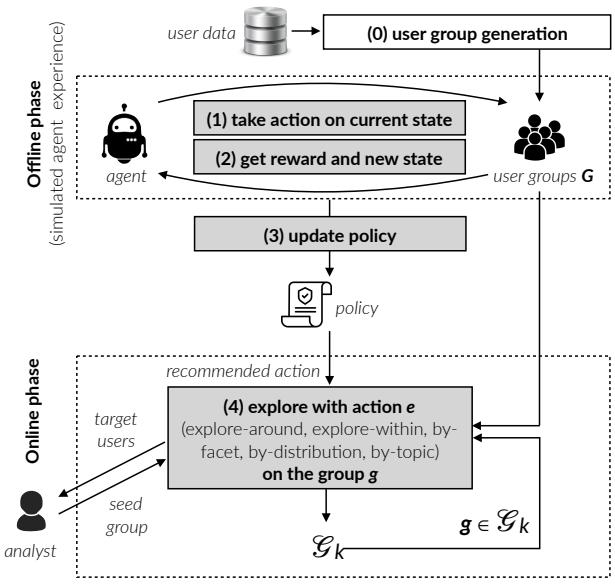


Figure 1: RL framework architecture. In the offline phase, the system iterates between steps 1 and 2 until it learns a policy (Algorithm 1). The policy (step 3) is used online to recommend exploration actions. In step 4, it is applied to any input seed group and returns target users (Algorithm 2).

instead the group is too small, it would apply explore-around at step i. Once a policy is learned, it is provided to a human analyst who applies it during the online phase to generate an interpretable exploration session that finds a PC for the same venue at a following year, e.g., WebDB PC in 2018, or for another venue, e.g., the SIGMOD PC in 2018.

4.2 Exploration model

We model EDA as a Markov Decision Process (MDP) comprising a quadruple (S, E, P, R) where:

- S is a set of states of the process;
- E is a set of exploration actions that change the process state;
- $P(s_{i+1}|s_i, e_i)$ are probabilities that action e_i will change state $s_i \in S$ to state $s_{i+1} \in S$;
- $R(s_{i+1}|s_i, e_i)$ are rewards for transitioning from state $s_i \in S$ to state $s_{i+1} \in S$ by applying exploration action e_i.

Each state s_{i+1} is a tuple $\langle g_{i+1}, \mathcal{G}_{ki+1} \rangle$ obtained by applying an exploration action e_i to a previous state $s_i = \langle g_i, \mathcal{G}_{ki} \rangle$, and for which an action e_{i+1} should be selected to continue the exploration. The probabilities $P(s_{i+1}|s_i, e_i)$ characterize the behavior of exploration actions, i.e., they represent what will be displayed next if e_i is selected. These probabilities are not known in advance and depend on the quality and relevance functions of the exploration actions and on properties of the dataset.

Reward design. The reward of a state $R(s_{i+1}|s_i, e_i)$ must reflect the utility of group g_{i+1} for a set of target users $\mathcal{U}_t$ defined in Section 3.3 (Equation 1). Reward design is known to be a challenging issue, because a reward must capture what a human analyst expects to achieve and a poorly specified reward may lead to counter-intuitive performance [9]. In our approach, the simulated RL agent is rewarded each time it discovers new target users, i.e.,

$$R(s_{i+1}|s_i, e_i) = g\_utility(g_{i+1}, \mathcal{U}_t)$$

It is important that the agent is rewarded only for targets which have not been found so far, because otherwise it would prefer to go back to the same target group [47]. This reward signal does not capture the need to maximize the total number of target users found in a session. It only prefers discovering more targets sooner starting from the current state. A reward that captures the overall utility of an exploration policy should be computed once at the end of the exploration. However, learning from such a sparse reward is too complex in our case. Therefore, we reward the agent at each intermediate step.

4.3 Reformulating the guided EDA problem

Following our MDP model, our guided EDA problem is reformulated as finding a policy $\pi : S \rightarrow E$, such that it maximizes the discounted cumulative reward $\hat{R}$:

$$\hat{R} = \sum_i \gamma^i R(s_{i+1}|s_i, e_i) \rightarrow max$$

where $e_i = \pi(s_i)$ and $\gamma \in [0, 1]$ is a discount factor.

Similarly to classical RL, given a policy π, we use a value function $V_\pi(s)$ and action-value function $Q_\pi(s, e)$:

$$V_\pi(s) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^{i+k} R(s_{i+k+1}|s_{i+k}, e_{i+k})|s_i = s]$$

$$Q_\pi(s, e) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^{i+k} R(s_{i+k+1}|s_{i+k}, e_{i+k})|s_i = s, e_i = e]$$

The function $V_\pi(s)$ computes the expected cumulative reward of the policy π gained after observing state s at step i. The function $Q_\pi(s, e)$ captures the expected cumulative reward that π gets from applying action e at state s. An optimal policy π^* always selects actions with the highest value in the current state, thus maximizing expected reward. This yields optimal functions V^* and Q^* which satisfy the Bellman optimality equations [16]:

$$V_\pi^*(s) = \max_e Q^*(s, e) =$$

$$\max_e \sum_i P(s_{i+1}|s_i = s, e_i = e)[R(s_{i+1}|s_i = s, e_i = e) + \gamma V^*(s_{i+1})]$$

Our goal is then to find π^* which yields the best exploration action at every exploration step.

4.4 RL framework

Reinforcement learning is a set of methods that find an optimal decision policy for an MDP when transition probabilities are not given. The input to an RL model is $(S, E, P, R) \setminus P$. RL fits our context, because (i) we do not assume exploration logs, and (ii) transition probabilities between exploration actions are unknown and depend on the data.

To learn an optimal policy, we simulate interactions between an analyst and groups in the offline phase. An RL agent interacts with different states and gathers several simulated exploration sessions. At each state $s_i = \langle g_i, \mathcal{G}_{ki} \rangle$, the

agent decides which exploration action e_i to select. This decision is based on the action-value function $Q(s_i, e_i)$, which the agent learns.

State-action features $\mathbf{f}(s, e)$. To enable learning interpretable EDA policies, we describe our states with a small set of domain-dependent features that reflect the result of applying an exploration action to a state. The features must enable learning how to choose between actions at each step. We expect the states generated by different actions to take different values for relevance and quality functions defined in Section 3.2: e.g., explore-around aims to generate states with higher diversity (quality), while explore-within aims to generate states with higher coverage (relevance). Features that capture relevance and quality functions will ensure that values of $P(s_i|s_{i-1}, e_{i-1})$ depend on e_{i-1}, so it is expected that we learn a policy that performs better than random. Table 2 in Section 5 contains a detailed description of the features we engineered for our empirical validation. The choice of those features is based on realistic exploration tasks in the literature [8, 44, 14] and on several trial-and-error steps and on different user datasets. We denote $\mathbf{f}(s, e)$ as a feature vector of size n for a corresponding state-action pair.

Representation of $Q(s, e)$. Typically an RL method learns a matrix of size $|S| \times |E|$ for $Q(s, e)$. In our case, our state space is vertically huge (i.e., all $\langle g, \mathcal{G}_k \rangle$ combinations). Consequently the matrix is large and highly specific to the group set $\mathcal{G}$. Hence the scope of the learned policy will be limited. To cope with that, we rely on approximate control methods [16] to learn an approximation of the action-value function $\hat{Q}(\mathbf{w}, s, e) \approx Q(s, e)$ as a function of a weight vector $\mathbf{w} \in \mathbb{R}^m$, where m is the number of state features and $m \ll |S|$. Given a state-action feature vector $\mathbf{f}(s, e)$, we define a linear approximation of $Q(s, e)$:

$$\hat{Q}(\mathbf{w}, s, e) = \mathbf{w}^T \mathbf{f}(s, e)$$

This approximation is essential for two reasons: (i) RL methods do not scale with a large number of discrete states as they need to store and train a huge matrix, but in our case we use a continuous state space and an approximate Q function as a linear combination of features (this is referred to as "approximate control"), (ii) it makes our representation data-driven but not strictly data-dependent and it blends together states with the same features in the same manner as grouping similar contexts in database exploration [9].

Learning procedure. To learn the approximation of the action-value function, we apply a common method, a semi-gradient method [16] based on a stochastic gradient descent (SGD) minimization of mean squared error:

$$err(\mathbf{w}) = \sum_{s \in S, e \in E} P(s, e)(Q(s, e) - \hat{Q}(\mathbf{w}, s, e))^2$$

where $P(s, e)$ denotes probabilities of the state-action pairs in the agent-environment interaction. During the simulated exploration, SGD updates the weights $\mathbf{w}$ to minimize the error function. The exact stochastic gradient step with the learning rate α is defined as follows:

Algorithm 1: Policy learning (offline)

Input: $E, \mathcal{G}, k, \mathcal{U}_t, \mathcal{G}_t^c, g_0, \epsilon, \alpha, \gamma, \mathbf{w}_0$
Output: *agent.policy*

```
agent.set(ε, α, γ, w_0, E)
while not end_of_learning do
    s_i ← (g_0, {g_0})
    e_i ← agent.get_exploration_action(s_i)
    while not end_of_session do
        G_{ki+1}, r ← explore(s_i, e_i)
        s_{i+1} ← (g_{i+1}, G_{ki+1})
        e_{i+1} ← agent.get_exploration_action(s_{i+1})
        agent.update_weights(s_i, e_i, r, s_{i+1}, e_{i+1})
        s_i = s_{i+1}, e_i = e_{i+1}
    end
end
```

Algorithm 2: Learned policy application (online)

Input: *agent.policy*, g_0
Output: *target users*

```
target_users ← ∅
s_i ← (g_0, {g_0})
e_i ← agent.get_exploration_action(s_i)
while not end_of_session do
    G_{ki+1}, r ← explore(s_i, e_i)
    s_{i+1} ← (g_{i+1}, G_{ki+1})
    e_rec ← agent.get_exploration_action(s_{i+1})
    target_users ← target_users ∪ e_rec(s_i)
    s_i ← s_{i+1}, e_i ← e_rec
end
```

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \alpha(Q(s_i, e_i) - \hat{Q}(\mathbf{w}_i, s_i, e_i))\nabla\hat{Q}(\mathbf{w}_i, s_i, e_i)$$

This update is done each time the RL agent observes a new state and selects an exploration action. While the true $Q(s, e)$ is unknown, we rely on its estimator as follows:

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \alpha(Q_i - \hat{Q}(\mathbf{w}_i, s_i, e_i))\nabla\hat{Q}(\mathbf{w}_i, s_i, e_i)$$

The employed estimator is proposed in the semi-gradient SARSA algorithm [16], i.e., $Q_i = R_{i+1} + \gamma\hat{Q}(\mathbf{w}_i, s_{i+1}, e_{i+1})$. SARSA is a slight variation of the popular Q-Learning algorithm. It uses the action performed by the current policy to learn the Q-value. Our final linear approximation function is therefore:

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \alpha(R_{i+1} + \gamma\mathbf{w}_i^T\mathbf{f}(s_{i+1}, e_{i+1}) - \mathbf{w}_i^T\mathbf{f}(s_i, e_i))\mathbf{f}(s_i, e_i) \quad (2)$$

These updates guarantee to converge to a local minimum with a sufficiently small learning rate α and a fixed $P(s, e)$ distribution [16].

Learning algorithm. We apply the learning procedure in an offline phase (Algorithm 1) and recommend the learned exploration policy online (Algorithm 2). Algorithm 1 shows

the process of the agent-environment interaction during policy learning. The algorithm iterates until *end_of_learning* is *true* which can be caused by a time limit or a limit on the proportion of target users found (details in Section 5). In each learning loop, the environment first returns to its initial state $s_0 = (g_0, \{g_0\})$. Then the agent iteratively observes environment states and selects corresponding actions using *get_exploration_action*(.) (line 8). With probability ϵ, the function returns

$$argmax_{e \in E} \mathbf{w}^T \mathbf{f}(g, \mathcal{G}_k, e)$$

otherwise it returns a random e. The observation steps break when the parameter *end_of_policy* is *true*. In line 6, the function *action_apply*(.) performs the selected exploration action e to change the environment state and get its corresponding reward. We set $r = |g \cap \mathcal{U}_t|$ if the selected group is in $\mathcal{G}_t$ and to 0 otherwise. The function *update_weights*(.) is responsible for learning the weights through semi-gradient updates as depicted in Equation 2 (line 9). Finally the agent with the learned set of weights $\mathbf{w}$ is returned.

Algorithm 2 describes the process of policy recommendation. At each step, the agent recommends the best exploration action e_{rec} as a function of the current state $\langle g, \mathcal{G}_k \rangle$ (line 7). The algorithm iterates until *end_of_session* is true which is the case if a time limit is reached or when a wanted proportion of target users is found.

5. EXPERIMENTS

Our experiments first seek to validate the use of RL for EDA on user data, and then to examine the utility of our learned policies on real user datasets.

Summary of results. We first observe that increasing the number of sessions in offline learning yields a decrease in the number of steps to find target users in the online phase (Figure 3), which validates the effectiveness of learned policies (Section 5.3). We show how the scattering of target users impacts the learned policy (Section 5.5): we see that explore-around and by-distribution are favored when targets are scattered in diverse groups, whereas by-facet is favored when targets are concentrated in fewer groups; we observe that the likelihood of using explore-around decreases after many target users have been discovered; also the likelihood of using by-distribution increases when handling smaller groups. Last, we illustrate that while different tasks may require different policies, it is possible to transfer policies between tasks (Section 5.4). We show that training policies on larger datasets (e.g., SIGMOD PC) may lead to overfitting when we transfer the policy to smaller datasets (e.g., WebDB PC). The inverse transfer is shown to perform well.

5.1 Datasets

We test our system on DM-Authors, a dataset we built from DBLP [4]. DM-Authors includes 1,860 researchers with a total of 200,000 publications in data management venues between 2000 and 2018: VLDB, SIGMOD, ICDE, WWW, RecSys, EDBT, DEXA, WebDB, and HILDA. We crawled profiles of researchers from DBLP and added demographic attributes.

Following our data model (Section 3), we describe the quadruple $\langle u, i, s, x \rangle$ in DM-Authors as follows: $u \in \mathcal{U}$ is a researcher, $i \in \mathcal{I}$ is a publication by a researcher, $s \in [1, 5]$ is a normalized value for publication recency, and x is a bag of words from the publication title. The score s is assigned based on the year of publication: $[2017, 2018] \rightarrow 5$, $[2014, 2017] \rightarrow 4$, $[2010, 2013] \rightarrow 3$, $[2006, 2009] \rightarrow 2$, $[2000, 2005] \leftarrow 1$. For instance, the quadruple $\langle$*Volker Markl*, *VLDB*, 5, {*fault-tolerance*, *dataflows*}$\rangle$ represents that Volker Markl published a paper in VLDB during the years 2017 and 2018 ($s = 5$) whose title contains "fault-tolerance" and "dataflows".[1]

Demographic and item attributes. Table 1 describes the set of demographic attributes $\mathcal{A}_\mathcal{U}$ for each researcher in DM-Authors. The set $\mathcal{A}_\mathcal{I}$ has one attribute, i.e., the venue (conference or workshop) that the paper was published in.

Table 1: Demographic attributes in DM-Authors.

Attribute	Description	Values
Seniority	Number of years since the first publication in researcher's DBLP. Values are chosen to equalize the number of researchers in each category.	*starting* (1-8 years), *junior* (9-12), *senior* (13-15), *highly senior* (16-21), *confirmed* (22+)
Publication rate	Average number of publications per year. Values are chosen to equalize the number of researchers in each category.	*active* (0.18-1.47), *very active* (1.48-2.48), *productive* (2.49-3.71), *very productive* (3.72-6.0), *prolific* (6.1+)
Location	Extracted from researchers' affiliations in DBLP profiles.	*North America*, *UK/Ireland*, *South America*, *Europe*, *East/South Asia*,*Australia*, *Middle East*, *other*
Gender	Extracted by matching researcher's first name to a database of more than 40,000 names.[2]	*male*, *female*

User groups. To build the set of groups $\mathcal{G}$, we rely on LCM, an implementation of the Apriori algorithm for closed frequent pattern mining [48]. LCM admits $\mathcal{D}$ and a support threshold η, and returns a set of frequent patterns which contain at least η users. Each frequent pattern is described with demographics, items, and item attributes which are common to all η users of the pattern. Hence each pattern forms a user group g where $label(g)$ is the pattern itself. We mined 26,648 groups with a support value set to $\eta = 10$. The size of group labels in $\mathcal{G}$ varies between 1 and 5.

State-action features. In Section 4.4, we proposed to describe each state-action pair (s, e) with a set of domain-dependent features, $\mathbf{f}(s, e)$, and use those features in our

[1] https://dblp.org/rec/html/journals/pvldb/XuLSM18
[2] https://github.com/ferhatelmas/sexmachine/

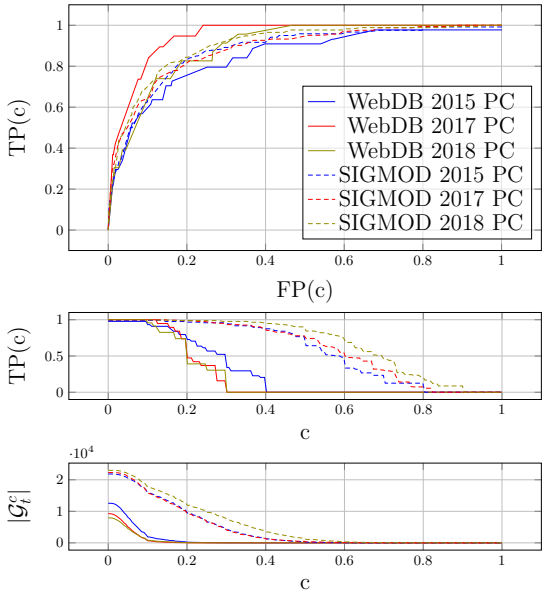


Figure 2: Quality of $\mathcal{G}$ with respect to targets.

learning process. The first column of Table 2 describes the state-action features considered in this work, whose design is influenced on one hand by the relevance and quality functions of our exploration actions (represented as diversity and coverage), and on the other hand by the analysis task of finding target users (e.g., support of input group, number of item attributes in its label, number of target users discovered at that step). The second column of the table describes how the features are instantiated for the dataset schema considered in our experiments. These features are applicable to different user datasets, such as forming a PC, and gathering a movie critic panel.

5.2 Experimental setup

Our exploration goal is to gather a set of researchers to serve on a PC. Our ground-truth consists of real PCs of one large and one small venue, i.e., the SIGMOD conference and the WebDB workshop, in the years 2015, 2017, and 2018.

Learning variants. To compare different policies, we define 3 learning variants each of which has a different train set. All the variants share the same test set, namely 2018 PC of either WebDB or SIGMOD. We run offline policy learning on the train set and use the learned policy to explore the test set. The learning variants are: `OLDEST` where the train set is the 2015 PC of the same venue, `LAST` where the train set is the PC of the previous year's venue, i.e., 2017, and `TRANSFER` where the train set is the PC of the other venue: SIGMOD for WebDB, and vice versa.

Learning parameters. In real exploration tasks, target users are *scattered* over the group set, and the analyst needs to scan many different user groups to achieve a task. We propose to measure the overall utility of $\mathcal{G}$ for locating the target users using "true positive" and "false positive" rates (Equation 3).

$$TPR(\mathcal{G}_t^c) = \frac{|\bigcup_{g\in\mathcal{G}_t^c}(\mathcal{U}_t \cap g)|}{|\mathcal{U}_t|};\ FPR(\mathcal{G}_t^c) = \frac{|\bigcup_{g\in\mathcal{G}_t^c}(g \setminus \mathcal{U}_t)|}{|\mathcal{U} \setminus \mathcal{U}_t|} \tag{3}$$

The true positive rate $TPR(.)$ measures the fraction of target users $\mathcal{U}_t$ found in target groups $\mathcal{G}_t^c$. Also the false positive rate $FPR(.)$ measures the fraction of non-target users found in target groups. Plotting $TPR(\mathcal{G}_t^c, c)$ against $FPR(\mathcal{G}_t^c, c)$ for different values of c is analogous to a receiver operating characteristic (ROC) curve. We can then employ the area under the curve (ROC-AUC) to measure the utility of a group set with respect to a set of target users $\mathcal{U}_t$.

To choose target groups and check how well a group set $\mathcal{G}$ locates target users, we vary the concentration parameter c and examine the evolution of TPR and FPR measures (Equation 3) in a ROC curve. The results are illustrated in Figure 2. In general, we observe that ROC-AUC values for all the sets of target users are high, hence exploring $\mathcal{G}$ is potentially beneficial for the task under investigation. In all our experiments, we set $c = 0.3$ for a large venue like SIGMOD, and $c = 0.1$ for WebDB. This means that, in the worst case, one needs to scan a group of ten people to find one PC member for SIGMOD. We observe in Figure 2 that with the aforementioned values of c, $\mathcal{G}_t$ includes only around 10% of all groups that overlap with $\mathcal{U}_t$, but still covers almost all PC members of WebDB 2015, 2017 and 2018. Table 3 summarizes the properties of $\mathcal{U}_t$ and $\mathcal{G}_t$.

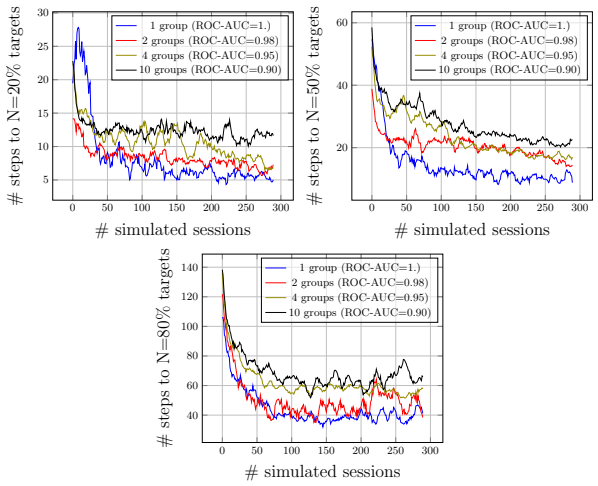


Figure 3: Learning curve for the synthetic experiments with different scattering levels: 20 target users are scattered equally over 1, 2, 4 or 10 groups.

In the offline phase, we set $k = 5$ and choose the next group to explore at random. If $\mathcal{G}_k$ includes groups with target users not discovered earlier, we simulate the choice of an analyst by randomly selecting one of those groups. And if there are no such groups, the choice of g is random.

Exploration actions. We use all the exploration actions in Section 3.2 with different values for the relevance threshold σ: explore-around with $\sigma = 0.2$, by-distribution with $\sigma = 0.05$, and by-topic with $\sigma = 0.1$, and explore-within and by-facet with $\sigma = 1.0$. We also consider the action undo to enable returning to the previous state if no target user has been

Table 2: State-action features. All features are encoded as Boolean values.

Feature	Description
Diversity of $\mathcal{G}_k$	binary features representing 5 equal-width intervals between 0 and 1
Coverage of g_{i-1}	binary features representing 5 equal-width intervals between 0 and 1
Number of displayed groups $\lvert\mathcal{G}_k\rvert$	2 binary features to determine whether more than one group is displayed or not
Size of input group $\lvert g_{in}\rvert$	binary features for the following intervals: $[0, 15]$, $[16, 50]$, $[51, 100]$, $[101, 200]$, $[201, 500]$ and $[501, \infty)$
Number of item attributes in *label*(g_i)	3 features for "0 item attributes", "between 1 and 2 item attributes", and "more than 2"
Number of demographic attributes in *label*(g_i)	3 features for "0 demographic attributes", "between 1 and 2 demographic attributes", and "more than 2"
Previously discovered target users	2 features to capture whether g_i contains target users or not
Rating distribution of g_i	3 features for *low*, *uniform*, and *high* distributions, computed using Earth Mover's Distance between $\tilde{g}_i$ and the distributions: $[1, 0, ..., 0]$, $[0, ..., 0, 1]$, $[\frac{1}{k}, \frac{1}{k}, ..., \frac{1}{k}]$
Number of discovered target users	features for the following intervals: $[0, 1]$, $[2, 3]$, $[4, 5]$, $[6, 7]$ and $[8, \infty)$
Presence of demographic attributes	8 features for 4 facets, i.e., gender, seniority level, productivity, and location
Reward	2 features to capture whether g_i yielded a positive reward (i.e., if new target users were selected from the group) or not
Previous exploration action	one feature per exploration action $e \in E$

Table 3: Summary of statistics about $\mathcal{U}_t$ and $\mathcal{G}_t$.

	WebDB'15	WebDB'17	WebDB'18	SIGMOD'15	SIGMOD'17	SIGMOD'18
# members $\lvert\mathcal{U}_t\rvert$	43	19	22	119	173	163
$\lvert\mathcal{G}_t\rvert$	1874	649	880	4145	3985	7405
ROC-AUC	0.85	0.94	0.89	0.83	0.81	0.82

observed. We use LDA to generate topic vectors of length 10 for by-topic.

The learning parameters are $\alpha = 0.001, \gamma = 0.5$. The number of sessions is set as $\epsilon = \min(10/n, 1)$. Initial weights $\mathbf{w}_0$ are set to zero. Each exploration action online and offline has a time limit of 100ms. An offline learning with 300 sessions with an average of 50 steps per session needs $100ms * 50 * 300 = 25min$. The total number of steps is limited to 200. All our results are averages of 10 runs of the offline learning with many random seed groups. During the learning process some actions are chosen at random (with decreasing probability ϵ), as it is usually done in RL.

5.3 Synthetic exploration

We run a number of synthetic experiments with different levels of concentration of target users $\mathcal{U}_t$ in groups in $\mathcal{G}$. In all the experiments, $|\mathcal{U}_t| = 20$. In the first experiment, all target users are concentrated in the same group. We choose "females from Europe who published in VLDB" as the set of target users, which fits in a single group with the label $[female, VLDB, Europe]$. In the second experiment, the same number of target users is gathered from two non-overlapping groups: $[female, VLDB, Europe]$ and $[male, AAAI, Asia]$ and each of these groups contains 10 targets. In two other experiments, we scatter targets over 4 and 10 groups in the same way. In all the experiments $c = 0.1$. Figure 3 reports the number of steps required in the learning phase of the experiments. One can clearly see that scattering targets over more groups increases the time it takes the agent to reach its targets.

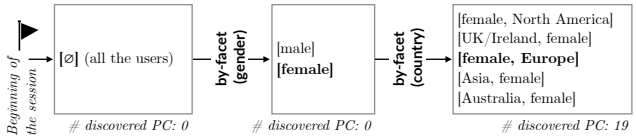
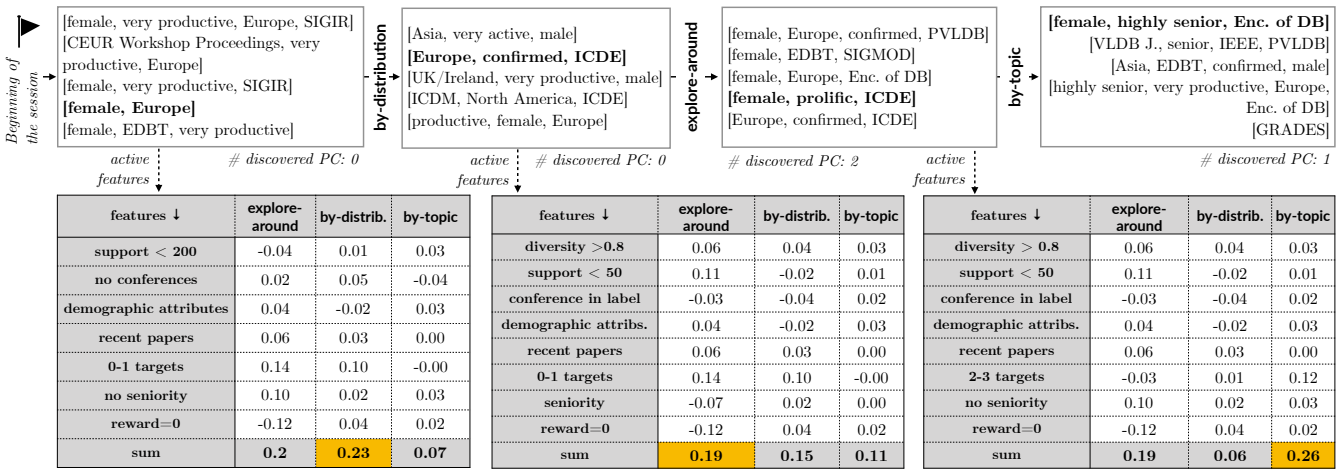


Figure 4: Simulated session with a policy trained on the synthetic task.

More specifically, we can see that the task of finding target users concentrated in one group takes only a few steps. The reason is that the ROC-AUC value of the synthetic setting is equal to 1.0. We delve into one of the learned policies that reaches the target in only 2 steps (Figure 4) and observe that this policy does not make use of explore-around and by-distribution. It favors by-facet on gender and location that help locate the single target group of interest more quickly. This simple experiment is a proof-of-concept showing that when target users are concentrated in the same group, our policy is equivalent to simple SQL queries (on gender and location). Our subsequent experiments will require more sophisticated policies to reach target users.

5.4 Impact of learning variants

Our second experiment examines the impact of the learned policies in each learning variant, `OLDEST`, `LAST` and `TRANSFER`. We measure the utility of each policy as the percentage of discovered PC members (see Section 3.3), denoted as ν. Figure 8 illustrates the learning curves.



features ↓	explore-around	by-distrib.	by-topic
support < 200	-0.04	0.01	0.03
no conferences	0.02	0.05	-0.04
demographic attributes	0.04	-0.02	0.03
recent papers	0.06	0.03	0.00
0-1 targets	0.14	0.10	-0.00
no seniority	0.10	0.02	0.03
reward=0	-0.12	0.04	0.02
sum	0.2	0.23	0.07

features ↓	explore-around	by-distrib.	by-topic
diversity >0.8	0.06	0.04	0.03
support < 50	0.11	-0.02	0.01
conference in label	-0.03	-0.04	0.02
demographic attribs.	0.04	-0.02	0.03
recent papers	0.06	0.03	0.00
0-1 targets	0.14	0.10	-0.00
seniority	-0.07	0.02	0.00
reward=0	-0.12	0.04	0.02
sum	0.19	0.15	0.11

features ↓	explore-around	by-distrib.	by-topic
diversity > 0.8	0.06	0.04	0.03
support < 50	0.11	-0.02	0.01
conference in label	-0.03	-0.04	0.02
demographic attribs.	0.04	-0.02	0.03
recent papers	0.06	0.03	0.00
2-3 targets	-0.03	0.01	0.12
no seniority	0.10	0.02	0.03
reward=0	-0.12	0.04	0.02
sum	0.19	0.06	0.26

Figure 5: Simulated session example. For each step, the labels of the k groups and their active features are shown. The agent chooses actions with the highest sum of weights (highlighted in yellow).

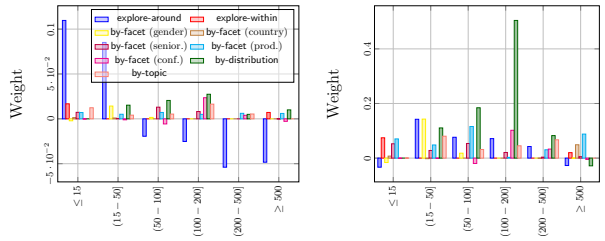


Figure 6: Learned weights for the feature describing the size of the input group (x-axis), trained on WebDB 2017 (left) and SIGMOD 2017 (right). Each line corresponds to one exploration action. For a given exploration state, only one of the binary features displayed here takes value 1, others are 0.

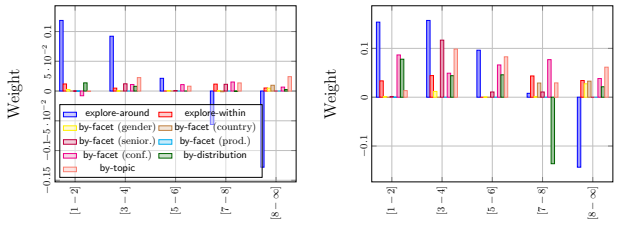


Figure 7: Learned weights for the features showing the number of targets discovered so far in the exploration (x-axis), trained on WebDB 2017 (left) and SIGMOD 2017 (right).

Initially, the agent acts randomly, choosing any exploration action, then over time it discovers useful actions in the observed states. One can see that the `TRANSFER` curve declines which means that the policy based on the state-action features is useful for similar exploration goals, not only for the goal used in the training.

The figure shows that the number of steps to find 20, 50, and 80% of target users decreases over time. This indicates that the RL agent is able to improve performance, i.e., increase ν, through the learning process. We observe that in general the policy learning improves the agent's performance as the agent needs on average fewer steps to find ν% of target users with more training sessions. We also observe that the performance of the learning variants differs significantly. For WebDB'18, `LAST` and `OLDEST` perform significantly better than `TRANSFER`, as they require fewer exploration steps to reach the same goal. The low performance of `TRANSFER` from SIGMOD to WebDB is potentially due to the large size of the SIGMOD PC which results in overfitting and increasing variance drastically. On the other hand, the policies trained on WebDB'18 perform well for SIGMOD'18, indicating that policy transfer is useful and could be examined for other venues in the future.

While different exploration tasks require different policies, it is also possible to transfer policies between similar tasks. We also observe that the difference in performance between learning variants increases with ν. The task becomes more specific when the goal is to discover more target users, so it is useful to train the policy on similar tasks. Finally, we validate the usefulness of our state-action features in guiding the learning process.

5.5 Decision making

To better understand how the RL agent makes decisions during exploration, we delve into the EDA process and visualize several steps in Figure 5. Our general observation is that the decision highly depends on the weights that are assigned to different features of the exploration.

To interpret and compare different policies discussed in Section 5.4, we examine the vector $\mathbf{w}$ of the learned feature weights describing the size of the input group (Figure 6) and the number of discovered targets (Figure 7). In both figures, WebDB'17 is used as a train set on the left and SIGMOD'17 is used as a train set on the right. Both policies use mostly explore-around and by-distribution. These two actions return *many diverse groups*, which is beneficial when target users are scattered over many different groups.

One can notice similar patterns in the policies: when increasing the number of targets discovered so far, the likelihood of using explore-around decreases (Figure 7). As it

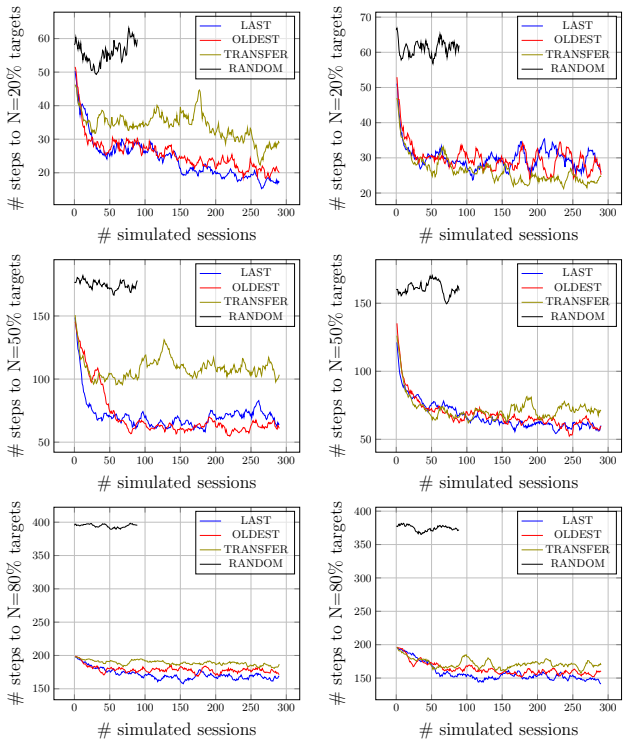


Figure 8: Learning curves of simulated sessions that run until discovering 20% (top), 50% (middle), 80% (bottom) of WebDB (left) and SIGMOD (right) PCs. The curves show the average number of steps the agent takes to reach the exploration goal after learning in a number of simulated sessions displayed in the x-axis. Each point in each run is the mean over a window of 10 last exploration sessions. Exploration probability decreases with the number of sessions leading to convergence.

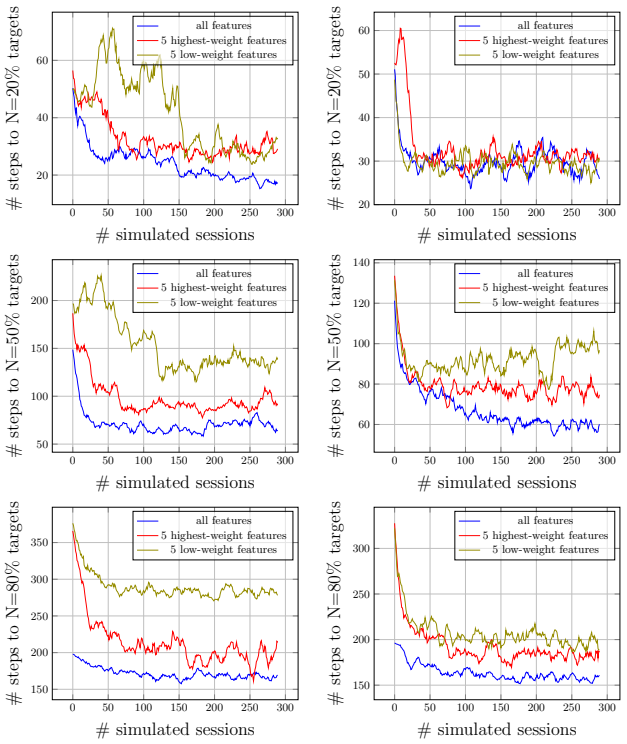


Figure 9: Learning curves for the experiments from Figure 8 run with three different sets of features: all the features from Table 2 (blue line), 5 features that had the greatest absolute weights in the policy learned with all the features (red line), 5 diversity features from Table 2 that had low weights in all the policies trained with all the features (brown line).

becomes less probable to find new targets in the groups overlapping with an input group (via other actions than explore-around), it is better to *jump to new groups* to increase the likelihood of finding target users.

One can also see some differences: the policy trained for SIGMOD strongly prefers by-distribution. A possible explanation is the larger ratio of target users to all users for the SIGMOD PC, as by-distribution can return more users with recent papers regardless of their research topics. The agent sees more active researchers from various areas which increases the likelihood of discovering SIGMOD PC members that make almost 20% of the dataset. On the other hand, the policy trained for WebDB strongly prefers explore-around, most likely because the agent explores more groups to locate sparsely distributed WebDB PC members that only make around 1% of the dataset.

5.6 Impact of features

Figure 9 shows how our state features described in Table 2, impact the performance of learned policies. We run the same experiments as in Figure 8 using only a subset of the features to see how that impacts the number of steps needed to reach target users. In particular, for each learning task we choose 5 features (10% of all features) that have high absolute weights in the policy trained with all the features. A high feature weight reflects that it contributes to the action-value function $Q(\mathbf{w}, s, e) = \mathbf{w}^T\mathbf{f}(s, e)$ and thus significantly impacts decision making. We also choose a subset of features that have low weights in all experiments, i.e., 5 diversity features from Table 2. One can see in Figure 9 that the subset of the high-weight features yields better performance in all experiments in comparison with the subset of low-weight features. However, the set of all features still outperforms both subsets. One can also see that the difference in performance is significant for $N = 50\%$ and $N = 80\%$ while to reach the smallest number of targets (i.e., $N = 20\%$), all policies are comparable because they are simpler and mostly rely on explore-around regardless of the state features.

6. CONCLUSION

We examined and validated the applicability of machine learning techniques to EDA on user data where the task is to find a set of users. Our unified formalization leverages a wide range of user data exploration actions, and our solution is based on reinforcement learning to learn an exploration policy. Our experiments validate the choice of state features we made, allowing us to claim generality for the task of finding a set of users.

We are currently addressing new challenges raised by adding text-based exploration operations that are applicable to reviews and tags.

7. REFERENCES

[1] Behrooz Omidvar-Tehrani and Sihem Amer-Yahia. User group analytics survey and research opportunities. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[2] Qualtrics Marketplace (SAP). https://www.qualtrics.com/marketplace/.

[3] Amplitude Behavioral Analytics Platform. https://amplitude.com/behavioral-analytics-platform/.

[4] DBLP computer science bibliography. https://dblp.uni-trier.de/db/.

[5] GroupLens Research. MovieLens dataset. https://grouplens.org/datasets/movielens/, 2019.

[6] Behrooz Omidvar-Tehrani and Sihem Amer-Yahia. Tutorial on user group analytics: Discovery, exploration and visualization. In *International Conference on Information and Knowledge Management (CIKM)*, pages 2307–2308, 2018.

[7] Fabian Colque Zegarra, Juan C Carbajal Ipenza, Behrooz Omidvar-Tehrani, Viviane P Moreira, Sihem Amer-Yahia, and João LD Comba. Visual exploration of rating datasets and user groups. *Future Generation Computer Systems (FGCS)*, 105:547–561, 2020.

[8] Behrooz Omidvar-Tehrani, Sihem Amer-Yahia, and Alexandre Termier. Interactive user group analysis. In *International Conference on Information and Knowledge Management (CIKM)*, pages 403–412, 2015.

[9] Tova Milo and Amit Somech. Next-step suggestions for modern interactive data analysis platforms. In *International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 576–585, 2018.

[10] Ori Bar El, Tova Milo, and Amit Somech. ATENA: an autonomous system for data exploration based on deep reinforcement learning. In *International Conference on Information and Knowledge Management (CIKM)*, pages 2873–2876, 2019.

[11] Ori Bar El, Tova Milo, and Amit Somech. Automatically generating data exploration sessions using deep reinforcement learning. In *International Conference on Management of Data (SIGMOD)*, 2020.

[12] Ning Yan, Chengkai Li, Senjuti B Roy, Rakesh Ramegowda, and Gautam Das. Facetedpedia: enabling query-dependent faceted search for wikipedia. In *International Conference on Information and Knowledge Management (CIKM)*, pages 1927–1928, 2010.

[13] Lanbo Zhang and Yi Zhang. Interactive retrieval based on faceted feedback. In *International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 363–370, 2010.

[14] Sihem Amer-Yahia, Sofia Kleisarchaki, Naresh Kumar Kolloju, Laks V.S. Lakshmanan, and Ruben H. Zamar. Exploring rated datasets with rating maps. In *International Conference on World Wide Web (WWW)*, 2017.

[15] Mahashweta Das, Saravanan Thirumuruganathan, Sihem Amer-Yahia, Gautam Das, and Cong Yu. Who tags what? an analysis framework. *Proc. VLDB Endow.*, 5(11):1567–1578, 2012.

[16] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[17] Amit Somech, Tova Milo, and Chai Ozeri. Predicting "what is interesting" by mining interactive-data-analysis session logs. In *International Conference on Extending Database Technology (EDBT)*, pages 456–467, 2019.

[18] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *International Conference on World Wide Web (WWW)*, pages 661–670, 2010.

[19] Shuai Li, Alexandros Karatzoglou, and Claudio Gentile. Collaborative filtering bandits. In *International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 539–548, 2016.

[20] Yilin Shen and Hongxia Jin. Epicrec: Towards practical differentially private framework for personalized recommendation. In *International Conference on Computer and Communications Security (SIGSAC)*, pages 180–191. ACM, 2016.

[21] Francesco Bonchi, Fosca Giannotti, Claudio Lucchese, Salvatore Orlando, Raffaele Perego, and Roberto Trasarti. Conquest: a constraint-based querying system for exploratory pattern discovery. In *International Conference on Data Engineering (ICDE)*, pages 159–159, 2006.

[22] Francesco Bonchi, Fosca Giannotti, Alessio Mazzanti, and Dino Pedreschi. Exante: Anticipated data reduction in constrained pattern mining. In *European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, volume 2838, pages 59–70. Springer, 2003.

[23] Niranjan Kamat, Prasanth Jayachandran, Karthik Tunga, and Arnab Nandi. Distributed and interactive cube exploration. In *International Conference on Data Engineering (ICDE)*, pages 472–483, 2014.

[24] Davide Mottin, Matteo Lissandrini, Yannis Velegrakis, and Themis Palpanas. New trends on exploratory methods for data analytics. *Proc. VLDB Endow.*, 10(12):1977–1980, 2017.

[25] Minsuk Kahng, Shamkant B. Navathe, John T. Stasko, and Duen Horng (Polo) Chau. Interactive browsing and navigation in relational databases. *Proc. VLDB Endow.*, 9(12):1017–1028, 2016.

[26] Florian Lemmerich, Martin Becker, Philipp Singer, Denis Helic, Andreas Hotho, and Markus Strohmaier. Mining subgroups with exceptional transition behavior. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 965–974, 2016.

[27] Mikalai Tsytsarau, Sihem Amer-Yahia, and Themis Palpanas. Efficient sentiment correlation for large-scale demographics. In *International Conference on Management of Data (SIGMOD)*, pages 253–264, 2013.

[28] Yi Yang, Da Yan, Huanhuan Wu, James Cheng, Shuigeng Zhou, and John CS Lui. Diversified temporal subgraph pattern mining. In *International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 1965–1974, 2016.

[29] Tianyang Zhang, Peng Cui, Christos Faloutsos, Yunfei Lu, Hao Ye, Wenwu Zhu, and Shiqiang Yang. Come-and-go patterns of group evolution: A dynamic model. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1355–1364, 2016.

[30] Gloria Chatzopoulou, Magdalini Eirinaki, and Neoklis Polyzotis. Query recommendations for interactive database exploration. In *International Conference on Scientific and Statistical Database Management (SSDBM)*, pages 3–18. Springer, 2009.

[31] Roee Ebenstein, Niranjan Kamat, and Arnab Nandi. Fluxquery: An execution framework for highly interactive query workloads. In *International Conference on Management of Data (SIGMOD)*, pages 1333–1345, 2016.

[32] Magdalini Eirinaki, Suju Abraham, Neoklis Polyzotis, and Naushin Shaikh. Querie: Collaborative database exploration. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 26(7):1778–1790, 2014.

[33] Nodira Khoussainova, YongChul Kwon, Magdalena Balazinska, and Dan Suciu. Snipsuggest: Context-aware autocompletion for SQL. *Proc. VLDB Endow.*, 4(1):22–33, 2010.

[34] Julien Aligon, Enrico Gallinucci, Matteo Golfarelli, Patrick Marcel, and Stefano Rizzi. A collaborative filtering approach for recommending OLAP sessions. *Decision Support Systems*, 69:20–30, 2015.

[35] Patrick Marcel and Elsa Negre. A survey of query recommendation techniques for data warehouse exploration. In *Journées Francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA)*, pages 119–134, 2011.

[36] Cristiana Bolchini, Elisa Quintarelli, and Letizia Tanca. Context support for designing analytical queries. In *Methodologies and Technologies for Networked Enterprises*

- *ArtDeco: Adaptive Infrastructures for Decentralised Organisations*, pages 277–289. Springer, 2012.

[37] Marina Drosou and Evaggelia Pitoura. Ymaldb: exploring relational databases via result-driven recommendations. *VLDB J.*, 22(6):849–874, 2013.

[38] Manas Joglekar, Hector Garcia-Molina, and Aditya G. Parameswaran. Interactive data exploration with smart drill-down. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 31(1):46–60, 2019.

[39] Sunita Sarawagi, Rakesh Agrawal, and Nimrod Megiddo. Discovery-driven exploration of OLAP data cubes. In *International Conference on Extending Database Technology (EDBT)*, pages 168–182, 1998.

[40] Manish Singh, Michael J Cafarella, and HV Jagadish. DBExplorer: Exploratory search in databases. In *International Conference on Extending Database Technology (EDBT)*, pages 89–100, 2016.

[41] Liangda Li, Hongbo Deng, Yunlong He, Anlei Dong, Yi Chang, and Hongyuan Zha. Behavior driven topic transition for search task identification. In *International Conference on World Wide Web (WWW)*, pages 555–565, 2016.

[42] Ioannis Arapakis, Mounia Lalmas, and George Valkanas. Understanding within-content engagement through pattern analysis of mouse gestures. In *International Conference on Information and Knowledge Management (CIKM)*, pages 1439–1448, 2014.

[43] Tobias Scheffer and Stefan Wrobel. A sequential sampling algorithm for a general class of utility criteria. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2000.

[44] Marti A. Hearst. Clustering versus faceted categories for information exploration. *Communications of the ACM (CACM)*, 49(4), April 2006.

[45] Hamed Jelodar, Yongli Wang, Chi Yuan, Xia Feng, Xiahui Jiang, Yanchao Li, and Liang Zhao. Latent dirichlet allocation (LDA) and topic modeling: models, applications, a survey. *Multimedia Tools and Applications*, 78(11):15169–15211, 2019.

[46] Zhiwen Tang and Grace Hui Yang. Deeptilebars: Visualizing term distribution for neural information retrieval. In *International Conference on Artificial Intelligence (AAAI)*, volume 33, pages 289–296, 2019.

[47] Jack Clark and Dario Amodei. Faulty reward functions in the wild. https://blog.openai.com/faulty-reward-functions/, 2016.

[48] Takeaki Uno, Masashi Kiyomi, and Hiroki Arimura. LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In *IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI)*, volume 126, 2004.