Guided exploration of user groups

Mariia Seleznova, Behrooz Omidvar-Tehrani, Sihem Amer-Yahia, Eric Simon

Guided Exploration of User Groups

Mariia Seleznova[1]*, Behrooz Omidvar-Tehrani[2], Sihem Amer-Yahia[1], Eric Simon[3]
[1]CNRS, University of Grenoble Alpes, [2]NAVER LABS Europe, [3]SAP Paris
[1]firstname.lastname@univ-grenoble-alpes.fr,
[2]behrooz.omidvar-tehrani@naverlabs.com, [3]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 Selezniova, Behrooz Omidvar-Tehrani, Sihem Amer-Yahia, Eric Simon. Guided Exploration of User Groups. *PVLDB*, 12(xxx): xxxx-yyyy, 2020.
DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx

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 [27]. 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. [1].

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, commerce and sales, and customer services [27]. For instance, a data scientist may conduct large-scale population studies to gain insights on the preferences of various population segments. An information consumer may want to explore alike user groups to be inspired for routine tasks such as choosing a product or picking a service subscription. In this paper, we take as an example a PC chair (a domain expert) who wants to build a program committee, starting with any seed group (researchers in this case) and iteratively looking for users in groups that match certain expected properties expected of a PC (geographic distribution, gender and topic balance, etc).

Performing user group analysis with hand-crafted queries, which is still the most popular approach, comes with several challenges. Firstly, the analyst needs to master a language (e.g., SQL) to formulate queries, as well as a global understanding of the underlying database schema. Secondly, the analyst usually lacks information about the distribution of attribute values that characterize users or items on which users performed some action. Thus, a query that selects users based on user or item attributes may either return too few or too many results. For instance, a query that separates users into groups of males and females can surprisingly return very small output groups when most users do not have gender information. Thirdly, an analyst may want to refine her queries as she discovers more users. For instance, a PC chair usually has a partial idea of who to look for, such as a mix of senior and junior male and female researchers around the world. Hence, the PC chair may want to navigate in the space of potential PC members to iteratively choose who to include in the final PC depending on a mix of gender, coverage of topics, seniority, and countries obtained so far.

Due to the iterative nature of the task and its inherent difficulty, 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.,

*This work was completed when the author was working as an intern at SAP.

Proceedings of the VLDB Endowment, Vol. 12, No. xxx
ISSN 2150-8097.
DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx

[1]https://www.qualtrics.com/marketplace/
https://amplitude.com/behavioral-analytics-platform/

groups of researchers who published in SIGMOD), 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, publish in VLDB] that has researchers with well distributed geographical location and gender, and apply an `explore-around` action to return k diverse groups that overlap with the selected group). Although there exists a large body of work on the recommendation of data exploration actions, carefully reviewed in Section 2, these works have important shortcomings. First, most work recommend (SQL/OLAP) queries, which is not adapted to analysts with no IT expertise, while other works focus on specific types of high-level actions that are too limited for user group analysis. Second, existing solutions either rely on tedious manual exploration [28], or on the collection of a large log of exploration sessions gathered from multiple users on the same dataset [25], which is not an acceptable assumption in our case since many analysis tasks are repeated only a few times or performed on different datasets. Last, very recent work on automating the exploration process does not generate interpretable policies [13].

In this paper, we propose the first framework for user group exploration that learns an exploration policy without requiring prior collection of exploration log. The learned policy is used to recommend an end-to-end exploration session on a given dataset. The recommended sessions can be easily interpreted by an analyst with no IT expertises. Our framework supports exploration sessions consisting of a sequence of high-level exploration actions of various types and can provide recommendations for the exploration of new datasets, provided they are similar (with respect to domain-specific features) to previously analyzed datasets. Additionally, our framework learns an exploration policy offline from a simulated agent experience in such a way that no human intervention is needed from an analyst during the online analysis task. It thus solves a fully guided EDA for user group analysis when the task is to find target users.

Our first contribution is to formalize the guided EDA problem applied to user groups as a Markov decision process where a state displays several user groups, a transition is the application of an exploration action to a chosen group, and the reward is a function of whether or not some target users are found when applying the action to the group. An exploration session is hence a sequence of exploration actions, and an exploration policy is a function that maps a state to an action. Our framework integrates a comprehensive set of high-level exploration actions for user groups that were introduced separately by previous work: explore-around [28], explore-within [28], by-facet [37], by-distribution [2], and by-topic [9]. Those actions were never considered all together before. We define a unified semantics for all these actions to make them composable and take advantage of their expressivity at each exploration step. We then formalize the problem of guided EDA on user groups as finding a policy that maximizes utility, i.e., that finds as many target users as possible regardless of the dataset, the user groups and the seed groups at every step.

Our second contribution is the of use Reinforcement Learning (RL) to learn an efficient policy from a simulated agent experience (i.e., an RL agent that acts as an analyst who chooses exploration actions), and recommend a policy. A notable advantage of RL methods is that, unlike supervised learning methods, they do not require to gather labeled datasets. Instead, the agent learns from rewards computed by the environment during the interaction [34]. 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 decision-making process. Our work bears similarity with recent proposals on RL-based EDA [25, 33] but differs in several ways. First, we address a well-defined user data analysis task (gathering target users) and the exploration actions are chosen accordingly. Additionally, in case of interactive exploration, the analysis task is often unique and needs to be accomplished only once by an analyst. Running a large number of manual explorations for each new dataset and new analysis task is hence impractical. In [13], the authors propose to use a deep RL approach for EDA. However, 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 [21, 23, 31]. Our reliance on classical RL methods is a first step towards a deeper and comprehensive understanding of how to learn data exploration policies for user data analysis, including thorough experiments on real datasets.

Our third contribution is an extensive set of experiments that validate the use of RL to solve the guided EDA problem for user groups. We study various cases where target users to be found are scattered in groups. We examine the utility of learned policies on real user datasets for different variants where policies are learned on the same conference or transferred from one conference to another. Our experiments show that the simulated agent succeeds to learn an interpretable strategy for several realistic use cases of interactive exploration. It also validates that our exploration actions and offline policy learning naturally capture what a human analyst does manually (as in [28]) and can hence be used to automate a 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 successively present related work on user group exploration frameworks, recommendation of exploration actions, and reinforcement learning for interactive data exploration.

2.1 User group exploration frameworks

User group exploration is a recent and dynamic research area in data management [27]. The most traditional way of exploring user groups is based on previous exploration frameworks [5, 6]. Most notably, `by-query` can be used to

select user groups that satisfy some 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*, [17, 37]. `by-example` exploration [26, 28] finds groups that are similar to/different from an input group. In [28], `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 statistics. Some actions admit as input a set of distributions and find groups with similar rating distributions [2]. Others are a variant of `by-example` and look for groups similar to, or different from, a given group in terms of their distributions [16]. `by-text` exploration actions [9] 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 [19, 20, 35, 38, 39, 40] 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 [7, 11]. One approach based on collaborative filtering [1, 12, 18, 24] uses previously collected query logs a dataset (SQL queries in [12, 18], and OLAP queries in [1, 24]) 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 works focus on the recommendation of SQL/OLAP queries while we are interested in more abstract operations that do not require IT expertise.

Another mode of interactive exploration, sometimes called data-driven approach [7, 4, 10, 15, 29, 32], 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, [15] 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 [4], sequence of actions within the same session [15, 32], or data returned by a query [10, 29]). 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 [25]. 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 [33] 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 [22] or the relevance of content items [3] based on the behavior 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, product rating) and an optional text x (e.g., publication titles/abstracts, product 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, hotel nightly price). 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 users in $\mathcal{U}$ associated with a set of attribute-value pairs $\langle a, v\rangle$ on attributes in $\mathcal{A}$, called $label(g)$. We denote the set of all groups as $\mathcal{G}$. We define $items(g)$ as the set of all 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 each label $\langle a, v\rangle$ in $label(g)$ as follows: if $a \in \mathcal{A}_U$ then $\langle a, v\rangle$ is true for u else ($a \in \mathcal{A}_I$) there exists $i \in items(g)$ such that $\langle a, v\rangle$ is true for i. Thus, $label(g) =$

$\{\langle gender, female\rangle, \langle location, CA\rangle, \langle venue, VLBDJ\rangle, \langle venue, SIGMOD\rangle\}$ represents a group of women researchers in California who published a VLDB Journal and a SIGMOD paper at least once.

EXAMPLE 1. *We consider Martin 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 find target users. Intuitively, an action should provide the ability to dive into a group or to expand a group, or to compare group members using any meaningful dimension. Therefore, we decided to adopt previously proposed exploration actions [27] that support those capabilities. We then unify the semantics of these actions to make them easily 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*:

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

The definitions of $relevance(\bullet)$ and $quality(\bullet)$ depend on the exploration action e. Our design is inspired by the approach developed in [33] to predict "interestingness" measures relevant to the next step of interactive exploration. The function $relevance(g, e)$ forces an action to return groups that are relevant to the input group g to ensure continuity in the exploration. The function $quality(\mathcal{G}_k, e)$ ensures that the collective set of k returned groups, $\mathcal{G}_k$, covers a wide range of exploration options. Both functions return a value between 0 and 1. The parameter σ is a relevance threshold where $\sigma = [0, 1]$.

We now define each exploration action used in our framework.

Exploration action explore-around. This action returns k diverse groups that overlap with an input group g_{in} [28]. Jaccard similarity is used to implement $relevance(\bullet)$, and diversity is used for $quality(\bullet)$:

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

$$quality(\mathcal{G}_k, \textsf{explore-around}) = diversity(\mathcal{G}_k) = \frac{|\bigcup_{g \in \mathcal{G}_k} g|}{\sum_{g \in \mathcal{G}_k} |g|} \quad (1)$$

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 [28]. This is akin to subgroup discovery [30]:

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

The function $relevance(g, \textsf{explore-within})$ 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 $quality(\bullet)$ as follows:

$$\begin{aligned} quality(\mathcal{G}_k, \textsf{explore-within}) &= coverage(g_{in}, \mathcal{G}_k) \\ &= |\cup_{g \in \mathcal{G}_k} (g \cap g_{in})| / |g_{in}| \end{aligned} \quad (2)$$

Exploration action by-facet. This action is a realization of faceted search in the group space [14]. 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, $relevance(\bullet)$ and $quality(\bullet)$ are defined as follows:

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

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

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

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:

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

The by-distribution action finds groups with score distributions similar to an input group g_{in} [2]. 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.

$$\begin{aligned} relevance(g, \textsf{by-distribution}) &= EMD(\tilde{g}, \tilde{g_{in}}) = \\ &\frac{min(work(\tilde{g}, \tilde{g_{in}}))}{min(\|\tilde{g}\|_2, \|\tilde{g_{in}}\|_2)} \end{aligned} \quad (3)$$

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 and associate a topic vector $\vec{x}$ to each tuple $\langle u, i, s, x\rangle \in \mathcal{D}$. In our work we use LDA where the corpus is the set of x 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 the ones whose Cosine similarity to g_{in}'s topic vector is higher than a threshold. The quality function is the diversity of the returned group labels.

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

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

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 which allows him to identify a new group of 38 researchers whose research covers "stream processing" and "data integration". The latter is 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 already 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. The example also illustrates several key aspects of the exploration: 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 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: $[\langle g_1, \{g_1\}, \textsf{explore-around}\rangle, \langle g_2, \mathcal{G}_{k2}, \textsf{explore-within}\rangle, \langle g_3, \mathcal{G}_{k3}, \textsf{by-topic}\rangle]$, 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: $\pi(s_i) = e_i$. 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. In our EDA environment, the aim of an analysis 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 target ones. 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 given user group. We refine the definition of group utility as follows:

$$\begin{aligned} g\_utility(g, \mathcal{U}_t) &= |g \cap \mathcal{U}_t| * \mathbb{1}[g \in \mathcal{G}^c_t] \\ \mathcal{G}^c_t &= \{g \in \mathcal{G} : \frac{|g \cap \mathcal{U}_t|}{|g|} > c\} \end{aligned} \tag{4}$$

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 $\gamma \in [0, 1]$:

$$s\_utility(\mathcal{S}, \mathcal{U}_t) = \sum_{(g_i, \mathcal{G}_{ki}, e_i) \in \mathcal{S}} \gamma^i g\_utility(g_i, \mathcal{U}_t \backslash \bigcup_{j<i} \{g_j \in \mathcal{G}^c_t\}) \tag{5}$$

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. Then, Section 4.2 presents our modeling using a Markov Decision Process (MDP) following which we reformulate our guided EDA problem for user groups 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 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.

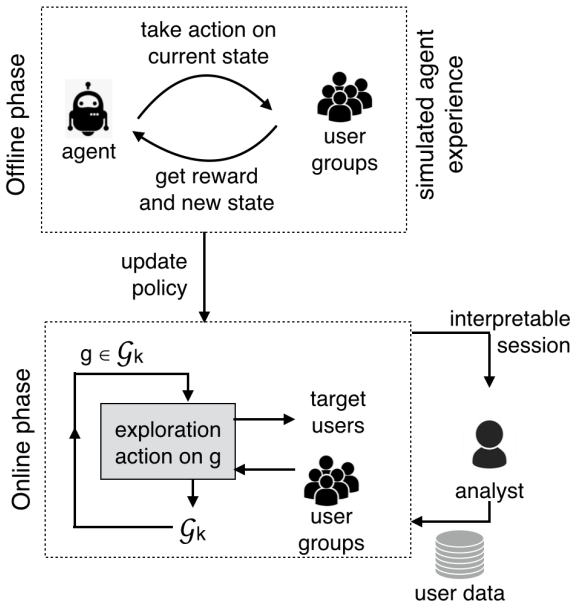


Figure 1: RL framework architecture

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} in S 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 5). 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 [25]. 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 [8]. 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 steps.

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 [34]:

$$V^*(s) = \max_a Q^*(s,e) =$$

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

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

4.4 RL framework

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

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 $\mathbf{f}(s,e)$. Our goal is to learn meaningful EDA policies that generate interpretable exploration sessions for human analysts. To enable that, we describe our states with a small set of domain-dependent features and use those features in our learning process. The features we use reflect the result of applying an exploration action to a state. They must enable learning how to choose between actions at each step. In Section 3.2, we specified relevance and quality functions that the actions maximize, such as overlap with input group (akin to coverage), and diversity. We expect the states generated by different actions to possess different values for these functions (e.g., explore-around aims to generate states with higher diversity, while explore-within aims to generate states with higher coverage). A set of features related to the relevance and quality functions ensures that values of $P(s_i|s_{i-1}, e_{i-1})$ depend on e_{i-1}, so it is possible to learn a policy that performs better than random. Section 5 (Table 2) 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 [28, 14, 2] and on several trial-and-error steps. 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 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. We rely on approximate control methods [34] 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 $\mathbf{w} \in \mathbb{R}^n$, where n is the number of state features and $n \ll |S|$. Given a state-action feature vector $\mathbf{f}(s,e)$, we define a linear approximation of $Q(s,e)$:

$$\hat{Q}(\mathbf{w}, s, e) = \mathbf{w}^T \mathbf{f}(s,e) \tag{6}$$

Algorithm 1: Policy learning

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

```
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)
        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
return agent
```

This approximation is essential as it still makes our representation data-driven but not strictly data-dependent. It blends together states with the same features in the same manner as grouping similar contexts is done in database exploration [25].

Learning procedure. To learn the approximation of the action-value function, we apply a common method, a semi-gradient method [34] based on stochastic gradient descent (SGD) minimization of mean squared error:

$$err(\mathbf{w}) = \sum_{s \in S, e \in E} P(s,e)(Q(s,e) - \hat{Q}(\mathbf{w}, s, e))^2 \tag{7}$$

where $P(s,e)$ denotes probabilities of the state-action pairs in the agent-environment interaction. During 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:

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \alpha(Q(s_i, e_i) - \hat{Q}(\mathbf{w}_i, s_i, e_i))\nabla \hat{Q}(\mathbf{w}_i, s_i, e_i) \tag{8}$$

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:

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \alpha(Q_i - \hat{Q}(\mathbf{w}_i, s_i, e_i))\nabla \hat{Q}(\mathbf{w}_i, s_i, e_i) \tag{9}$$

The estimator we use is proposed in the semi-gradient SARSA algorithm [34], 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:

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \alpha(R_{i+1} + \gamma \mathbf{w}_i^T \mathbf{f}(s_{i+1}, e_{i+1}) - \mathbf{w}_i^T \mathbf{f}(s_i, e_i))\mathbf{f}(s_i, e_i) \tag{10}$$

These updates guarantee to converge to a local minimum with a sufficiently small learning rate α and a fixed $P(s,e)$ distribution [34].

Algorithm 2: Policy recommendation

Input: $agent, g_0$
Output: *exploration action recommendation* e_{rec}

```
1  s_i ← (g_0, {g_0})
2  e_i is chosen by a human analyst
3  while not end_of_session do
4  |  G_{ki+1}, r ← explore(s_i, e_i)
5  |  s_{i+1} ← (g_{i+1}, G_{ki+1})
6  |  e_rec ← agent.get_exploration_action(s_{i+1})
7  |  e_{i+1} is chosen by a human analyst with
   |    recommendation e_rec
8  |  s_i ← s_{i+1}
9  |  e_i ← e_{i+1}
10 end
```

Learning algorithm. We apply the learning procedure in an offline phase (Algorithm 1) and then recommend the best exploration policy in the online phase (Algorithm 2). Algorithm 1 shows the process of the agent-environment interaction during policy learning. The algorithm iterates until the *end_of_learning* is *true* which can be caused by a time limit or a limit on the proportion of target users found (more details are provided 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 a probability of 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 4, 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 $\mathcal{G}_t$, otherwise 0. The function *update_weights*() is responsible for learning the weights through semi-gradient updates as depicted in Equation 10 (line 9). Finally the agent with the learned set of weights $\mathbf{w}$ is returned (line 14).

Given the updated agent, Algorithm 2 describes the process of policy recommendation. At each step, the algorithm picks the best exploration action e_{rec} as a function of the current state $\langle g, \mathcal{G}_k \rangle$ (line 6).

5. EXPERIMENTS

We follow two distinct goals in our experiments. We first seek to validate the use of RL for EDA on user data, and then we examine the utility of our learned policies on real user datasets.

5.1 Datasets

We test our system on DM-Authors, a dataset we built from DBLP.[2] DM-Authors includes 1,860 researchers with a total of 200,000 publications in data management venues between 2000 and 2018: WWW, VLDB, SIGMOD, ICDE, SIGMOD, RecSys, EDBT, DEXA, WebDB, and HILDA. We crawled profiles of researchers from DBLP and added demographic attributes.

[2] https://dblp.uni-trier.de/db/

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".[3]

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.

Attribute $a \in \mathcal{A}_\mathcal{U}$	**Description**	**Values** $v \in dom(a)$
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 to 8 years), *junior* (9 to 12), *senior* (13 to 15), *highly senior* (16 to 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 to 1.47), *very active* (1.48 to 2.48), *productive* (2.49 to 3.71), *very productive* (3.72 to 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.[4]	*male*, *female*

Table 1: Demographic attributes in DM-Authors.

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 [36]. 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

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

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.

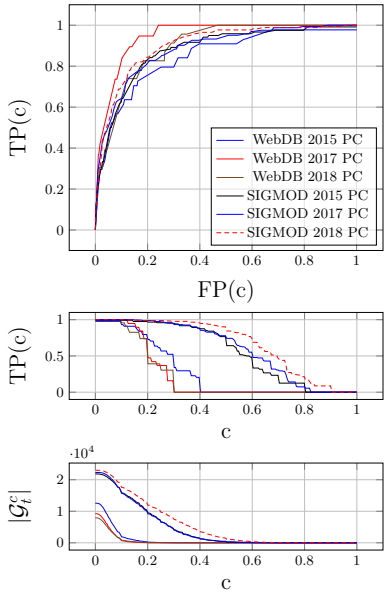


Figure 2: Quality of $\mathcal{G}$ with respect to exploration targets.

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 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. The engineering of these features is therefore strongly guided.

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:

1. `OLDEST`: the train set is the 2015 PC of the same venue.
2. `LAST`: the train set is the PC of the previous year's venue, i.e., 2017.
3. `TRANSFER`: 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 11).

$$TPR(\mathcal{G}_t^c, c) = \frac{|\bigcup_{g \in \mathcal{G}_t^c} (\mathcal{U}_t \cap g)|}{|\mathcal{U}_t|};\ FPR(\mathcal{G}_t^c, c) = \frac{|\bigcup_{g \in \mathcal{G}_t^c} (g \setminus \mathcal{U}_t)|}{|\mathcal{U} \setminus \mathcal{U}_t|} \tag{11}$$

The true positive rate $TPR(\bullet)$ measures the fraction of target users $\mathcal{U}_t$ found in target groups $\mathcal{G}_t^c$. Also the false positive rate $FPR(\bullet)$ 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 11) 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 $\mathcal{G}_t$.

In the offline simulation 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 as . And if there are no such groups, the choice of g is random.

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 has a time limit of 100ms. The number of iterations is limited to 200. All our results are averages of 10 runs of the offline learning.

Exploration actions. We make use of all the exploration actions described in Section 3.2 with different default 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 an additional exploration action undo to enable returning to the previous state if no target user has been observed in the current state. We use LDA to generate the topic vectors that are used in by-topic. LDA generates 10 topics.

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

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

	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

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

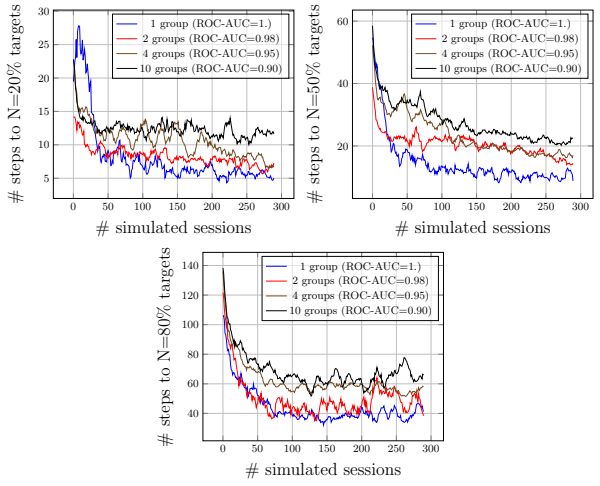


Figure 3: Learning curve for the synthetic experiments with different scattering levels: 20 target users are scattered equally over 1, 2, 4 or 10 groups from the group set.

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.

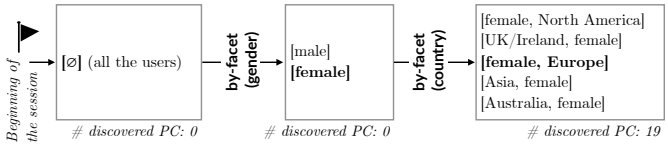


Figure 4: Simulated session with a policy trained on the synthetic task.

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 learned policies

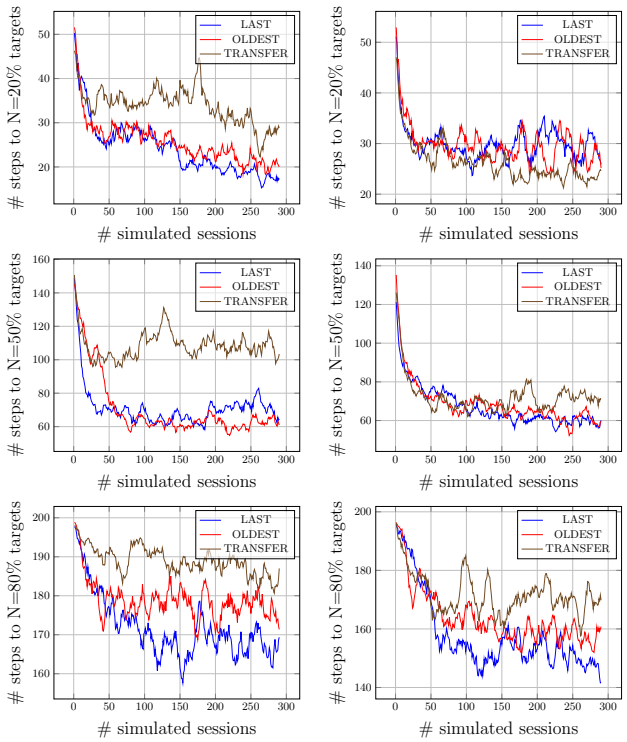


Figure 5: Learning curves of simulated sessions that run until discovering 20% (top), 50% (middle), 80% (bottom) of WebDB (left) and SIGMOD (right) PCs. The curves show the average number of steps that the agent takes to reach the exploration goal after learning in a number of simulated sessions displayed 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.

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 5 illustrates the learning curves.

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 training curve and the test curve increase together, which means that the policy based on the state-action features is useful for similar exploration goals, not only for the goal used in 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 over-fitting 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.

From this experiment we can conclude that different exploration tasks require different policies, but it is also possible to transfer policies between similar tasks. We also observe that the difference in performance between learning variants increases with ν. In other words, 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 to capture each state and guide 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 6. 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 7) and the number of discovered targets (Figure 8). 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.

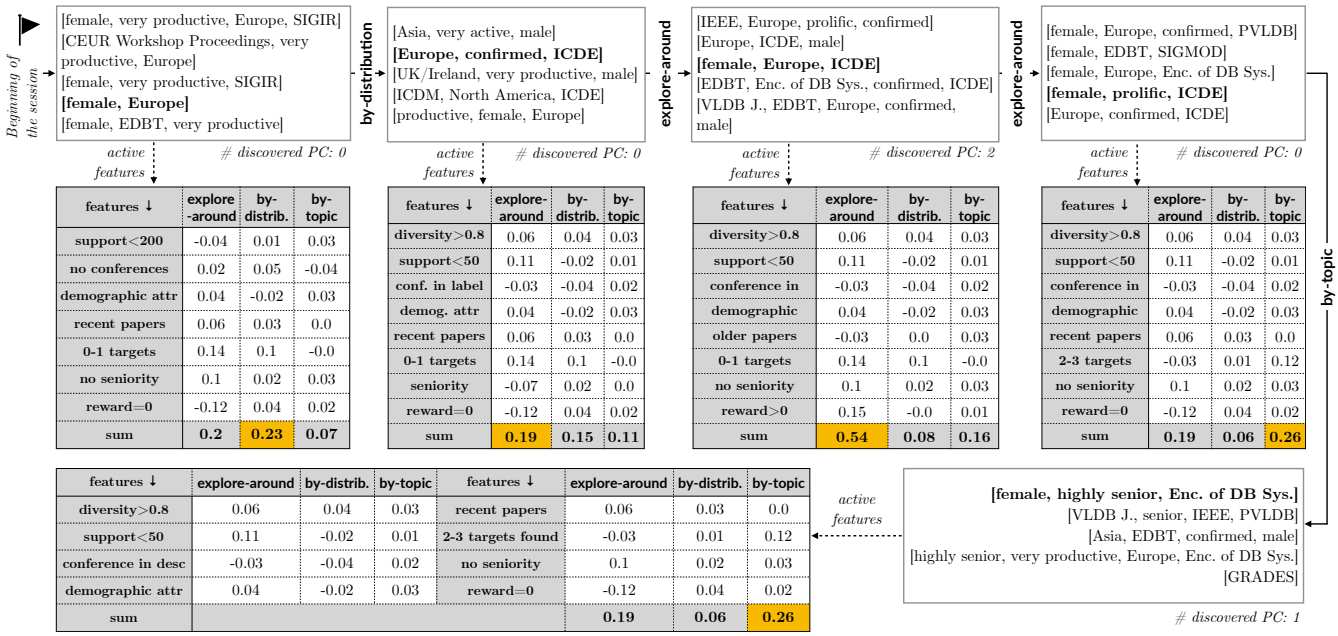
One can notice similar patterns in the policies: when increasing the number of targets discovered so far, the likelihood of using explore-around decreases (Figure 8). As it becomes less probable to find new targets in the groups overlapping with an input group (via other actions than explore-around), 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 for smaller groups. A possible explanation is the larger ratio of target users to all users for the SIGMOD PC, as by-distributioncan 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.

This last experiment allows us to validate that our exploration actions and offline policy learning naturally capture what a human analyst does manually [28].

6. CONCLUSION

In this paper, we examined the applicability of machine learning techniques to EDA on user data where the task is to find a set of users. We first proposed a unified formalization of the guided EDA problem that leverages a wide



features ↓	explore-around	by-distrib.	by-topic
support<200	-0.04	0.01	0.03
no conferences	0.02	0.05	-0.04
demographic attr	0.04	-0.02	0.03
recent papers	0.06	0.03	0.0
0-1 targets	0.14	0.1	-0.0
no seniority	0.1	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
conf. in label	-0.03	-0.04	0.02
demog. attr	0.04	-0.02	0.03
recent papers	0.06	0.03	0.0
0-1 targets	0.14	0.1	-0.0
seniority	-0.07	0.02	0.0
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	-0.03	-0.04	0.02
demographic	0.04	-0.02	0.03
older papers	-0.03	0.0	0.03
0-1 targets	0.14	0.1	-0.0
no seniority	0.1	0.02	0.03
reward>0	0.15	-0.0	0.01
sum	**0.54**	**0.08**	**0.16**

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	-0.03	-0.04	0.02
demographic	0.04	-0.02	0.03
recent papers	0.06	0.03	0.0
2-3 targets	-0.03	0.01	0.12
no seniority	0.1	0.02	0.03
reward=0	-0.12	0.04	0.02
sum	**0.19**	**0.06**	**0.26**

features ↓	explore-around	by-distrib.	by-topic	features ↓	explore-around	by-distrib.	by-topic
diversity>0.8	0.06	0.04	0.03	recent papers	0.06	0.03	0.0
support<50	0.11	-0.02	0.01	2-3 targets found	-0.03	0.01	0.12
conference in desc	-0.03	-0.04	0.02	no seniority	0.1	0.02	0.03
demographic attr	0.04	-0.02	0.03	reward=0	-0.12	0.04	0.02
sum					**0.19**	**0.06**	**0.26**

Figure 6: Simulated session example. For each step in the simulated session, the labels of the k groups and their active features are shown. The agent chooses actions with the highest sum of weights (highlighted).

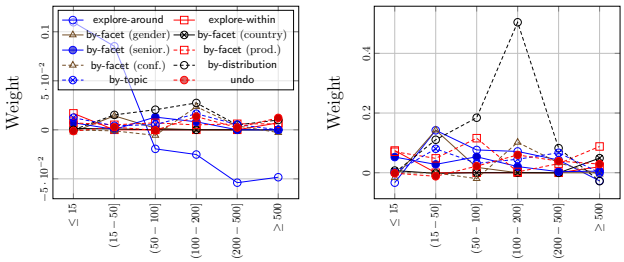


Figure 7: Learned weights for the feature describing the size of the input group (x-axis), trained on WebDB 2017 (left) and SIGMOD 2017 (right). Each line corresponds to one exploration action. For a given exploration state, only one of the binary features displayed here takes value 1, others are 0.

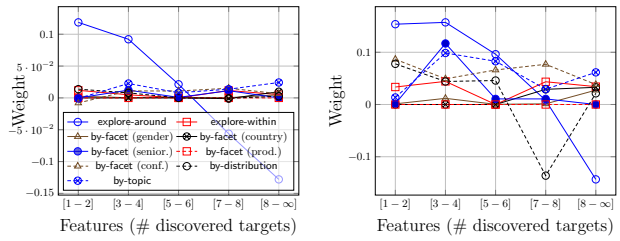


Figure 8: Learned weights for the features showing the number of targets discovered so far in the exploration, trained on WebDB 2017 (left) and SIGMOD 2017 (right).

range of user data exploration actions. We developed a solution based on a Markov Decision Process and reinforcement learning to learn an exploration policy. We conducted experiments on real datasets from DBLP where the purpose was to build the program committees of WebDB and SIGMOD. Our results validate the utility of different learning variants to find an exploration policy for guided PC formation.

Our work opens many new questions at the intersection of data management and machine learning. A deeper experimental validation of several choices we made is warranted. In particular, we need to test the robustness of learned strategies with different state-action features. We also need to investigate the definition of different exploration policy utilities and corresponding reward functions. In the near future, we would like to investigate the utility of text-based exploration actions for new tasks such as finding products in retail. Such actions will help us explore user reviews and tags in addition to the item space. We believe this will raise new challenges in the design of the feature space and the reward function.

7. REFERENCES

[1] J. Aligon, E. Gallinucci, M. Golfarelli, P. Marcel, and S. Rizzi. A collaborative filtering approach for recommending olap sessions. *Decision Support Systems*, 69:20–30, 2015.

[2] S. Amer-Yahia, S. Kleisarchaki, N. K. Kolloju, L. V. Lakshmanan, and R. H. Zamar. Exploring rated datasets with rating maps. In *WWW*, 2017.

[3] I. Arapakis, M. Lalmas, and G. Valkanas. Understanding within-content engagement through pattern analysis of mouse gestures. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1439–1448. ACM, 2014.

[4] C. Bolchini, E. Quintarelli, and L. Tanca. Context support for designing analytical queries. In *Methodologies and Technologies for Networked Enterprises - ArtDeco:*

Adaptive Infrastructures for Decentralised Organisations, pages 277–289. 2012.

[5] F. Bonchi, F. Giannotti, C. Lucchese, S. Orlando, R. Perego, and R. Trasarti. Conquest: a constraint-based querying system for exploratory pattern discovery. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pages 159–159. IEEE, 2006.

[6] F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi. Exante: Anticipated data reduction in constrained pattern mining. In *PKDD*, volume 2838, pages 59–70. Springer, 2003.

[7] G. Chatzopoulou, M. Eirinaki, and N. Polyzotis. Query recommendations for interactive database exploration. In *International Conference on Scientific and Statistical Database Management*, pages 3–18. Springer, 2009.

[8] J. Clark and D. Amodei. Faulty reward functions in the wild. https://blog.openai.com/faulty-reward-functions/, 2016.

[9] M. Das, S. Thirumuruganathan, S. Amer-Yahia, G. Das, and C. Yu. Who tags what?: an analysis framework. *Proceedings of the VLDB Endowment*, 5(11):1567–1578, 2012.

[10] M. Drosou and E. Pitoura. Ymaldb: exploring relational databases via result-driven recommendations. *The VLDB JournalThe International Journal on Very Large Data Bases*, 22(6):849–874, 2013.

[11] R. Ebenstein, N. Kamat, and A. Nandi. Fluxquery: An execution framework for highly interactive query workloads. In *Proceedings of the 2016 International Conference on Management of Data*, pages 1333–1345. ACM, 2016.

[12] M. Eirinaki, S. Abraham, N. Polyzotis, and N. Shaikh. Querie: Collaborative database exploration. *IEEE Transactions on knowledge and data engineering*, 26(7):1778–1790, 2014.

[13] O. B. El, T. Milo, and A. Somech. ATENA: an autonomous system for data exploration based on deep reinforcement learning. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, pages 2873–2876, 2019.

[14] M. A. Hearst. Clustering versus faceted categories for information exploration. *Commun. ACM*, 49(4):59–61, Apr. 2006.

[15] M. Joglekar, H. Garcia-Molina, and A. G. Parameswaran. Interactive data exploration with smart drill-down. *IEEE Trans. Knowl. Data Eng.*, 31(1):46–60, 2019.

[16] M. Kahng, S. B. Navathe, J. T. Stasko, and D. H. P. Chau. Interactive browsing and navigation in relational databases. *Proceedings of the VLDB Endowment*, 9(12):1017–1028, 2016.

[17] N. Kamat, P. Jayachandran, K. Tunga, and A. Nandi. Distributed and interactive cube exploration. In *2014 IEEE 30th International Conference on Data Engineering*, pages 472–483. IEEE, 2014.

[18] N. Khoussainova, Y. Kwon, M. Balazinska, and D. Suciu. Snipsuggest: Context-aware autocompletion for sql. *Proceedings of the VLDB Endowment*, 4(1):22–33, 2010.

[19] P. Lee, L. V. Lakshmanan, and E. E. Milios. Incremental cluster evolution tracking from highly dynamic network data. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 3–14. IEEE, 2014.

[20] F. Lemmerich, M. Becker, P. Singer, D. Helic, A. Hotho, and M. Strohmaier. Mining subgroups with exceptional transition behavior. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 965–974. ACM, 2016.

[21] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[22] L. Li, H. Deng, Y. He, A. Dong, Y. Chang, and H. Zha. Behavior driven topic transition for search task identification. In *Proceedings of the 25th International Conference on World Wide Web*, pages 555–565. International World Wide Web Conferences Steering Committee, 2016.

[23] S. Li, A. Karatzoglou, and C. Gentile. Collaborative filtering bandits. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 539–548. ACM, 2016.

[24] P. Marcel and E. Negre. A survey of query recommendation techniques for data warehouse exploration. In *EDA*, pages 119–134, 2011.

[25] T. Milo and A. Somech. Next-step suggestions for modern interactive data analysis platforms. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 576–585. ACM, 2018.

[26] D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas. New trends on exploratory methods for data analytics. *Proceedings of the VLDB Endowment*, 10(12):1977–1980, 2017.

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

[28] B. Omidvar-Tehrani, S. Amer-Yahia, and A. Termier. Interactive user group analysis. In *CIKM*, pages 403–412. ACM, 2015.

[29] S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of olap data cubes. In *International Conference on Extending Database Technology*, pages 168–182. Springer, 1998.

[30] T. Scheffer and S. Wrobel. A Sequential Sampling Algorithm for a General Class of Utility Criteria. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2000.

[31] Y. Shen and H. Jin. Epicrec: Towards practical differentially private framework for personalized recommendation. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 180–191. ACM, 2016.

[32] M. Singh, M. J. Cafarella, and H. Jagadish. Dbexplorer: Exploratory search in databases. In *EDBT*, pages 89–100, 2016.

[33] A. Somech, T. Milo, and C. Ozeri. Predicting "what is interesting" by mining interactive-data-analysis session logs. In *Advances in Database Technology - 22nd International Conference on Extending Database Technology, EDBT 2019, Lisbon, Portugal, March 26-29, 2019*, pages 456–467, 2019.

[34] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[35] M. Tsytsarau, S. Amer-Yahia, and T. Palpanas. Efficient sentiment correlation for large-scale demographics. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, pages 253–264, 2013.

[36] T. Uno, M. Kiyomi, and H. Arimura. Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In *Fimi*, volume 126, 2004.

[37] N. Yan, C. Li, S. B. Roy, R. Ramegowda, and G. Das. Facetedpedia: enabling query-dependent faceted search for wikipedia. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1927–1928. ACM, 2010.

[38] J. Yang, J. McAuley, J. Leskovec, P. LePendu, and N. Shah. Finding progression stages in time-evolving event sequences. In *Proceedings of the 23rd international conference on World wide web*, pages 783–794. ACM, 2014.

[39] Y. Yang, D. Yan, H. Wu, J. Cheng, S. Zhou, and J. C. Lui. Diversified temporal subgraph pattern mining. In *KDD*, pages 1965–1974, 2016.

[40] T. Zhang, P. Cui, C. Faloutsos, Y. Lu, H. Ye, W. Zhu, and S. Yang. Come-and-go patterns of group evolution: A dynamic model. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1355–1364. ACM, 2016.