SubDEx: Exploring Ratings in Subjective Databases Technical Report

Sihem Amer-Yahia
CNRS, Univ. Grenoble Alpes
sihem.amer-yahia@univ-grenoble-alpes.fr

Tova Milo
Tel Aviv University
milo@post.tau.ac.il

Brit Youngmann
Tel Aviv University
brity@mail.tau.ac.il

***Abstract*—We demonstrate SubDEx, a dedicated framework for Subjective Data Exploration (SDE). SubDEx enables the joint exploration of items, people, and people's opinions on items, in a guided multi-step process where each step aggregates the most *useful* and *diverse* trends in the form of *rating maps*. Because of the large search space of possible rating maps, we leverage pruning strategies to enable interactive running times. We demonstrate the need for a dedicated SDE framework and the effectiveness and efficiency of our approach, by interacting with the ICDE'21 participants who will act as data analysts. A video of SubDEx is available at [1]**

I. Introduction

Subjective data is characterized by a mix of facts and opinions. With the proliferation of user-generated content, subjective databases have grown in size [2]. The valuable information they contain is virtually infinite and satisfies various needs. Yet, as of today, dedicated tools for Subjective Data Exploration (SDE) are lacking.

As in general-purpose Exploratory Data Analysis (EDA), SDE requires iterative data filtering and generalization. For instance, a social scientist examining restaurants in Yelp, may benefit from seeing aggregated ratings on a certain cuisine in some neighborhood by reviewers in a certain age range, followed by request to cover additional neighborhoods. Like in EDA, SDE users need guidance as they seldom know precisely what they are looking for and may have only partial knowledge of the underlying data. But, in addition to the common EDA guidance requirements, SDE must additionally satisfy specific needs that occur when exploring a mix of facts and opinions. Let us consider an example.

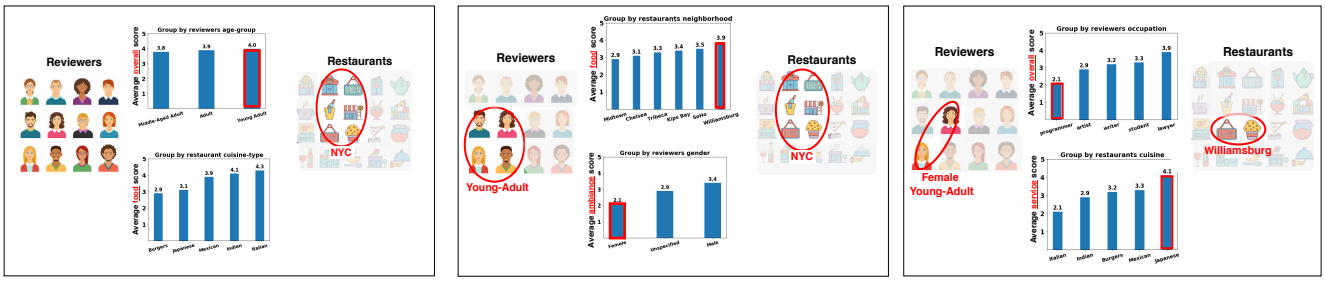
Mary is a social scientist who would benefit from the ability to extract insights on restaurants in New York City. Figure 1 summarizes a 3-step exploration of those restaurants and their reviewers. In Step I, Mary examines the reviewers' overall ratings and sees no significant difference between age groups (upper histogram). As a young adult, her next operation is to look deeper into that group (Step II). She discovers that they gave the highest ratings for food to restaurants in Williamsburg (upper histogram). She also finds that on average, young female adults have given the lowest ambiance rating (lower histogram). In Step III, Mary dives deeper into the ratings of young female adults and finds that programmers among them provided the lowest overall ratings (upper histogram). She also sees that those reviewers gave the highest service ratings to Japanese restaurants (lower histogram). As illustrated. if chosen properly, in only a few steps, Mary can obtain detailed insights on people's opinions on New York City restaurants.

Mary's example illustrates two key needs that characterize SDE: the need to select (simultaneously) subsets and supersets of items and reviewers whose aggregated ratings demonstrate *useful and diverse facets* of reviewers' opinions, and the need to explore *different rating dimensions*, e.g., food vs. service for restaurants. In this work we present SubDEx, a dedicated SDE framework. There are three key challenges in building such a system. First, the system should cater to the above-mentioned needs (challenge **C1**). Namely, it should display to users aggregated ratings that demonstrate useful and diverse facets of the data, while aggregating reviewers and items by different rating dimensions. Second, as in modern EDA tools [3], [4], the system should provide to users some guidance on the next operation to perform, to discover interesting trends in the data (challenge **C2**). Last, the system should enable interactive running times (challenge **C3**).

To address **C1**, SubDEx provides the ability to apply, at each step, a filtering or generalization operation on the items and reviewers of interest. It then displays, alongside the resulting rating records, a set of *k rating maps* [5] (see Figure 4(a)). Rating maps are histograms that provide a bird's-eye view of ratings by some reviewers for some items. The rating maps displayed at each step are chosen to be *useful* and *diverse*. Our notion of utility generalizes previous interestingness measures [6], whereas our notion of diversity ensures that different facets of the data are revealed. To ensure that the selected rating maps depict different rating dimensions, we use weighted utility scores where the weights reflect the number of times a rating dimension has been previously shown.

To address **C2**, SubDEx offers two exploration modes: *Recommendation-Powered*, and *Fully-Automated*. In the first mode, the system presents the current k most useful and diverse rating maps at each step, and recommends o next-step operations based on the utility and diversity of the rating maps they generate (see Figure 4). The user can choose one recommendation or perform an operation of her own. This was the case for Mary. The second *Fully-Automated* mode relieves the user from choosing an operation, and generates a fixed-size exploration path, by applying the top-1 operation at each step.

To address **C3**, SubDEx applies pruning optimizations that estimate the weighted utility score for each rating map based on sampling techniques and prune low utility ones. To enable that, we adapted highly efficient sharing and pruning techniques [7] for identifying high-utility rating maps and reduce computational costs.



Step I Step II Step III

Fig. 1: **Example of a three-step exploration. The user iteratively examines subsets of the reviewer and item tables. Links between selected reviewer and item groups are aggregated as rating maps, showing "interesting" trends in the data.**

Demonstration Overview: We demonstrate the operation of SubDEx over multiple real-world subjective datasets. Our demonstration illustrates real-life scenarios where a data analyst attempts to identify special data characteristics. The audience will play the role of data analysts, using one (or more) of SubDEx exploration modes. Then, the audience will explore statistics describing the results of other participants, enabling to observe the effect of guidance in SDE. Last, the audience will be allowed to look "under the hood", examining the efficiency of our solution.

Related Work: Subjective data analysis is an emerging research field [2]. Such data is widely used in web applications, online rating systems, and social sciences [8]. SDE can be used for large-scale population studies whose purpose is to extract trends and insights on the users/items, or for extracting recommendations [5]. To the best of our knowledge, SubDEx is the first system dedicated to SDE.

Exploratory Data Analysis (EDA) is an essential task for data scientists, with the goal of extracting insights from a dataset. Guiding users in performing EDA is a well-studied task [3], [4]. While general-purpose EDA tools could also be used for SDE, as mentioned, an SDE tool must cater to additional needs that require tailored solutions. Auto-generation of interesting views for a dataset has been studied extensively [9]. A common approach that we follow, is to use heuristic measures of interestingness [8], searching the space of all views, and returning the most interesting ones [7]. Multiple techniques to enable scalable visualization have been proposed [10]. Here we leverage pruning optimizations to identify low-utility rating maps, based on an adaptation of techniques presented in [7].

II. Data Model and SDE

A. Data Model

We consider a special type of database, called a subjective database [2], which includes both objective and subjective attributes. Specifically, we model our database $\mathcal{D}$ as a triple $\langle \mathcal{I}, \mathcal{U}, \mathcal{R} \rangle$, representing the sets of items, users (which we refer to as reviewers), and rating records, resp. Items and reviewers are associated with objective attributes, such as a restaurant address, and a reviewer occupation. The rating records have subjective attributes that reflect the ratings assigned by reviewers to items. Such ratings can be extracted from reviews [2] or directly provided by reviewers. In practice, there could be one table per rating dimension or a single table where each rating dimension is a column. For instance, a reviewer may rate a restaurant on several dimensions: food, service, and ambiance. In that case, there will be one attribute per rating dimension in the rating table. Formally, let $r_1, \ldots, r_t$ denote the rating dimensions. Each rating record $r \in \mathcal{R}$ is itself a tuple $\langle i, u, s_1, \ldots, s_t \rangle$, where $i \in \mathcal{I}$, $u \in \mathcal{U}$, and $s_j, j \in [1, t]$ is the numerical rating score that reviewer u assigned to item i for the j-th rating dimension. The values of s are application-dependent and do not affect our model.

$\mathcal{I}$ is associated with a set of objective attributes, denoted by $\mathcal{I}_A = \{ia_1, ia_2, \ldots\}$, and each item $i \in \mathcal{I}$ is a tuple with $\mathcal{I}_A$ as its schema. In other words, $i = \langle iv_1, iv_2, \ldots \rangle$, where each iv_j is a value for attribute ia_j. The value itself may be an atomic value or of complex type (e.g., a set of values). For example, for the restaurant *Joe's Farm Grill* in Figure **??**, the set of attribute values are $\langle$`{Burgers, Barbeque}`, `North_Carolina`, `Charlotte`, `77474`$\rangle$ for the schema $\langle$`cuisine`, `state`, `city`, `zip`$\rangle$. The attribute `cuisine` is multi-valued. Similarly, we have the schema $\mathcal{U}_A = \{ua_1, ua_2, \ldots\}$ for reviewers, i.e., $u = \langle uv_1, uv_2, \ldots \rangle \in \mathcal{U}$, where each uv_j is a value for attribute ua_j.

We now define the notions of reviewer, item and rating groups.

Definition 2.1 (Reviewer/Item Group): A reviewer group g_U (resp., item group g_I) is a set of reviewers (resp., items) that share the same values for a set of objective attributes which define its description $\{\langle a_1, v_1 \rangle, \langle a_2, v_2 \rangle, \ldots\}$ where each $a_i \in \mathcal{U}_A$ for a reviewer group (resp., $\mathcal{I}_A$ for an item group).

For example, g_U= {$\langle$`gender`, `female`$\rangle$, $\langle$`age_group`, `young`$\rangle$} contains all young female reviewers, and g_I= {$\langle$`cuisine`, `Italian`$\rangle$, $\langle$`state`, `NY`$\rangle$} contains the set of all Italian restaurants in New York.

Definition 2.2 (Rating Group): Given reviewer and item groups g_U and g_I, a rating group g_R for g_U and g_I is defined as the group of all rating records $r = \langle u, i, s_1, \ldots, s_t \rangle$ s.t. $u \in g_U$ and $i \in g_I$.

A rating group is captured by a set of attribute value pairs shared among reviewers and items, and can thus be interpreted as a predicate on the rating table.

B. SDE Operations and Rating Maps

An SDE process starts when a user loads a dataset to an analysis UI. She then executes a series of filtering/generalization operations. After each operation, she examines the results and decides to execute or not a new operation.

1) Exploration Operations: In each exploration step, the user examines a rating group g_R, defined by a reviewer group g_U and an item group g_I. To move to the next step, the user performs a filtering/generalization operation q on g_U, on g_I, or on both (i.e., perform two operations *simultaneously*). We abuse the notation and refer to an operation as q. An operation may add, remove, or change the value of the selection criteria of a group, correspondingly generating a new rating group. An operation can be expressed as a standard SQL query. For example, the user may FILTER the reviewers table by 'occupation' = 'student', or the restaurants table by 'cuisine' = 'Indian'. The execution of an operation generates a new rating group to be displayed to the user.

2) Rating Maps: To provide a bird's eye view of the ratings in a group g_R, we use *rating maps* [5] - histograms that aggregate ratings in g_R for some rating dimension using some item/reviewer attributes. Previous work has shown that such histograms are an adequate means of understanding rated datasets [11]. We now define the notions of rating distributions and rating maps.

Definition 2.3 (Rating Distribution [5]): The rating distribution of g_R for a rating dimension r_i, denoted by $dist(g_R, r_i)$, is a probability distribution $dist(g_R, r_i)$=$[w_1, \ldots, w_m]$, where the rating scale is $\{1, ..., m\}$, and w_j is the number of rating records with value j for the rating dimension r_i in g_R.

Definition 2.4 (Rating Map [5]): A rating map of a rating group g_R for a rating dimension r_i is a set of (subgroup,rating distribution) pairs: $rm(g_R, r_i) = (\langle g_1, dist(g_1, r_i)\rangle, \ldots, \langle g_k, dist(g_k, r_i)\rangle)$, where $g_R = \bigcup_{j=1}^{k} g_j$ and all subgroups are disjoint. The rating map also associates to each subgroup $g_j \in g_R$ an aggregated score.

We use average in this work. Other aggregations could be used such as the highest probability for the rating dimension r_i. To simplify the notation, we use $rm_{r_i}^{g_R}$ to denote $rm(g_R, r_i)$, and omit r_i and g_R whenever it is clear from the context.

Example 2.5: We assume a rating group g_R containing the ratings of young reviewers for restaurants in New York city: g_R contains rating records for items in g_I= {⟨`city`, `NYC`⟩} and reviewers in g_U= {⟨`age_group`, `young`⟩. The top two tables in Figure 2 show two example rating maps that correspond to the ones displayed in Figure 1, Step *II*. The first rating map is obtained by partitioning g_R on neighborhood. It associates to each subgroup its rating distribution for food, and the average score. The second one partitions g_R by gender, resulting in 3 subgroups: males, unspecified and females. It aggregates each subgroup by ambiance.

W.l.o.g and to simplify exposition, we assume that a rating map *rm* partitions g_R using solely one reviewer or item attribute. Thus, a rating map can be seen as the result of a GroupBy operation over g_R, followed by an aggregation function (average in this work) to assign a single rating score to each subgroup.

3) Utility of Rating Maps: In each exploration step, the user sees a set of k rating maps. This naturally raises the question of how to select this set. As argued in Section **??**, an SDE tool must address two novel needs: the need to select (simultaneously) subsets of items and reviewers whose aggregated ratings demonstrate *useful and diverse facets* of

rm: **GroupBy *neighborhood*, aggregated by *food* score**

No.	city	# of records	rating distribution	avg. score
1	Williamsburg	16	{1 : 1, 2 : 2, 3 : 1, 4 : 5, 5 : 7}	3.9
2	SoHo	20	{1 : 3, 2 : 3, 3 : 2, 4 : 5, 5 : 7}	3.5
3	Kips Bay	12	{1 : 2, 2 : 2, 3 : 2, 4 : 1, 5 : 5}	3.4
4	Tribeca	12	{1 : 3, 2 : 1, 3 : 2, 4 : 1, 5 : 5}	3.3
5	Chelsea	20	{1 : 3, 2 : 1, 3 : 9, 4 : 5, 5 : 2}	3.1
6	Midtown	20	{1 : 3, 2 : 3, 3 : 9, 4 : 3, 5 : 2}	2.9

rm′: **GroupBy *gender*, aggregated by *ambiance* score**

No.	gender	# of records	rating distribution	avg. score
1	Male	35	{1 : 5, 2 : 6, 3 : 4, 4 : 9, 5 : 11}	3.4
2	Unspecified	30	{1 : 5, 2 : 8, 3 : 7, 4 : 5, 5 : 5}	2.9
3	Female	35	{1 : 14, 2 : 10, 3 : 5, 4 : 5, 5 : 1}	2.1

Interestingness scores

No.	conciseness	agreement	self peculiarity
rm	16.6	0.74	0.21
rm′	33.3	0.76	0.27

Fig. 2: Example of two rating maps, and their associated interestingness scores.

the links between items and reviewers (**N1**), and the need to explore *different rating dimensions* (**N2**). To address need **N1**, we ensure that high-utility and diverse rating maps are selected in each exploration step. To address **N2**, we ensure that rating maps of different rating dimensions are chosen.

To define the utility of a rating map, we generalize commonly used interestingness measures for data exploration [6], [8]. We present next an intuitive definition of the used interestingness criteria. Formal definitions of the measures used in our prototype implementation are provided in Section III-A.

Conciseness. The conciseness score of a rating map, $Conc(rm)$, is a function of the number of subgroups in *rm*. It favors rating maps containing a small, human-readable number of subgroups that summarizes a large number of records in g_R.

Agreement. The agreement score of a rating map, $Agr(rm)$, conveys that each subgroup in g_R contains reviewers who agree among themselves on items for the examined rating dimension [12].

Peculiarity This measure ranks a rating map higher if its rating distribution demonstrates a difference from some reference rating distribution. To capture that, we consider two peculiarity scores. One measures the peculiarity of a rating map w.r.t. itself, denoted as $Pec_{self}(rm)$, examining the peculiarity of each individual subgroup within it, compared with the rating distribution of the entire group. The second, $Pec_{global}(rm)$, measures the peculiarity of a rating map w.r.t. previously displayed rating maps (global). It captures the ability of a rating map to show a new facet of the data that the user has not explored in previous steps.

As the values of interestingness measures are on different scales, we normalize them as proposed in [6]. This process consists of two steps applied to each measure: (1) we apply a Box-Cox [13] power transformation to produce a normal-distribution; (2) we calculate the mean and standard deviation of the obtained distribution and a z-score standardization [14] to adjust the value ranges.

We now have all the ingredients needed to define our utility score. The utility of a rating map *rm* at a given step, where the user has seen a set of rating maps RM, is denoted by $u(rm, RM)$, and is defined as the maximal score that best

captures its "interestingness":

$$u(rm, RM) := max(Conc(rm), Agr(rm), Pec_{\text{self}}(rm), Pec_{\text{global}}(rm, RM))$$

To address need **N2**, we introduce the refined notion of *dimension-weighted* (DW) utility score of a rating map.

Let RM denote the set of rating maps seen by the user, where $|RM|=m$. Assume that the rating dimensions are $r_1, r_2, \ldots, r_t$, and that, so far, the number of rating maps aggregated by dimension r_i, $i \in [1,t]$, is m_{r_i}. Namely, $m = \sum_{j=1}^{t} m_{r_j}$. Intuitively, the DW utility score of a rating map rm_{r_i} is a combination of its utility and a weight reflecting how important it is to promote dimension r_i. Rating dimensions that have been rarely selected would be promoted at the expense of those that have been frequently selected.

The DW utility score of a rating map rm_{r_i}, where RM is the set of rating maps seen by the user, is defined as:

$$\widehat{u}(rm_{r_i}, RM) := (1 - \frac{m_{r_i}}{m}) \cdot u(rm_{r_i}, RM) \quad (1)$$

Example 2.6: Let r_1 be the overall rating score, r_2 is the food score, r_3 is the service score, and r_4 is the ambiance score. Assume that the number of previously seen rating maps is $m=10$, where $m_{r_1}=3, m_{r_2}=3, m_{r_3}=3$ and $m_{r_4}=1$. Consider the rating group g_R consists of all young reviewers who have rated some restaurant from New York city. Recall that Figure 2 depicts a numeric description of two rating maps associated with g_R. The first (rm_{r_2}) obtained by grouping the records according to the restaurants' neighborhood, and is defined over the food dimension. The second (rm'_{r_4}) obtained by grouping the reviewers according to their gender, and is defined over the ambiance dimension. Let us assume that $u(rm_{r_2})=0.6$ and $u(rm'_{r_4})=0.8$. We get that: $\widehat{u}(rm_{r_2}, RM)=0.7 \cdot 0.6=0.42$, and $\widehat{u}(rm'_{r_4}, RM)=0.9 \cdot 0.8=0.72$.

Other weighting schemes, such as round-robin (ensuring that rating maps of different dimensions are selected *in every step*), could also be used without impacting our solution.

4) Diversity of Rating Maps: Following [15], we define $div(RM)$, the diversity of a set of maps RM, as follows:

$$div(RM) := \min_{rm, rm' \in RM} d(rm, rm') \quad (2)$$

where d is a distance function between rating map pairs. Several definitions of d are possible. In this work, we use the Earth Mover's Distance (EMD), a measure that was shown to be well-adapted to comparing rating distributions [5], [7]. EMD ensures that rating maps having different rating distributions are selected. As we will demonstrate in our experiments, this also increases the probability of choosing rating maps aggregated by different attributes, thereby exposing different data facets.

Our goal is to select *a diverse k-size set of high-utility rating maps*. There are multiple approaches for maximizing the two objectives of utility and diversity. In this work, we select the most diverse k-size set of rating maps, out of a set of the top-($k \times l$) rating maps with the highest DW utility scores, where $l>1$. Our experimental study shows that a reasonable choice for the value of l is 3. An alternative approach is to consider only rating maps whose utility scores are higher than some threshold [6]. However, this may result in too few or too many maps to consider. We leave the exploration of other possibilities for future work.

We now define the *Diverse Rating Map Selection Problem*.

Problem 2.7 (Diverse Rating Map Set Selection): Given a set of rating records g_R, a set of all possible rating maps RM_{g_R} associated with g_R, a set of rating maps RM that have been seen by the user so far, and two positive integers l and k, find a diverse k-size set of rating maps $RM' \subseteq RM_{g_R}$. Specifically, we solve: $argmax_{RM' \subseteq RM_l} div(RM')$, where RM_l is an $l \times k$ set of rating maps found by solving:
$argmax_{RM_l \subseteq RM_{g_R}} \Sigma_{rm \in RM_l} \widehat{u}(rm, RM)$.

Finding a diverse k-size set of rating maps RM' in a larger input set RM_l is a well-studied problem that requires to optimize diversity. An efficient PTIME 2-approximation algorithm can be used to solve this problem in the case diversity is a diameter [16] (which is the case in our definition). A main challenge is to avoid materializing low-utility rating maps, i.e., to build RM_l. We refer to l as the *pruning-diversity factor*. As we shall see, this factor affects our pruning optimizations. When $l=1$, solving the above problem finds a k-size set of rating maps with the highest utility scores, and when $l>1$, the diversity increases, possibly at the expense of utility.

C. The SDE Paradigm

To fully realize the SDE paradigm, we allow users to explore subjective data following one of two optional modes: *Recommendation-Powered* where at each step, the system displays a k-size set of rating maps and the top-o next-step recommendations to the user; *Fully-Automated* where the system displays a set of rating maps in each step, and generates a sequence of m steps (a.k.a. an exploration path), by applying the top-1 recommendation at each step. In both modes, the system needs to solve Problem 2.7, and to recommend next-step operations. We first define the utility of an operation, then formalize the problem of next-step recommendations.

Utility of an Exploration Operation: Recall that for each operation, the essence of the resulting rating group is presented to the user in the form or a set of rating maps. Correspondingly, we define the utility score of an operation q for a rating group g_R to reflect the utility scores of the resulting rating maps. Let RM_q denote the k rating maps generated by the application of q. The utility of q is denoted by $u(q, RM)$ and depends on the set RM of rating maps that have been seen by the user so far (i.e., up to the previous step). It is defined as the sum of the DW utilities of rating maps in RM_q:

$$u(q, RM) := \sum_{rm \in RM_q} \widehat{u}(rm, RM) \quad (3)$$

Problem 2.8 (Next-Step Recommendations): Given a group of rating records g_R, a set of previously displayed rating maps RM, and a number o, recommend the top-o next operations Q whose aggregated utility is maximized: $argmax_Q \Sigma_{q_i \in Q} u(q_i, RM)$ and $|Q| = o$.

In the next section, we will present our proposed optimizations to solve our first problem. This will also serve solving our second problem since the utility of an operation is a function of the utilities of the rating maps it returns. Our experiments will study and compare the exploration paths resulting from

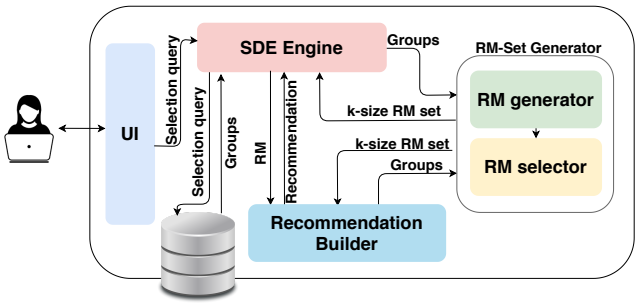


Fig. 3: SubDEx Architecture.

three exploration modes, and will demonstrate the efficiency of our solution.

III. Our SDE Framework

We outline the system workflow, then we describe our algorithms.

The architecture of SubDEx is depicted in Figure 3. Given a user selection query (that is either suggested by SubDEx or manually specified by the user), the *SDE engine* first extracts from the database the corresponding reviewer, item and rating groups. It then sends those groups to the *RM-Set generator* which returns a k-size set RM of *diverse* rating maps describing the *most interesting* trends in the current rating group. Each rating map rm, is then passed to the *Recommendation Builder* which returns the top-o most interesting next-step operations associated with rm. The *SDE Engine* then selects the overall top-o operations with the highest utility (among all generated $k \times o$ operations), and displays the selected rating maps and next-step recommendations to the user. To speed-up computation, the *SDE Engine* calls the *Recommendation Builder* several times in parallel, each time with a different rating map.

System UI. The user interacts with the system using a dedicated UI, implemented in HTML5/CSS3, depicted in Figure 4. The user investigates a rating group, by specifying the attribute-value pairs of interest defining the reviewer and item groups. The selection is done using a simple drop-down menu, or, for advanced users, by providing SQL predicates using the advanced screen (see the bottom part of Figure 4(a)). When the user investigates a rating group she can decide whether she wants to perform a recommended operation, or to provide an operation of her own. By clicking on "Apply Selection", the corresponding rating group is displayed alongside a set of rating maps. By clicking on "Get Recommendation", a pop-up window depicting next-step recommendations appears (Figure 4 (b)). To select one recommended operation, the user may click on "Apply Selection" associated with it.

In Section III-A, we provide our implementation details for assessing utility. In Section III-B, we explain how the *RM-Set Generator* selects a set of rating maps for a given rating group, and how it prunes "non-useful" rating maps. In Section III-C, we describe how the *Recommendation Builder* operates on a selected rating map.

A. Implementation Details

We describe the measures chosen in our prototype implementation to compute the utility of rating maps.

Conciseness. We measure conciseness using the compaction gain measure [17]. The conciseness score of arating map rm defined over a rating group g_R is $Conc(rm) := \frac{|g_R|}{|rm|}$, where $|rm|$ is the number of subgroups in rm. One can use other conciseness definitions, such as the Log-Length measure [18].

Agreement. To measure agreement within a subgroup of a rating map one can use existing dispersion measures (e.g., *Schutz* and *MacArthur* [19]), that favor groups of similar records. A common such measure, that we use, is *Standard Deviation* (SD), which measures the amount of dispersion of a set of values w.r.t. the mean. The agreement score of a rating map rm is defined as: $Agr(rm) := \frac{1}{\tilde{\sigma}}$, where $\tilde{\sigma}$ is the average SD score of each subgroup in rm.

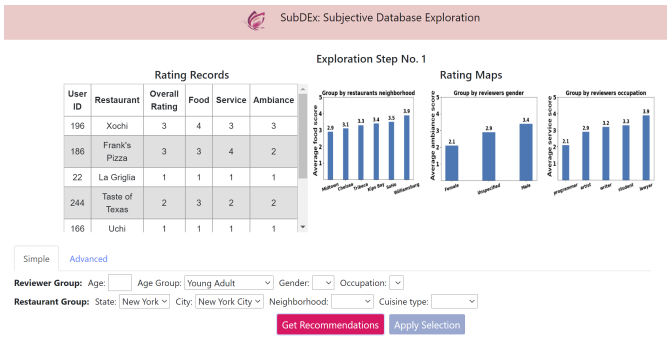
Peculiarity. To measure the peculiarity of a rating map from some reference rating map, one can use Deviation-based measures [7], that favor a rating map if it demonstrates a difference from the reference rating map. We use the *Total variation distance*, a distance measure for probability distributions. Informally, this measure is defined as the largest possible difference between the probabilities that two probability distributions (i.e., rating distributions) can assign to the same event. To compute $Pec_{self}(rm)$, the self peculiarity score of a rating map rm of a rating group g_R, we compare the rating distribution of each subgroup $g_j \subseteq g_R$ to the rating distribution of the entire rating map. Following [6], the final self peculiarity score is the maximum of the subgroups' individual scores. Given a set of rating maps the user has seen RM, we compute $Pec_{global}(rm, RM)$, the global peculiarity score by examining the peculiarity between rm's rating distribution and the distributions of each map in RM. The final global peculiarity score is the maximum of the individual scores. Alternative peculiarity measures can use the Kullback-Leibler divergence distance, or the Outlier Score Function [2].

Example 3.1: Consider again the two rating maps in Figure 2. The conciseness score of rm' is higher than that of rm, as the number of subgroups in rm' is smaller. The average agreement among each subgroup in rm' is slightly higher than that of rm. Not surprisingly, as the subgroups' rating distributions of rm are quite similar, the self peculiarity score of rm is low. In contrast, the third subgroup in rm' depicts a slightly different rating distribution from the other subgroups, and hence its self peculiarity is higher than that of rm.

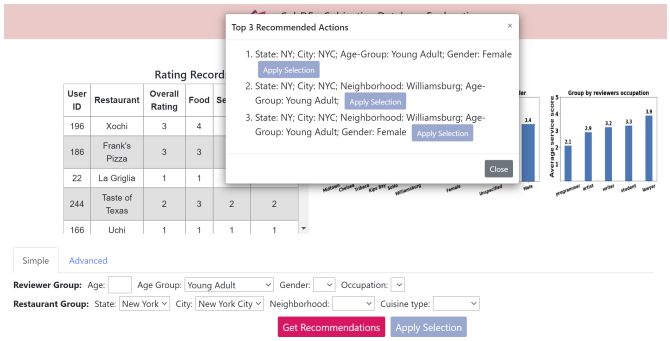
B. RM-Set Generator

At each exploration step, SubDEx displays to the user a k-size set of rating maps. To achieve that, *RM-Set Generator* is composed of the following two modules:
RM Generator that outputs, w.h.p, the $l \times k$ rating maps with the highest DW utilities. It does that by employing highly efficient pruning techniques [7] for identifying high-utility rating maps. In our setting, (and unlike in [7] where each rating map is associated with a single utility score), the utility score of a rating map is the maximum of 4 criteria (see Section II-B). Thus, the key challenge here is to adapt the optimizations of [7] to our context.
RM Selector selects a diverse k-size set of rating maps, by employing the GMM algorithm [16].

Our contribution here is the assembly and adaptation of existing techniques to our context, to serve our novel exploration



(a) An example of exploration step screen with 3 rating maps.



(b) Top-3 next-step recommendations pop-up.

Fig. 4: UI of SubDEx

Algorithm 1: Phase-based Execution Framework

input : A rating group g_R, the set of all seen rating maps RM, two constants l and k, and the number of phases n.
output: A $k\times l$-size set of rating maps R

1 $R \leftarrow$ all possible rating maps for g_R
2 $w \leftarrow$ *getWeights(RM)*
3 **foreach** $i \in [1,n]$ **do**
4 $\quad D_i \leftarrow$ the i-th fraction of the group g_R
5 $\quad$ updateResults(R,w, RM, D_i)
6 $\quad R \leftarrow$pruneResults(R, RM, $k\times l$)
7 **return** R

needs.

1) RM-Generator: We now explain how SubDEx prunes low-utility rating maps, generating only the top $l\times k$ maps with the highest DW utility scores, where k is the number of maps to be displayed, and l is a pruning-diversity factor. We adapted two types of optimizations [7] to identify high-utility rating maps for a given rating group. The first type is *sharing*, where GroupBy queries are combined to share computation. The second is *pruning*, where low-utility rating maps are dropped from consideration without scanning the whole dataset. As in [7], we operate in a phased execution framework, where each phase operates on a different, equally-sized subset of the dataset (i.e., the underlying rating group).

The framework is depicted in Algorithm 1. Our novel extensions are highlighted in red. We begin with the set of all possible rating maps (Line 1). We then compute the weights to be used for computing the DW utility scores, according to the set of previously-seen rating maps RM, by calling the procedure getWeights (Line 2). This procedure is described in Algorithm 2. During phase i, we update partial results for the rating maps that are still under consideration using the i^{th} fraction of the rating group (Lines 4−5). We do so by applying *sharing-based optimizations* to minimize the number of queries on this i-th fraction. At the end of phase i, we use *pruning-based optimizations* to determine which rating maps to discard (Line 6). In the sequel we explain our novel adaptations to the pruning optimization of [7]. The retained rating maps are then processed in the $i+1$-th phase, and the process continues. Last, we return the resulting $k\times l$-size set of rating maps R (Line 7).

The authors of [7] have empirically found that setting n, the number of phases, to 10 works well in practice, and that the results are not very sensitive to the value of this parameter. Thus, here as well, we set n=10.

Algorithm 2: The getWeights Procedure

input : The set of all previously seen rating maps RM.
output: The weights for each rating dimension $[r_1,\ldots,r_t]$.

1 $m \leftarrow |RM|$
2 $w \leftarrow$ a t-size vector initiates with zeros.
3 **foreach** $rm \in RM$ **do**
4 $\quad j \leftarrow$ the index of the rating dimension used for rm
5 $\quad w[j] \leftarrow w[j]+1$
6 **foreach** $i \in [1,t]$ **do**
7 $\quad w[i] \leftarrow w[i]/m$
8 **return** w

Sharing-based Optimizations: : The goal of these optimizations is to batch queries, reducing the number of queries issued to the database and, in turn, minimizing the number of scans of the dataset. We adapted two of the sharing optimizations of [7] that are relevant to our setting: (1) *Combining Multiple Aggregates*: Given a rating group g_R, we examine all rating maps associated with g_R. Rating maps with the same GroupBy attribute can be rewritten as a single query with multiple aggregations, i.e., multiple rating dimensions. (2) *Parallel Query Execution*: we execute multiple GroupBy queries in parallel. As noted in [7] (and also confirmed in our experiments), the optimal number of queries to run in parallel is (approximately) equal to the number of available cores.

Pruning-based Optimizations: : In practice, most rating maps are low-utility and generating them wastes computational resources. Thus, at the end of every phase, we use pruning optimizations to determine which rating maps to discard. Specifically, partial results for each rating map based on the data processed so far are used to estimate DW utility, and rating maps with low DW utility are dropped. We adapted two pruning schemes that were presented in [7]. The first uses a *confidence-interval technique* to bound utilities of rating maps. The second uses *Multi-Armed Bandit (MAB) allocation strategies* to find high utility rating maps.

Confidence-interval pruning This pruning scheme uses worst-case statistical confidence intervals to bound rating maps utilities. In our setting (and unlike in [7] where each rating map is associated with a single utility score), the utility score of a rating map is the maximum among 4 criteria (as explained in Section II-B). We use this technique in two levels. First, for each rating map, we stop estimating non-promising criteria defining utility. Second, we prune low-utility rating maps. The full pruning scheme is depicted in Algorithm 3. Our extensions to the original confidence-interval pruning optimization of [7]

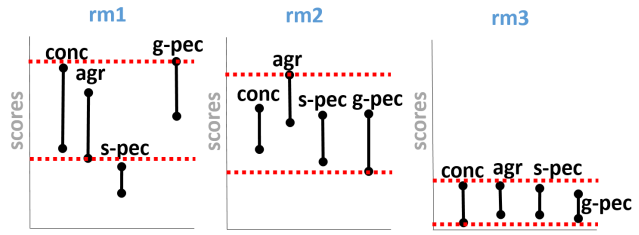


Fig. 5: Example of Confidence-Based Pruning

are highlighted in red.

We define the confidence-interval I of a rating map rm as follows. Let I_1, I_2, I_3 and I_4 denote the confidence intervals of the (normalized) conciseness, agreement, self-peculiarly and global-peculiarity scores, resp., of rm. Every interval I_i that lies entirely below another interval I_j is discarded. We define the upper-bound (resp., lower-bound) of I to be the maximum (resp., minimum) value among all remaining intervals (Line 3, resp., Lines 4 – 9). We then multiply the upper and lower bounds of the confidence interval of each rating map by the weight associated with its rating dimension (Lines 10 – 11). At that point, we keep an estimate of the mean DW utility for every rating map, and a confidence interval around that mean. We use the following rule to prune low-utility rating maps: If the upper bound of a rating map rm is less than the lower bound of k' or more rating maps, then rm is discarded (Lines 12 – 17). As in [7], we use worst-case confidence intervals derived from the Hoeffding-Serfling inequality [20].

Example 3.2: Consider the rating maps depicted in Figure 5. Each rating map is associated with 4 confidence intervals, one for each criterion defining utility. The confidence interval of $rm1$ (red dashed lines) is defined by the upper value of the interval of its global-peculiarity, and the lower value of the interval of its agreement. Observe that there is no need to estimate its self-peculiarity score, as its confidence interval lies entirely below the confidence interval of $rm1$. Now suppose that we want to identify the top-2 rating maps. The confidence interval of $rm3$ lies entirely below the intervals of $rm1$ and $rm2$. Since, w.h.p, the utility of $rm3$ lies within its confidence interval, $rm3$ utility is likely to be lower than that of $rm1$ and $rm2$, and it can thus be pruned.

MAB based Pruning Recall that our goal is to find the top $l{\times}k$ rating maps (arms) with the highest utility (reward). The authors of [7] showed that, w.h.p, the Successive Accepts and Rejects algorithm of [21] can be used to find rating maps with the highest mean utility[1]. As the MAB-based pruning technique of [7] can be directly used in our setting, we provide here, for completeness, only a brief overview of this technique. First, rating maps that are still under consideration are ranked by their DW utility means. We compute two differences between those means: Δ_1, the difference between the highest mean and the k'+1-th highest mean, and Δ_2, the difference between the lowest mean and the k'-th highest mean. If $\Delta_1{>}\Delta_2$, the rating map with the highest mean is "accepted" in the top-k', and is no longer a candidate for pruning. Otherwise, the rating map with the lowest mean is discarded from consideration.

[1] under certain assumptions about (normalized) reward distributions that hold in our setting, as was proven in [21]

Algorithm 3: Confidence-interval based pruning

input : The set of all considered rating maps R, the rating dimension weights w, and the number of rating maps to be considered $k'{=}k{\times}l$.
output: An updated set of rating maps R.

```
1  foreach rm ∈ R do
2      intervals ← SORTBYUPPERBOUND(rm.intervals)
3      rm.ub ← intervals[0].ub
4      rm.lb ← intervals[0].lb
5      foreach I ∈ intervals do
6          if I.ub ∈ [rm.ub, rm.lb] and I.lb < rm.lb then
7              rm.lb ← I.lb
8          else
9              rm.intervals.REMOVE(I)
10     rm.ub ← rm.ub · w[rm.dim]
11     rm.lb ← rm.lb · w[rm.dim]
12 R ← R.SORTBYUPPERBOUND()
13 topRatingMaps ← R.GETTOPK(k')
14 lowestLowerbound ← MIN(LOWERBOUND(topRatingMaps))
15 foreach rm ∉ topRatingMaps do
16     if rm.up <lowestLowerbound then
17         R.REMOVE(rm)
18 return R
```

2) RM-Selector: The *RM-Generator* outputs, w.h.p, the top-$k{\times}l$ rating maps with the highest DW utility. Our goal is to select the most diverse k-size set of rating maps, among these rating maps. To this end, we employ the simple and efficient GMM algorithm [16], which operates as follows. It starts with an arbitrary rating map. For $k{-}1$ times, it then chooses a new rating map whose minimum distance to the currently chosen maps is maximized. As was proven in [16], this algorithm achieves a 2-approximation factor, and its running time is $O(k^2{\cdot}l)$.

C. Recommendation Builder

Each rating map rm for a rating group g_R is associated with a o-size set of next-step recommendations. Recall that an operation q is a selection criteria defined over the underlying reviewer and item groups (i.e., g_U and g_I) of g_R. Namely, q is a set of attribute-value pairs, defined as the union of g_U and g_I. For example, in Figure 4(a) q= {⟨age_group, young⟩, ⟨state, NY⟩, ⟨city, NYC⟩}.

Let q' denote the current selection operation over a rating group g_R, and let q denote a next-step operation. Although the space of possible choices for q is very large, it is natural to expect that a user would be interested in a *small adjustment* to the current selection query [22]. Thus, to ensure that operation recommendations are understandable to users and preserve their train of thought [23], we limit q to be different from q' in at most 2 attribute-values pairs. Namely, q may add a new attribute-value pair to q', and may remove or change one of the existing attribute-value pairs in q'. Formally, given a rating group g_R and a rating map rm^{g_R}, we define q as follows. W.l.o.g, assume that rm^{g_R} partitions g_R by a reviewer attribute. Let $g{\subseteq}g_R$ be a subgroup defined by rm. g itself defines a selection criteria (i.e., an attribute-value pair) on the underling reviewer group. We add this new attribute-value pair to q, and may remove or change one of the existing attribute-value pairs.

Example 3.3: Consider again the middle rating map in Figure 4(a). This rating map partitions the rating records according to attribute gender. Assume that the current examined subgroup g selects female reviewers, and recall that q'= {⟨age_group, young⟩, ⟨state, NY⟩, ⟨city, NYC⟩}. The possible choices for q include {⟨age_group,

young⟩, ⟨state, NY⟩, ⟨city, NYC⟩, ⟨gender, female⟩} (only add a new attribute-value pair to q'), {⟨age_group, young⟩, ⟨state, NY⟩, ⟨gender, female⟩} (also remove one attribute-value pair from q'), and {⟨age_group, adult⟩, ⟨state, NY⟩, ⟨city, NYC⟩,
⟨gender, female⟩} (also change one attribute-value pair from q').

Candidate operations are ranked according to their utility, as defined in Section II-C. To compute the utility of an operation q, the *Recommendation Builder* extracts from the database the relevant reviewer, item and rating groups. It then uses the *RM-set Builder*, to find the k-size set of rating maps to be displayed in the next step. We can compute the utility scores of x operations simultaneously, where x is the number of available cores. Finally, given a rating map *rm*, the *Recommendation Builder* returns the top-o operations associated with *rm* with the highest utility scores.

An interesting future work is to develop additional optimizations for pruning low-utility operations. Nonetheless, as our experiments show, our prototype produces the recommendations sufficiently fast by using parallelism, enabling interactive running times.

IV. Demonstration

We demonstrate the operation of SubDEx over three real-world subjective datasets: 12) Movielens[2], which contains reviewers' ratings on movies; (2) Yelp[3], which contains people's reviews of various businesses, including restaurants; (3) Hotel Review[4], which consists of reviewers' reviews on hotels. For the last two datasets, following [2], we extracted from the reviews text the rating scores for multiple rating dimensions (e.g., food, service, and ambiance for restaurants). We evaluate two aspects of SubDEx (i) learnability and usability, showing the ability of users to use the functionalities of SubDEx for different information needs , and (iii) scalability, examining how different parameters affect the performance.

Learnability and usability: We demonstrate the learnability and usability of the system via two scenarios.

Identifying special data characteristics. We simulate a scenario where a data analyst seeks "irregular" groups. An irregular group is described by two or three attribute-values shared by the reviewers (resp., items), whose rating scores for the same rating dimension have all been set to 1. This scenario simulates a common real-life event where the goal is to identify special data characteristics. To examine the benefit of guidance during exploration, we will randomly assign each participant with one of the optional exploration modes, and will ask her to load one of the datasets. The participants would then use the system to find the irregular groups.

Insight extraction. In the second scenario, we will use SubDEx for the task of insight extraction - a common goal of data exploration. For all examined datasets, the Kaggle platform[5] contains several EDA notebooks, manually created by fellow data scientists to demonstrate their EDA process in obtaining insights. From these notebooks, we gathered three lists containing between 5 to 10 insights on each dataset. An example of insight on MovieLens is that the average rating score young adult reviewers gave to thriller movies is significantly higher than that of adult reviewers. Here again, each participant can choose a dataset, and will be randomly assigned with one of the exploration modes.

In both scenarios, the audience can examine statistics describing the aggregated results of other participants. These statistics include the average time it took users to complete the task, the average number of exploration steps, and the average precision and recall. These statistics are obtained by aggregating the results by different exploration modes and by different datasets.

Scalability: Last, the audience will be allowed to look "under the hood", examining the efficiency of our algorithms. For this part of the demonstration, we will use growing fragments of the underlying database, showing the effect of different data and system parameters on performance.

[2]https://grouplens.org/datasets/movielens/100k/

[3]https://www.yelp.com/dataset

[4]https://www.kaggle.com/datafiniti/hotel-reviews

[5]https://www.kaggle.com/

References

[1] "Technical report," shorturl.at/sKQS9, 2020.

[2] Y. Li, A. Feng, J. Li, S. Mumick, A. Halevy, V. Li, and W.-C. Tan, "Subjective databases," *PVLDB Endowment*, 2019.

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

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

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

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

[7] M. Vartak, S. Rahman, S. Madden, A. Parameswaran, and N. Polyzotis, "Seedb: Efficient data-driven visualization recommendations to support visual analytics," in *PVLDB Endowment*, 2015.

[8] B. Omidvar-Tehrani and S. Amer-Yahia, "User group analytics survey and research opportunities," *TKDE*, 2019.

[9] M. Vartak, S. Huang, T. Siddiqui, S. Madden, and A. Parameswaran, "Towards visualization recommendation systems," *SIGMOD*, 2017.

[10] A. Kim, E. Blais, A. Parameswaran, P. Indyk, S. Madden, and R. Rubinfeld, "Rapid sampling for visualizations with ordering guarantees," in *PVLDB*, 2015.

[11] J. L. Herlocker, J. A. Konstan, and J. Riedl, "Explaining collaborative filtering recommendations," in *CSCW*, 2000.

[12] M. Das, S. Amer-Yahia, G. Das, and C. Yu, "Mri: Meaningful interpretations of collaborative ratings," *VLDB*, 2011.

[13] G. E. Box and D. R. Cox, "An analysis of transformations," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 26, no. 2, pp. 211–243, 1964.

[14] T. C. Urdan, *Statistics in plain English*. Taylor & Francis, 2016.

[15] S. Abbar, S. Amer-Yahia, P. Indyk, and S. Mahabadi, "Real-time recommendation of diverse related articles," in *WWW*, 2013.

[16] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical computer science*, 1985.

[17] V. Chandola and V. Kumar, "Summarization–compressing data into an informative representation," *KAIS*, 2007.

[18] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[19] R. J. Hilderman and H. J. Hamilton, *Knowledge discovery and measures of interest*. Springer Science & Business Media, 2013, vol. 638.

[20] R. J. Serfling, "Probability inequalities for the sum in sampling without replacement," *The Annals of Statistics*, pp. 39–48, 1974.

[21] S. Bubeck, T. Wang, and N. Viswanathan, "Multiple identifications in multi-armed bandits," in *International Conference on Machine Learning*, 2013, pp. 258–265.

[22] N. Koudas, C. Li, A. K. Tung, and R. Vernica, "Relaxing join and selection queries," in *PVLDB*, 2006.

[23] O. Bar El, T. Milo, and A. Somech, "Automatically generating data exploration sessions using deep reinforcement learning," in *SIGMOD*, 2020.