Exploring Ratings in Subjective Databases

Sihem Amer-Yahia, Tova Milo, Brit Youngmann

Exploring Ratings in Subjective Databases (Authors' Copy)

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

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

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

ABSTRACT

Subjective data links people to content items and reflects who likes or dislikes what. The valuable information this data contains is virtually infinite and satisfies various information needs. Yet, as of today, dedicated tools to explore this data are lacking. In this paper, we develop a framework for Subjective Data Exploration (SDE). Our solution 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 based on confidence intervals and multi-armed bandits. Our large-scale experiments with human subjects and real datasets, demonstrate the need for dedicated SDE frameworks and the effectiveness and efficiency of our approach.

CCS CONCEPTS

• **Human-centered computing** → *Systems and tools for interaction design*; **Visualization systems and tools**.

KEYWORDS

Subjective Data, Data Exploration, Recommender Systems.

ACM Reference Format:
Sihem Amer-Yahia, Tova Milo, and Brit Youngmann. 2021. Exploring Ratings in Subjective Databases (Authors' Copy). In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21), June 20–25, 2021, Virtual Event, China.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3448016.3457259

1 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 [39, 52]. Yet, as of today, dedicated exploration tools are lacking. In this paper, we develop SubDEx, a framework for Subjective Data Exploration (SDE).

As in general-purpose Exploratory Data Analysis (EDA), SDE requires iterative data filtering and generalization. Additionally, the purpose of SDE is to examine user-item relationships. For instance, a social scientist who studies cinema-related population trends in Movielens, would benefit from seeing aggregated movie ratings, followed by a selection to explore the opinion of some reviewers. Similarly, when examining restaurants in Yelp, one may benefit from seeing aggregated ratings on a type of cuisine in a given neighborhood by reviewers in a certain age range, followed by a request to cover additional neighborhoods. Like in EDA, SDE users need guidance as they seldom know precisely what they are looking for and generally only have partial knowledge of the underlying data. Additionally, SDE must satisfy specific needs that occur when exploring a mix of facts and opinions. Let us consider an example.

Example. *Mary is a social scientist looking for 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 in 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). With only a few steps, Mary obtained detailed insights on people's opinions on New York City restaurants.*

Mary's example illustrates the key characteristics of SDE. The first is the need to process items, users, and ratings as first-class citizens, to simultaneously filter them and aggregate their relationships (need **N1**). The second need is to diversify aggregation dimensions, e.g., food vs. service for restaurants, across exploration steps (need **N2**). Table 1 summarizes those needs and positions our work appropriately. Columns in bold font emphasize the new requirements, namely: (1) revisit the "quality" of an exploration step to capture a new type of diversity on aggregation dimensions, (2) simultaneously recommend drill-down/roll-up operations and visualizations, and (3) combine optimizations of operations and visualizations due to the very large number of user-item relationships. In this work, we approach the problem holistically and develop a dedicated framework for SDE to achieve all those requirements.

Our first contribution is to formalize the SDE framework. We represent the data as a bipartite graph with reviewer-nodes and item-nodes, and links between them. At each exploration step, the user applies filtering or generalization: e.g., select an age group for reviewers to examine their opinions, or generalize the location of restaurants to cover more neighborhoods. Each operation returns

SIGMOD '21, June 20–25, 2021, Virtual Event, China

ACM ISBN 978-1-4503-8343-1/21/06…$15.00
https://doi.org/10.1145/3448016.3457259

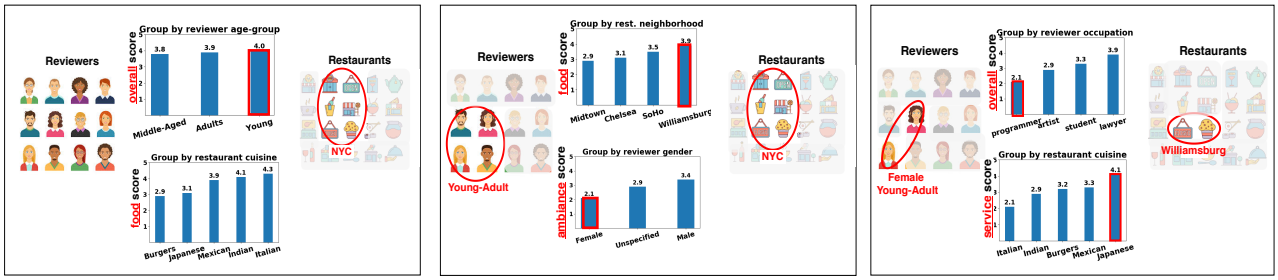


Figure 1: Example of a three-step exploration. The user iteratively examines subsets of the reviewer and item tables. Links between selected reviewer and item groups are aggregated as rating maps, showing the most "interesting" trends in the data.

subsets or supersets of items and reviewers whose ratings are aggregated into k *rating maps* [9]. 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 [43, 51] and is defined as the highest value between *conciseness* that favors a small number of aggregation dimensions (bars in the histogram), *agreement* among reviewers within each histogram [16], and *peculiarity*, to highlight unseen patterns [51]. Our notion of diversity reveals different data facets. To ensure that the selected rating maps depict different rating dimensions (need **N2**), we use weighted utility scores where the weights reflect the number of times a rating dimension was previously shown.

Our second contribution is to realize the SDE paradigm by offering three exploration modes: *User-Driven*, *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 the user decides the next-step operation. This was the case for Mary in our example. In the *Recommendation-Powered* mode, the system recommends o next-step operations based on the utility and diversity of the rating maps they generate. The user can choose one recommendation or perform an operation of her own. The third *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. All exploration modes need to solve the *Diverse Rating Map Set Selection Problem* to determine the k ratings maps to display at each step. The *Recommendation-Powered* and *Fully-Automated* modes further give rise to the *Next-Step Recommendations Problem* that returns the most useful next-operations.

Our third contribution is computational. In practice, most rating maps are low-utility and generating them wastes resources. We therefore rely on pruning optimizations that estimate the weighted utility of partial results for each map based on the data processed so far, and prune low utility ones. To enable that, we adapt the two pruning schemes proposed in [54] to find "interesting" visualizations for a given dataset. The first uses a *worst-case confidence-interval technique* derived from the Hoeffding-Serfling inequality [48], to bound the utilities of rating maps, and the second uses *multi-armed bandit allocation strategies* to find the most useful rating maps. To address the *Diverse Rating Map Set Selection Problem*, we compute $l{\times}k$ highest-utility maps, where l is a positive pruning-diversity factor. We then choose k most diverse rating maps. To this end, we employ the simple and efficient GMM algorithm [29], which starts with an arbitrary rating map and iterates $k{-}1$ times to find rating maps whose minimum distance to the currently chosen maps is maximized. This algorithm achieves an approximation factor of 2, and its running time is $O(k^2{\cdot}l)$.

Our fourth contribution is a thorough empirical investigation to: (1) examine if user guidance (via recommendations or full automation) in SDE is useful in addressing information needs for users with different CS expertise and domain knowledge, and (2) study the scalability of our solution. Our experimental study is a first step toward designing an SDE-specific benchmark that differs from data exploration benchmarks [22] in that it focuses on extracting insights on user-item relationships. Our user study with two common SDE scenarios, identifying special data characteristics and extracting insights, finds that (1) Only showing rating maps does not provide enough information to guide users effectively, even when they are CS experts; (2) *Fully-Automated* SDE is not flexible enough, as the user cannot intervene; (3) *Recommendation-Powered* SDE is helpful regardless of CS expertise; (4) Results do not depend on domain knowledge. This validates the need for an iterative partially-guided SDE. We further compare the quality of SubDEx's recommendations with state-of-the-art drill-down view exploration and resul summarization, showing SubDEx outperforms its competitors. Additionally, we show that the exploration paths we generate address needs **N1** and **N2**. In particular, when the task is to identify special data characteristics, utility-only exploration is superior. In contrast, diversity-only exploration allows to examine various data facets when the task is to extract. This suggests that SubDEx could be tuned according to the task at hand. Finally, we show that our optimization strategies offer a scalable solution to SDE.

A demonstration of SubDEx's usability and its suitability to end-to-end employment was presented in [10]. The short paper accompanying the demonstration provides only a high-level description of the system, whereas the present paper provides the theoretical foundations and algorithms underlying the demonstrated system.

2 RELATED WORK

Table 1 summarizes the specific needs of SDE, pointing out the unique features of SubDEx, compared with previous work on data exploration and result summarization. SubDEx ensures that selected results in each step are diverse (one-shot diversity). It also ensures multi-step diversity and diversity among different aggregation dimensions (dimension diversity). Furthermore, SubDEx recommends data visualizations (viz recommendation) and next-action

Table 1: Positioning of SDE with respect to Data Exploration and Result Summarization.

Related Work		Diversity			Recommendation			Optimization		
		One-Shot	Multi-Step	**Dimensions**	Op	Viz	**Op&Viz**	Op	Viz	**Op&Viz**
Exploration	[17, 23, 33, 42, 47, 51]				**X**			**X**		
	[11]		**X**		**X**			**X**		
	[35]	**X**			**X**			**X**		
	[18, 30, 36, 38, 49, 54]					**X**			**X**	
	[9, 59]	**X**				**X**			**X**	
Summarization	[45]	**X**			**X**	**X**	**X**	**X**	**X**	**X**
	[24, 55, 58]	**X**			**X**			**X**		
	[8, 46]				**X**			**X**		
SDE	SubDEx	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**

Reviewers

User ID	Name	Gender	Age Group	Occupation
1	Alice	F	Middle Aged	Lawyer
2	Bob	M	Young	Artist
3	Carol	F	Young	Student
4	David	M	Middle Aged	Teacher

Restaurants

Rest ID	Name	Cuisine	State	City	Zip
1	Joe's Farm Grill	Burgers, Barbeque	North Carolina	Charlotte	77474
2	Uchi	Japanese, Sushi	Texas	Austin	65414
3	Taqueria Pueblo	Mexican	Michigan	Detroit	37268
4	Home Slice Pizza	Pizza, Italian	New York	NYC	72297

Rating Records

User ID	Rest ID	Overall	Food	Service	Ambiance
1	4	4	3	5	4
1	7	5	5	5	4
2	1	4	4	3	5
2	2	3	4	3	3
3	4	5	5	5	4

Figure 2: A subjective database with 4 rating dimensions.

operations (op recommendation) *simultaneously*, and employs dedicated optimizations for both types of recommendations.

Subjective Databases. Subjective data analysis is an emerging research field [25, 39, 41, 56, 63], allowing to mine and analyze user-generated data. Such data is widely used in web applications, online rating systems, and more generally in the social sciences [14, 43]. SDE serves large-scale population studies whose purpose is to extract insights on user-item relationships [9, 21].

Exploratory Data Analysis. Exploratory Data Analysis (EDA) consists in examining a new dataset with the goal of extracting insights [40, 42, 64]. Guiding users in performing EDA is a well-studied task [11, 27]. Numerous works proposed next-step recommendations [42], by using logs of previous operations (e.g., [23]), or by relying on real-time feedback [17, 33]. Fully automated generation of EDA sessions has been examined in [11, 47]. A novel operator for interactively exploring a relational table to discover and summarize "interesting" groups of tuples was introduced in [35]. The authors of [38] detect *drill-down fallacy*, an error that may occur when incomplete insights are extracted along a drill-down path.

Data visualization is an essential step in extracting insights from datasets. Auto-generation of interesting views is studied extensively [20, 30, 35, 50, 53, 59]. As mentioned, SubDEx provides data visualizations by generating *rating maps* [9]. Previous work has shown that such histograms are an adequate means of understanding rated datasets [9, 31]. A common approach that we follow is to use heuristic measures of interestingness [43, 51], searching the space of all views, and returning the most interesting ones [49, 54]. Numerous works have proposed solutions to enable scalable data visualization [30, 38, 49]. For example, the authors of [36] proposed sampling to find approximate visualizations while preserving crucial properties. The authors of [18] employ in-memory caching and pre-fetching to improve interactivity. A framework for accelerating statistical analysis by avoiding repeated data access and computation was presented in [57]. Here we leverage pruning and sampling optimizations to determine which rating maps are low-utility and can be discarded [54].

Result Diversification. Result diversification is well-studied in query answering in databases [44], search engines [28] and recommender systems [61]. This problem aims to return k results that take both utility and diversity into consideration [19]. In many cases, diversity comes at the cost of utility [44, 65]. A common approach to measuring diversity, which we also adopted in our work, relies on pairwise similarities [26, 44]. The main difference from previous work is that we also account for: (1) diversity among rating maps selected in previous steps (multi-step diversity, accounted by the global peculiarity scores), and (2) diversity among different aggregation dimensions (dimensions diversity, accounted by the dimension-weighted utility scores). This makes that choice of which k-size set of rating maps to display at each step more complex, as it requires to account for three factors of diversity *simultaneously*, while also ensuring only high-utility rating maps are selected.

A related problem is imposing diversity constraints that explicitly increase the representation of historically disadvantaged populations or improving overall representativeness of selected populations [60, 62]. This line of work is complementary to ours, and could be used to ensure that the opinions of different sub-populations are displayed. Namely, one may require that rating maps depicting the aggregated rating scores of specific sub-populations are selected.

Result summarization and explanation. The goal of result summarization is to make a large set of results more informative [24, 58]. Unlike our work, summarization is a one-shot process. Such works produce k clusters showing their common properties such that the clusters are diverse [45]. However, as can be seen in Table 2, since this work is not specific to SDE, it does not handle multi-step diversity, neither does it explicitly handle diversity among aggregation dimensions. Summaries offer interpretable explanations of query results and are referred to as *explanation tables* [8, 46, 55]. As we demonstrate in our experiments, such algorithms can be leveraged to produce "interesting" next-action recommendations. However, they yield only operations that select subsets of an input, whereas SubDEx also recommends operations that extend it.

Recommender systems. Recommender systems make use of different sources of information to provide users with recommendations

of content items [12], of next-actions [42], or of data visualizations [30, 38, 54, 59]. Unlike previous work (e.g., [42, 54]), SubDEx is able to recommend data visualizations and next-actions *simultaneously*. This is made possible by the ability to rank next-step operations using the interestingness of the rating maps they return.

3 DATA MODEL AND SDE

3.1 Data Model

We consider a special type of database, called a subjective database [39], which includes both objective and subjective attributes. We model our database $\mathcal{D}$ as a triple $\langle \mathcal{I}, \mathcal{U}, \mathcal{R} \rangle$, representing the sets of items, users (referred 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 [39] or directly provided by reviewers. Here we assume that there is 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 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 assigned by reviewer u 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 2, 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.

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 attributes defining its description $\{\langle a_1, v_1 \rangle, \langle a_2, v_2 \rangle, \ldots\}$ where $a_i \in \mathcal{U}_A$ (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. 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 expressed as a predicate on the rating table.

3.2 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 (i.e., drill-down/roll-ups) operations. After each operation, she examines the results and decides to execute or not a new operation.

3.2.1 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.

3.2.2 Rating Maps. To provide a bird's eye view of the ratings in a group g_R, we use *rating maps* [9] - 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 [31]. We now define the notions of rating distributions and rating maps.

Definition 1 (Rating Distribution [9]). *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 (Rating Map [9]). *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. *We assume a rating group g_R containing the ratings of young reviewers for restaurants in NYC: g_R contains rating records for items in $g_I = \{\langle$`city, NYC`$\rangle\}$ and reviewers in $g_U = \{\langle$`age_group, young`$\rangle$. The top two tables in Figure 3 show two example rating maps that correspond to the ones displayed in Figure 1, Step II. The first partitions g_R by neighborhood. It associates to each subgroup its rating distribution for food, and the average score. The second partitions g_R by gender. 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.

3.2.3 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 mentioned, an SDE tool must address two novel needs: the need to process items, users, and ratings as first-class citizens, to simultaneously filter them and aggregate their relationships (**N1**), and the need to diversify aggregation 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 [43, 51]. 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 4.1.

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

city	# of records	rating distribution	avg. score
Williamsburg	16	{1 : 1, 2 : 2, 3 : 1, 4 : 5, 5 : 7}	3.9
SoHo	20	{1 : 3, 2 : 3, 3 : 2, 4 : 5, 5 : 7}	3.5
Kips Bay	12	{1 : 2, 2 : 2, 3 : 2, 4 : 1, 5 : 5}	3.4
Tribeca	12	{1 : 3, 2 : 1, 3 : 2, 4 : 1, 5 : 5}	3.3
Chelsea	20	{1 : 3, 2 : 1, 3 : 9, 4 : 5, 5 : 2}	3.1
Midtown	20	{1 : 3, 2 : 3, 3 : 9, 4 : 3, 5 : 2}	2.9

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

gender	# of records	rating distribution	avg. score
Male	35	{1 : 5, 2 : 6, 3 : 4, 4 : 9, 5 : 11}	3.4
Unspecified	30	{1 : 5, 2 : 8, 3 : 7, 4 : 5, 5 : 5}	2.9
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

Figure 3: Example of two rating maps, and their associated interestingness scores.

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 [16].

Peculiarity This measure ranks a rating map higher if its rating distribution demonstrates a difference from some reference rating distribution. 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 subgroup within it w.r.t. 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 yet.

As the values of interestingness measures are on different scales, we normalize them as proposed in [51].

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 *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 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. *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 3 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: $\widehat{u}(rm_{r_2}, RM)$=0.7·0.6=0.42, and $\widehat{u}(rm'_{r_4}, RM)$=0.9·0.8=0.72.*

3.2.4 Diversity of Rating Maps. Following [7], we define $div(RM) = \min_{rm,rm' \in RM} d(rm, rm')$, 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 [9, 54]. 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.

PROBLEM 1 (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 [29] (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.

Note that we account for three facets of diversity: (1) diversity among rating maps selected in the current step (accounted by Problem 1); (2) diversity among rating maps selected in previous steps (accounted by global peculiarity), and (3) diversity among different aggregation dimensions (accounted by the DW utility scores).

3.3 The SDE Paradigm

To fully realize the SDE paradigm, we allow users to explore subjective data following one of the modes: *User-Driven* where the system displays a set of rating maps at each step, and the user inputs the next operation to be applied to the underlying reviewer and item

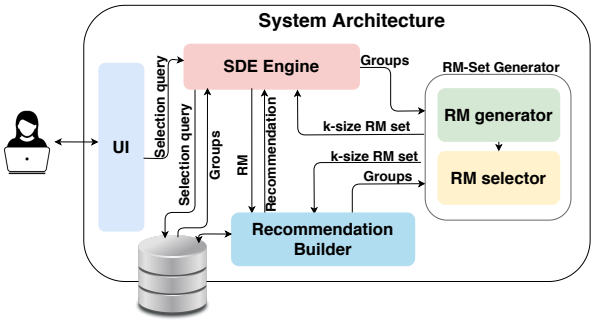


Figure 4: SubDEx Architecture.

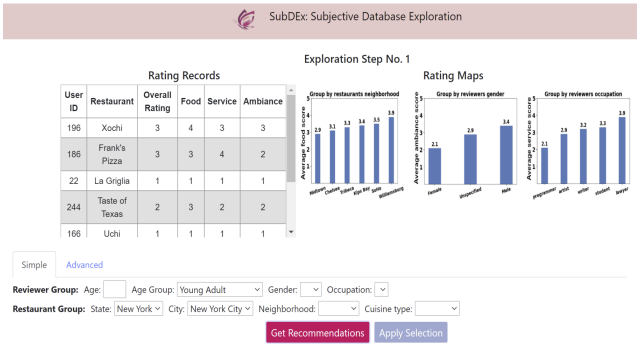


Figure 5: UI of SubDEx.

groups; *Recommendation-Powered* where at each step, the system displays a 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, by applying the top-1 recommendation at each step. In all modes, the system needs to solve Problem 1. The *Recommendation-Powered* and *Fully-Automated* modes, also require to recommend next-step operations. We next define the utility of an operation, and formalize the problem of next-step recommendations.

Utility of an Exploration Operation. For each candidate operation, the essence of the resulting rating group is presented to the user in the form of a set of rating maps, describing its most interesting trends. Correspondingly, we define the utility score of an operation q 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 previously seen by the user (i.e., up to this step). It is defined as the sum of the DW utilities of rating maps in RM_q:

$$u(q, RM) := \sum_{rm \in RM_q} \widehat{u}(rm, RM) \quad (2)$$

Problem 2 (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 $|Q| = o$.*

Solving Problem 1 also serves solving Problem 2 since the utility of an operation depends on utilities of rating maps it returns.

4 OUR SDE FRAMEWORK

The architecture of SubDEx is depicted in Figure 4. 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 5. The user investigates a rating group, by specifying reviewer/item attribute-value pairs of interest. The selection is done using simple drop-down menus, or, for advanced users, by providing SQL predicates using the advanced screen. To move to the next step, the user 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.

4.1 Implementation Details

In our implementation, we measure **conciseness** using the compaction gain measure [15]: $Conc(rm) := \frac{|g_R|}{|rm|}$. To measure **agreement** one can use existing dispersion measures (e.g., *Schutz* and *MacArthur* [32]), that favor groups of similar records: $Agr(rm) := \frac{1}{\tilde{\sigma}}$, where $\tilde{\sigma}$ is the average SD score of each subgroup in rm. To measure **peculiarity** of a rating map from some reference rating map, we use the *Total variation distance*, a distance measure for probability distributions. 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 rm. Following [51], 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 the global peculiarity score, by examining the peculiarity between rm's rating distribution and the distributions of each map in RM. The final score is the maximum of the individual scores. Alternative peculiarity measures are the Kullback-Leibler divergence distance, or the Outlier Function [39].

Example. *Consider again the two rating maps in Figure 3. 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.*

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

4.2 RM-Set Generator

At each exploration step, SubDEx displays to the user a diverse k-size set of high-utility rating maps. As mentioned, we account to three aspects of diversity: one-shot, multi-steps, and aggregation dimension. To achieve that, *RM-Set Generator* is composed of 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 [54] for identifying high-utility rating maps. In our setting, (and unlike in [54] 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 3.2). Thus, the key challenge here is to adapt the optimizations of [54] to our context. ***RM Selector*** selects a diverse k-size set of rating maps, by employing the GMM algorithm [29].

4.2.1 RM-Generator. This module generates only the top $l{\times}k$ maps with the highest DW utility scores, where l is a pruning-diversity factor. We adapted two types of optimizations [54]. 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 [54], we operate in a phased execution framework, where each phase operates on a different, equally-sized subset of the dataset.

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 [54]. 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 [54] found that setting n, the number of phases, to 10 works well in practice Thus, here as well, we set n=10.

Sharing-based Optimizations. : The goal of these optimizations is to batch queries, reducing the number of queries issued to the database. We adapted two of the sharing optimizations of [54]: (1) *Combining Multiple Aggregates*: Given a rating group g_R, we examine all rating maps associated with g_R. Rating maps with the same grouping attribute can be rewritten as a single query with multiple aggregations. (2) *Parallel Query Execution.* As noted in [54] (and also confirmed in our experiments), the optimal number of queries to run in parallel is equal to the number of available cores.

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

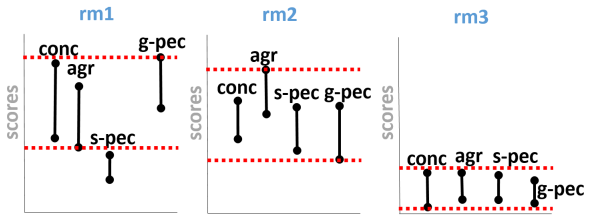


Figure 6: Example of Confidence-Based Pruning

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 [54]. 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 [54] 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 3.2). We use this technique to estimate non-promising criteria defining utility and to prune low-utility rating maps (Algorithm 3).

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. The upper-bound (resp., lower-bound) of I is defined to be the maximum (resp., minimum) value among all remaining intervals (Line 3, resp., Lines 4–9). We then multiply the bounds of the interval of a 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 prune low-utility rating maps as follows. 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 [54], we use worst-case confidence intervals derived from the Hoeffding-Serfling inequality [48].

Example. *Consider the rating maps depicted in Figure 6. 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*

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.

```
foreach rm ∈ R do
    intervals ← sortByUpperBound(rm.intervals)
    rm.ub ← intervals[0].ub
    rm.lb ← intervals[0].lb
    foreach I ∈ intervals do
        if I.ub ∈ [rm.ub, rm.lb] and I.lb < rm.lb then
            rm.lb ← I.lb
        else
            rm.intervals.remove(I)
    rm.ub ← rm.ub · w[rm.dim]
    rm.lb ← rm.lb · w[rm.dim]
R ← R.sortByUpperBound()
topRatingMaps ← R.getTopK(k')
lowestLowerbound ← min(lowerBound(topRatingMaps))
foreach rm ∉ topRatingMaps do
    if rm.up < lowestLowerbound then
        R.remove(rm)
return R
```

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. Assume 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 this interval, rm3 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 [54] showed that, w.h.p, the Successive Accepts and Rejects algorithm of [13] can be used to find rating maps with the highest mean utility[1]. The MAB-based pruning technique of [54] 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 and the k'+1-th means, and Δ_2, the difference between the lowest and the k'-th means. If $\Delta_1>\Delta_2$, the rating map with the highest mean is "accepted" in the top-k'. Otherwise, the rating map with the lowest mean is discarded.

4.2.2 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 them (i.e., one-shot diversity). We employ the simple and efficient GMM algorithm [29]. 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 [29], this algorithm achieves a 2-approximation factor, and its running time is $O(k^2\cdot l)$.

4.3 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 5(a) q= {⟨age_group, young⟩, ⟨state, NY⟩, ⟨city, NYC⟩}.

[1]under certain assumptions about (normalized) reward distributions that hold in our setting, as was proven in [13]

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 [37]. Thus, to ensure that operation recommendations are understandable to users and preserve their train of thought [11], 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'.

Example. *Consider again the middle rating map in Figure 5(a). 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 3.3. 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. Namely, we are simultaneously recommending on rating maps and next-actions, providing an optimized solution for both tasks. 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.

5 EXPERIMENTAL STUDY

The goal of our experiments is to (1) examine if guidance in SDE addresses users' needs, and (2) study the scalability of our solution.

5.1 Experimental Setup

The experiments were executed on a Linux server with a 2.1GHz CPU, and 96GB memory.

Datasets. We examine three datasets, which suitably include both demographics and subjective opinions, as depicted in Table 2.[2]

Movielens [5]. This commonly used dataset includes information about 943 reviewers who rated at least 20 movies each. The data includes age, gender, occupation, and zip code. We enriched this dataset with the reviewers' city, state and ge-group (extracted from the zip code and age), and the movies' release year and decade (extracted from the release date given in the dataset).

Yelp [6]. We use the subset of Yelp which contains restaurant reviews. Following [39], we analyzed the reviews and extracted the rating scores for the restaurants' food, service and ambiance (dimensions that were shown to be relevant in this domain [39]). Given a rating dimension, e.g., service, we extracted all phrases which include the word "service" and a fixed window of words around it of size 5. We assigned a sentiment to each phrase using the Vedar sentiment analysis measure [34] and computed the average sentiment of all relevant phrases for each rating dimension.

[2]The datasets are available at [3].

Hotel Reviews [2]. This publicly available dataset includes information on more than 15K reviewers. Here as well, we extracted from each review the rating scores for the hotels' cleanliness, food, and comfort level (following [39]). As the Hotel Review dataset demonstrated similar trends to Yelp, we omit it to save space.

Baseline Algorithms. For quality evaluation, we implemented two state-of-the-art baselines for producing next-action recommendations. **Smart Drill-Down (SDD)** takes a rating group g_R and returns k next-action operations following [35] that finds a k-size rule-list of "interesting" parts of a table (i.e., a rating group). Each rule is a selection operation over reviewer and item groups. Three factors make a rule-list interesting. One is if it contains rules that cover a large fraction of g_R. A second factor is if the rules are "specific", , i.e., the subgroups. The third factor is diversity, which measures how different the rules are from one another. *This serves as a baseline drill-down view exploration approach.* **Qagview** takes a rating group g_R and returns k next-action operations [58], which finds a k-size diverse summary of a query result (i.e., a rating group). The summary consists of k clusters, each corresponds to a selection operation over the underlying reviewer and item groups. Intuitively, the summary is chosen to cover as many rating records in g_R as possible, and ensures that the selected patterns are different from one another. *This serves as a baseline result summarization approach.*

For these baselines, we joined the item, reviewer and rating tables, so that each next-action recommendation corresponds to a simultaneous selection query over the reviewer and item groups.

For scalability evaluation, we examined the following baselines, to evaluate the marginal contribution of each of our proposed optimizations: **(I) No-Pruning** A restricted variant of SubDEx with no pruning. **(II) CI Pruning** A restricted variant of SubDEx that uses only the confidence interval pruning (Section 4.2). **(III) MAB Pruning** A restricted variant of SubDEx that uses only the multi-armed bandit pruning (Section 4.2). **(IV) No Parallelism** A restricted variant of SubDEx that uses the recommendation builder sequentially, each time processing only a single rating map. **(V) Naive** A restricted variant of SubDEx with no pruning and no parallelism.

Table 3 contains all default values. For Qagview, we set the value of all records to 1, as the rating records are not valued. We set the threshold on the number of records in a rating group g_R to be covered, to be $\frac{|g_R|}{2}$, and set the distance parameter D, requiring that the clusters are differ in at least D attribute-values, to be 2.

5.2 Qualitative Evaluation

These experiments examine the role of exploration guidance in addressing users' information needs, the quality of SubDEx's next-action recommendations, and validate our formulation for selecting a diverse set of high-quality rating maps. We do not examine alternative data visualizations since previous work have shown that rating maps are adequate for understanding rated datasets [9, 31].

Unlike in general EDA, SDE focuses on extracting insights on "user-item relationships". While SubDEx caters to many SDE tasks, in what follows, we focus on two typical scenarios whose goal is to find particular types of user-item relationships:

Scenario I. *Identifying special data characteristics.* In this scenario the goal is to find 'irregular" item/reviewer 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 (the minimal value of) 1. Each irregular group was created with at least five reviewers or items and was generated by selecting attribute-value pairs uniformly at random. This scenario simulates a real-life event where the goal of a data analyst is to identify special data characteristics. The subjects are asked to use SubDEx to find two irregular groups (one reviewer group and one item group). For each subject, we measure the number of correctly identified irregular groups (0, 1, or 2).

Scenario II. *Insight extraction.* In this scenario, we use SubDEx for the task of insight extraction - a common goal in data exploration. For both Movielens and Yelp, the Kaggle platform [4] contains several EDA notebooks, manually created by fellow data scientists to demonstrate their EDA process in obtaining insights. From these notebooks, we extracted 5 insights on each dataset. The subjects are tasked with the goal of using SubDEx to extract as many insights as possible from a dataset. For each subject, we measure the number of correctly identified insights ([0, 5]).

5.2.1 Exploration Guidance. To examine the benefit of guidance during exploration, we run a user study and compare *User-Driven*, *Recommendation-Powered*, and *Fully-Automated* modes for different datasets, scenarios, and user CS and domain expertise. Our study is conducted in 3 stages: pre-qualification, exploration, post-test. The purpose of pre-qualification is to assess users' expertise in CS and their domain knowledge. The post-test measures how well the subjects succeeded in the task. For each dataset and scenario, we recruited 120 subjects on Amazon Mechanical Turk [1]. This sample size enables us to observe a 95% confidence level with a 10% margin of error. We used the results of pre-qualification to group subjects into 4 treatment groups *"high/low domain knowledge, and high/low CS expertise"*. Subjects with high CS expertise were assigned to the *User-Driven* and *Recommendation-Powered* modes. Others were assigned to the *Recommendation-Powered* and *Fully-Automated* modes. Subjects were asked to perform a task twice (each time using a different mode) and identify different irregular groups/insights. To control for selection bias, in each treatment group, half of the subjects (15) started with one mode then performed the task again with the second mode, and the others did the reverse.

Pre-qualification. For Movielens, the questionnaire consists of 10 questions on movies. Subjects who answered correctly more than 5 questions were assigned to high domain knowledge groups. Similarly, for CS expertise, 10 questions were used to group subjects. [3] For Yelp, subjects were asked to report how often they go to a restaurant. Those who reported they visit a restaurant at least once a week were assigned to high domain knowledge groups.

Results. Figure 7 summarizes the results. The experiment shows that exploration mode order did not affect the results.[4]. Therefore, whenever we discuss the average result of a treatment group, we aggregate the results of 30 subjects. The same is true for the same

[3] Both questionnaires are available at [3].

[4] The results of every two subgroups of 15 subjects within each treatment group were not statistically significant (ANOVA test, $p < .05$)

Table 2: Examined Datasets.

Dataset	# of Atts	Max # of vals	# of Rating Dimensions	$\|\mathcal{R}\|$	$\|\mathcal{U}\|$	$\|\mathcal{I}\|$
Movielens	12	29	1	100K	943	1682
Yelp	24	13	4	200500	150318	93
Hotel Reviews	8	62	4	35912	15493	879

Table 3: Default Values.

Parameter	Default Value
# of rating maps, k	*3*
# of next-step recommendations, o	3
pruning-diversity factor, l	*3*
length of exploration path	**scenario I**: 7 for all modes **scenario II**: 10 for all modes

treatment group in each dataset and scenario. [5]. This implies the same level of difficulty of both tasks across datasets. Moreover, the results of subjects with the same CS expertise were found to be similar[6] regardless of their domain knowledge.

In all cases, subjects using *Recommendation-Powered* mode achieved the best results, regardless of their domain knowledge or CS expertise. In *Fully-Automated* mode, subjects had no control over the exploration path, and thus identified at best one irregular group in scenario *I*, or 4 insights in scenario *II*. Subjects in *User-Driven* had little information on which operation is the most "interesting". Thus, in most cases, they identified at most one irregular group (resp., three insights). *These observations enable us to conclude that (1)* User-Driven *SDE does not provide enough information to guide users effectively, even when they are CS experts; (2)* Fully-Automated *SDE is not flexible enough, as users cannot intervene to modify the paths; (3)* Recommendation-Powered *SDE is helpful to users regardless of their CS expertise. This validates the need for an iterative partially-guided SDE, regardless of the task in hand and users' domain knowledge.*

# of steps. We vary the number of exploration steps, examining their effect on recall. To this end, for each mode, we asked 30 subjects to use SubDEx for both scenarios, without limiting the number of steps. We provide the results obtained for Movielens, and omit those for Yelp, as they were similar. The results are depicted in Figure 8. Here again, subjects using the *Recommendation-Powered* mode achieved the best results, regardless of the task at hand.

In what follows, we omit the results obtained for the second scenario, as they are similar to the first scenario.

5.2.2 Quality of Recommendations. To examine the quality of SubDEx's next-action recommendations, we compare the results obtained for different baselines (as described in Section 5.1). The set of rating maps displayed in each step is fixed across all baselines. W.l.o.g., we generate exploration paths with the *Fully-Automated* mode, and measure the number of identified irregular groups.

The results are depicted in Table 4. The standard deviation in all cases is < 0.2. Best results were obtained with SubDEx's recommendations. This stems from the fact that both SDD and Qagview always produce "drill-down" recommendations. Namely, each next-action operation focuses on a subset of the examined rating group. However, to identify more than one irregular group, a "roll-up" operation is needed. This demonstrates that SDE (like general EDA) requires more than just summarization and drill-down view exploration. Thus, off-the-shelf summarization and view exploration are ill-suited in our setting. Additionally, due to the modular nature of SubDEx the *Recommendation Builder* may be replaced with alternative implementations, yielding personalized recommendations using logs of previous operations [23, 42], or user feedback [17, 33].

Table 4: Quality of recommendations. We report the average number of correctly identified irregular groups, while using three baselines to produce recommendations.

Baseline	Movielens	Yelp
SubDEx	0.9	0.8
SDD	0.6	0.4
Qagview	0.7	0.5

5.2.3 Parameter Tuning. We validate our formulation for selecting a set of rating maps. To this end, we generate exploration paths with the *Fully-Automated* mode, to fix the next-action operations, and examine the effect of different parameters on the selected maps.

Utility vs. diversity. We vary the value of the pruning-diversity parameter l and examine its effect on utility and diversity. Recall that when l=1, rating maps with the highest DW are selected. As l increases, diversity increases, at the expense of utility. We measure diversity using EMD. When diversity increases, rating maps of different attributes are more likely to be chosen. To demonstrate that, we report the number of distinct attributes shown, and the average diversity score of the rating maps in each step. Recall that the default number of steps in an exploration path in scenario *I* is 7, and the number of rating maps in each step is 3 (Table 3). Thus, the possible number of distinct attributes is bounded by 21 (or the number of attributes in the dataset). The results are depicted in Table 5. As expected, when l increases, the diversity increases. As a consequence, users see more attributes, and hence more facets of the data. However, this comes at the cost of utility, as less "interesting" rating maps are shown. Hence, the choice of l clearly balances utility and diversity. As we shell see, l also affects running times.

We further report the average number of irregular groups identified by subjects examining utility-only or diversity-only exploration paths. To this end, we asked 30 subjects to examine paths obtained with the *Fully-Automated* mode. Half of the exploration paths were generated using utility-only and the others using diversity-only for both Movielens and Yelp. Table 6 summarizes the results. Observe that in both datasets, subjects examining the utility-only paths have succeeded in identifying more irregular groups than for the diversity-only paths. This result is not surprising, since high-utility maps are more likely to reveal irregular patterns in the data. For the second scenario, on the other hand, examining various facets of the data via diversity-only paths was preferable. This suggests that our framework could be tuned according to the task at hand.

[5] The difference between the results for the same treatment group in different datasets were not statistically significant (ANOVA test, $p < .05$)
[6] i.e., not statistically significant (ANOVA test, $p < .05$).

Scenario *I*

Movielens

	High Domain Knowledge	Low Domain Knowledge
High CS Expertise	*UD*: 0.8, *RP*: 1.4	*UD*: 0.7 , *RP*: 1.5
Low CS Expertise	*RP*: 1.3, *FA*: 0.9	*RP*: 1.4, *FA*: 0.8

Yelp

	High Domain Knowledge	Low Domain Knowledge
High CS Expertise	*UD*: 0.6, *RP*: 1.4	*UD*: 0.7 , *RP*: 1.3
Low CS Expertise	*RP*: 1.3, *FA*: 0.7	*RP*: 1.2,*FA*: 0.8

Scenario *II*

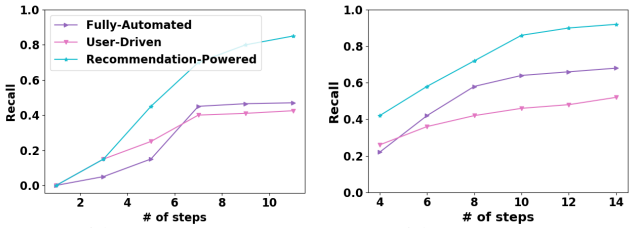
Movielens

	High Domain Knowledge	Low Domain Knowledge
High CS Expertise	*UD*: 2.2, *RP*: 4.1	*UD*: 2.4 ,*RP*: 4.0
Low CS Expertise	*RP*: 4.3 , *FA*: 3.1	*RP*: 4.2 , *FA*: 3.3

Yelp

	High Domain Knowledge	Low Domain Knowledge
High CS Expertise	*UD*: 2.4, *RP*: 4.4	*UD*: 2.3, *RP*: 4.2
Low CS Expertise	*RP*: 4.4 , *FA*: 3.3	*RP*: 4.3, *FA*: 3.4

Figure 7: Exploration guidance results. We report the average number of identified irregular groups/insights using three modes: *User-Driven* (UD), *Recommendation-Powered* (RP), and *Fully-Automated* (FA). The average standard deviation is for scenario *I* (resp. *II*) is 0.2 (resp. 0.4).



(a) Scenario *I*. **(b) Scenario *II*.**

Figure 8: Recall as a function of exploration steps.

Table 5: Utility and diversity experiment.

Utility/Diversity	*Movielens*	*Yelp*
Utility-Only	attributes = 4 utility = 25.2 diversity = 0.02	attributes = 6 utility = 26.1 diversity = 0.03
$l = 2$	attributes = 6 utility = 22.4 diversity = 0.05	attributes = 10 utility = 23.4 diversity = 0.06
$l = 3$	attributes = 9 utility = 19.1 diversity = 0.09	attributes = 15 utility = 20.1 diversity = 0.09
Diversity-Only	attributes = 12 utility = 14.8 diversity= 0.11	attributes = 19 utility = 15.5 diversity = 0.11

Table 6: Avg # of identified irregular groups[7].

Dataset	Utility-only	Diversity-only
Movielens	1.4	0.6
Yelp	1.3	0.6

# of rating dimensions. To examine the effect of DW, we report the number of rating maps per rating dimension, with/without using dimension weights (Figure 9). We omit Movielens as it only has one rating dimension. The results show that the weights help balance the number of rating maps from each dimension. Without weights, rating maps from a single dimension can be shown frequently at the cost of other dimensions. *This demonstrates that the DW utility scores ensure rating maps of different rating dimensions are selected.*

Utility criteria. . We study the individual contribution of the criteria used in computing utility, and their aggregation. To this end, we examined different variants of utility functions, using only one utility criteria or the average aggregation function. We examined exploration paths generated by injecting each of these utility variants into the *Fully-Automated* mode and measure the number of identified irregular groups. Due to space limitation, we summarize our main findings. The results indicate that using only a single measure for defining utility (no matter the tested measure) is inferior. This implies that the combination of multiple interestingness measures is necessary. The results of employing the average aggregation function instead of maximum are also inferior, suggesting that maximum is a preferable aggregation function for our setting.

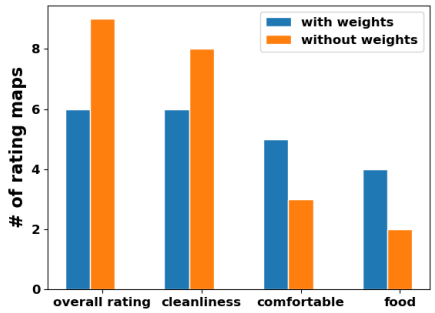


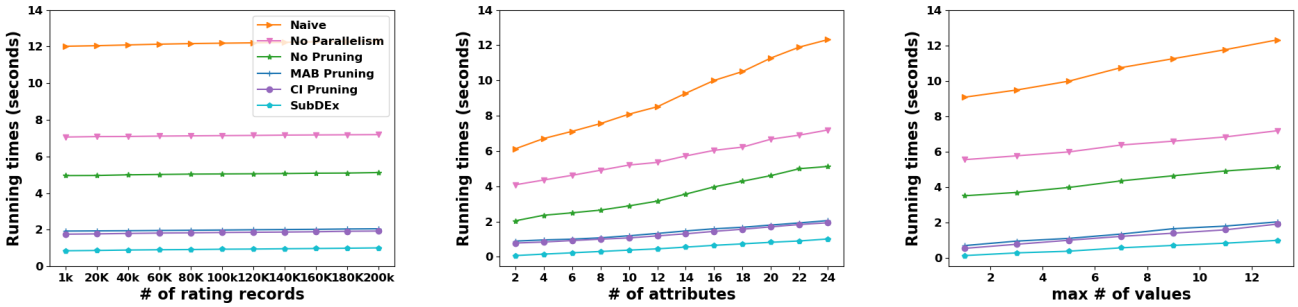
Figure 9: Rating dimensions experiment.

5.3 Scalability Evaluation

To examine the scalability of our solution and the marginal contribution of each of the proposed optimizations, we report the running times of each baseline algorithm described in Section 5.1. The running time of a step is measured between the time an operation is picked and the time the resulting rating maps and next-step recommendations are displayed. We examine paths generated with *Fully-Automated* for Scenario *I* on the Yelp dataset, and report the average processing time across all steps of all paths. We omit results for the second scenario and for other datasets and exploration modes, as they demonstrated similar trends.

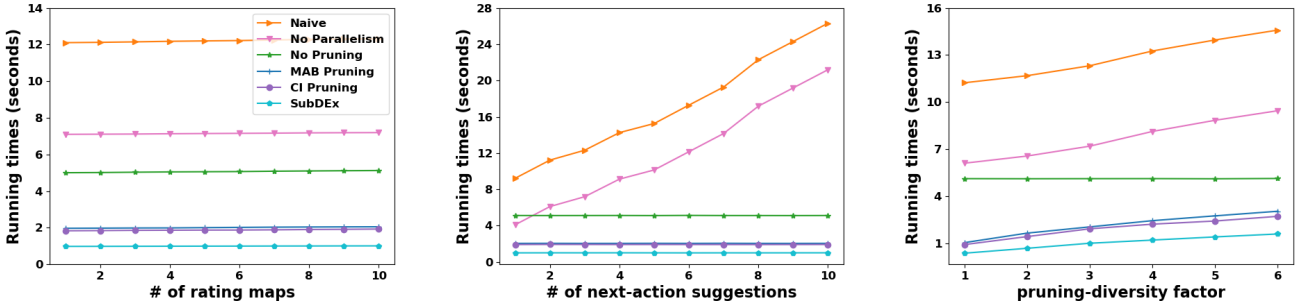
Data Properties. We examine the effect of data properties on running times. We report the average results obtained over 10 runs.

Database size. We vary the database size by randomly sampling reviewers. For each reviewer, we extract her rating records. We also examined the option of sampling items and found similar results. The results are depicted in Figure 10(a). In all cases, SubDEx runs in less than 1 second. The database size has little effect on running times. This can be explained by the number of possible rating maps and next-step operations, which are determined by the number of



(a) # of rating records. (b) # of attributes. (c) # of attribute-values.

Figure 10: Running times as a function of different data properties.



(a) # of rating maps. (b) # of recommendations. (c) pruning-diversity factor.

Figure 11: Running times as a function of different system parameters.

attributes and values. As the records were randomly chosen, there is no difference between different fractions of the data.

# of attributes and # of attribute-values We vary the # of attributes (akin to # GroupBys or # of rating maps) and # of attribute-values (akin to # of next-step operations) and measure running time. To do that, we randomly omit reviewer and item attributes (resp., attribute-values). Results are shown in Figure 10(b) and (c). *For all baselines, we see a near-linear growth in running times.*

System Parameters. We vary the system parameters and examine their effect on running time. We use the full Yelp dataset.

# of rating maps We examine how the number of rating maps displayed in each step affects performance. The results are depicted in Figure 11(a). *In all cases, we see almost no change in running times when the number of rating maps increases.* This result is expected, since the pruning-diversity factor is fixed, and hence the same overall number of rating maps is examined.

# of recommendations We vary the number of recommendations. The results are shown in Figure 11(b). *Due to parallelism, we observe almost no change in the running times of SubDEx and its variants that use parallelism, as the number of recommendations increases.* This stems from the fact that the *Recommendation Builder* may process multiple rating maps simultaneously, producing corresponding recommendations for each rating map. On the other hand, the *No Parallelism* and *Naïve* baselines exhibit a linear growth in running times as a function of the number of recommendations, as they process one rating map at a time. The limit of the number of rating maps that can be processed simultaneously is the number of available cores. However, previous work has shown that a reasonable number of recommendations to show at each point is typically 3 [42], implying that a rather small number of cores is sufficient.

The pruning-diversity factor We examine the effect of the pruning-diversity factor l on running times. Recall that this factor dictates how many rating maps are discarded. When l increases, fewer rating maps are being pruned. The results are depicted in Figure 11(c). Observe that in all baselines that use pruning, l has a great effect on running times. As discussed in Section 5.2.3, there is a trade-off between running times, diversity, and utility. *Our experiments show that setting this parameter to* 3 *yields reasonable running times while ensuring high diversity and utility.*

6 CONCLUSION

We presented SubDEx a dedicated tool for Subjective Data Exploration (SDE). We motivated the practical need for such a tool and explained the additional needs, beyond the requirements of regular data exploration, that require tailored solutions. We proposed pruning optimization techniques to enable interactive running times. Our extensive experimental study on various datasets validates our formulation and demonstrates the scalability of our solution.

We are pursuing the extension of our work to support personalized exploration including accounting for indirect observations such as mouse tracking. We are also considering the application of machine learning techniques such as in [11, 47], to enable task-specific global optimizations yielding new exploration paths.

Acknowledgment. This work has been partially funded by the European Union's Horizon 2020 research and innovation program

(grant agreement No 863410), the Israel Science Foundation, the Binational US-Israel Science Foundation, Tel Aviv University Data Science center, and eBay Israel.

REFERENCES

[1] 2020. Amazon Mechanical Turk. https://www.mturk.com/.
[2] 2020. Datafiniti Dataset. https://www.kaggle.com/datafiniti/hotel-reviews.
[3] 2020. Git Repository. https://github.com/subjectiveDataExploration/Exploring-Ratings-in-Subjective-Databases.git.
[4] 2020. Kaggle. https://www.kaggle.com/.
[5] 2020. MovieLens 100K Dataset. https://grouplens.org/datasets/movielens/100k/.
[6] 2020. Yelp Dataset. https://www.yelp.com/dataset.
[7] Sofiane Abbar, Sihem Amer-Yahia, Piotr Indyk, and Sepideh Mahabadi. 2013. Real-time recommendation of diverse related articles. In *Proceedings of the 22nd international conference on World Wide Web*. 1–12.
[8] Deepak Agarwal, Dhiman Barman, Dimitrios Gunopulos, Neal E Young, Flip Korn, and Divesh Srivastava. 2007. Efficient and effective explanation of change in hierarchical summaries. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*.
[9] Sihem Amer-Yahia, Sofia Kleisarchaki, Naresh Kumar Kolloju, Laks VS Lakshmanan, and Ruben H Zamar. 2017. Exploring rated datasets with rating maps. In *Proceedings of the 26th International Conference on World Wide Web*. 1411–1419.
[10] S. Amer-Yahia, T. Milo, and B. Youngmann. 2021. SubDEx: Exploring Ratings in Subjective Databases. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*.
[11] Ori Bar El, Tova Milo, and Amit Somech. 2020. Automatically generating data exploration sessions using deep reinforcement learning. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1527–1537.
[12] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. 2013. Recommender systems survey. *Knowledge-based systems* (2013).
[13] Sébastian Bubeck, Tengyao Wang, and Nitin Viswanathan. 2013. Multiple identifications in multi-armed bandits. In *International Conference on Machine Learning*. 258–265.
[14] David Carmel, Vanja Josifovski, and Yoelle Maarek. 2011. User modeling for web applications. In *Proceedings of the fourth ACM international conference on Web search and data mining*. 7–8.
[15] Varun Chandola and Vipin Kumar. 2007. Summarization–compressing data into an informative representation. *Knowledge and Information Systems* 12, 3 (2007), 355–378.
[16] Mahashweta Das, Sihem Amer-Yahia, Gautam Das, and Cong Yu. 2011. Mri: Meaningful interpretations of collaborative ratings. *VLDB* (2011).
[17] Kyriaki Dimitriadou, Olga Papaemmanouil, and Yanlei Diao. 2016. AIDE: an active learning-based approach for interactive data exploration. *IEEE Transactions on Knowledge and Data Engineering* 28, 11 (2016), 2842–2856.
[18] Punit R Doshi, Elke A Rundensteiner, and Matthew O Ward. 2003. Prefetching for visual data exploration. In *Eighth International Conference on Database Systems for Advanced Applications, 2003.(DASFAA 2003). Proceedings.* IEEE, 195–202.
[19] Marina Drosou, HV Jagadish, Evaggelia Pitoura, and Julia Stoyanovich. 2017. Diversity in big data: A review. *Big data* (2017).
[20] Marina Drosou and Evaggelia Pitoura. 2013. Ymaldb: exploring relational databases via result-driven recommendations. *The VLDB Journal* 22, 6 (2013), 849–874.
[21] Fan Du, Catherine Plaisant, Neil Spring, and Ben Shneiderman. 2017. Finding similar people to guide life choices: Challenge, design, and evaluation. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 5498–5544.
[22] Philipp Eichmann, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. 2020. Idebench: A benchmark for interactive data exploration. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1555–1569.
[23] Magdalini Eirinaki, Suju Abraham, Neoklis Polyzotis, and Naushin Shaikh. 2013. Querie: Collaborative database exploration. *IEEE Transactions on knowledge and data engineering* 26, 7 (2013), 1778–1790.
[24] Kareem El Gebaly, Parag Agrawal, Lukasz Golab, Flip Korn, and Divesh Srivastava. 2014. Interpretable and informative explanations of outcomes. *Proceedings of the VLDB Endowment* (2014).
[25] Sara Evensen, Aaron Feng, Alon Halevy, Jinfeng Li, Vivian Li, Yuliang Li, Huining Liu, George Mihaila, John Morales, Natalie Nuno, et al. 2019. Voyageur: An experiential travel search engine. In *The World Wide Web Conference*. 3511–5.
[26] Piero Fraternali, Davide Martinenghi, and Marco Tagliasacchi. 2012. Top-k bounded diversification. In *SIGMOD*.
[27] Jerome H Friedman and John W Tukey. 1974. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on computers* 100, 9 (1974), 881–890.
[28] Sreenivas Gollapudi and Aneesh Sharma. 2009. An axiomatic approach for result diversification. In *WWW*. 381–390.
[29] Teofilo F Gonzalez. 1985. Clustering to minimize the maximum intercluster distance. *Theoretical computer science* 38 (1985), 293–306.
[30] David Gotz and Zhen Wen. 2009. Behavior-driven visualization recommendation. In *Proceedings of the 14th international conference on Intelligent user interfaces*. 315–324.
[31] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. 2000. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference*

on Computer supported cooperative work. 241–250.

[32] Robert J Hilderman and Howard J Hamilton. 2013. *Knowledge discovery and measures of interest*. Vol. 638. Springer Science & Business Media.

[33] Enhui Huang, Liping Peng, Luciano Di Palma, Ahmed Abdelkafi, Anna Liu, and Yanlei Diao. 2018. Optimization for active learning-based interactive database exploration. (2018).

[34] Clayton J Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*.

[35] Manas Joglekar, Hector Garcia-Molina, and Aditya Parameswaran. 2017. Interactive data exploration with smart drill-down. *IEEE Transactions on Knowledge and Data Engineering* (2017).

[36] Albert Kim, Eric Blais, Aditya Parameswaran, Piotr Indyk, Sam Madden, and Ronitt Rubinfeld. 2015. Rapid sampling for visualizations with ordering guarantees. In *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, Vol. 8. NIH Public Access, 521.

[37] Nick Koudas, Chen Li, Anthony KH Tung, and Rares Vernica. 2006. Relaxing join and selection queries. In *Proceedings of the 32nd international conference on Very large data bases*. 199–210.

[38] Doris Jung-Lin Lee, Himel Dev, Huizi Hu, Hazem Elmeleegy, and Aditya Parameswaran. 2019. Avoiding drill-down fallacies with VisPilot: assisted exploration of data subsets. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*.

[39] Yuliang Li, Aaron Feng, Jinfeng Li, Saran Mumick, Alon Halevy, Vivian Li, and Wang-Chiew Tan. 2019. Subjective databases. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1330–1343.

[40] Patrick Marcel, Nicolas Labroche, and Panos Vassiliadis. 2019. Towards a benefit-based optimizer for Interactive Data Analysis.

[41] Zhengjie Miao, Yuliang Li, Xiaolan Wang, and Wang-Chiew Tan. 2020. Snippext: Semi-supervised opinion mining with augmented data. In *Proceedings of The Web Conference 2020*. 617–628.

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

[43] Behrooz Omidvar-Tehrani and Sihem Amer-Yahia. 2019. User Group Analytics Survey and Research Opportunities. *IEEE Transactions on Knowledge and Data Engineering* (2019).

[44] Lu Qin, Jeffrey Xu Yu, and Lijun Chang. 2012. Diversifying Top-k Results. (2012).

[45] Senjuti Basu Roy, Sihem Amer-Yahia, Ashish Chawla, Gautam Das, and Cong Yu. 2010. Constructing and exploring composite items. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, Ahmed K. Elmagarmid and Divyakant Agrawal (Eds.). ACM, 843–854.

[46] Sunita Sarawagi. 1999. Explaining differences in multidimensional aggregates. In *VLDB*, Vol. 99. 7–10.

[47] Mariia Seleznova, Behrooz Omidvar-Tehrani, Sihem Amer-Yahia, and Eric Simon. 2020. Guided Exploration of User Groups. *Proc. VLDB Endow.* 13, 9 (2020), 1469–1482.

[48] Robert J Serfling. 1974. Probability inequalities for the sum in sampling without replacement. *The Annals of Statistics* (1974), 39–48.

[49] Tarique Siddiqui, Albert Kim, John Lee, Karrie Karahalios, and Aditya Parameswaran. 2016. Effortless data exploration with zenvisage: an expressive and interactive visual analytics system. *arXiv preprint arXiv:1604.03583* (2016).

[50] Manish Singh, Michael J Cafarella, and HV Jagadish. 2016. DBExplorer: Exploratory Search in Databases.. In *EDBT*. 89–100.

[51] Amit Somech, Tova Milo, and Chai Ozeri. 2019. Predicting" What is Interesting" by Mining Interactive-Data-Analysis Session Logs.. In *EDBT*. 456–467.

[52] Wang-Chiew Tan. 2020. Unleashing the Power of Subjective Data: Managing Experiences as First-Class Citizens. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*. 3610.

[53] Manasi Vartak, Silu Huang, Tarique Siddiqui, Samuel Madden, and Aditya Parameswaran. 2017. Towards visualization recommendation systems. *ACM SIGMOD Record* 45, 4 (2017), 34–39.

[54] Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis. 2015. Seedb: Efficient data-driven visualization recommendations to support visual analytics. In *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, Vol. 8. NIH Public Access, 2182.

[55] Michael Vollmer, Lukasz Golab, Klemens Böhm, and Divesh Srivastava. 2019. Informative Summarization of Numeric Data. In *Proceedings of the 31st International Conference on Scientific and Statistical Database Management*.

[56] Xiaolan Wang, Yoshihiko Suhara, Natalie Nuno, Yuliang Li, Jinfeng Li, Nofar Carmeli, Stefanos Angelidis, Eser Kandogann, and Wang-Chiew Tan. 2020. ExtremeReader: An interactive explorer for customizable and explainable review summarization. In *Companion Proceedings of the Web Conference 2020*. 176–180.

[57] Abdul Wasay, Xinding Wei, Niv Dayan, and Stratos Idreos. 2017. Data canopy: Accelerating exploratory statistical analysis. In *Proceedings of the 2017 ACM International Conference on Management of Data*. 557–572.

[58] Yuhao Wen, Xiaodan Zhu, Sudeepa Roy, and Jun Yang. 2018. Interactive summarization and exploration of top aggregate query answers. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*. NIH Public Access.

[59] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2015. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE transactions on visualization and computer graphics* (2015).

[60] Ke Yang, Vasilis Gkatzelis, and Julia Stoyanovich. 2019. Balanced Ranking with Diversity Constraints. In *IJCAI*. 6035–6042.

[61] Cong Yu, Laks Lakshmanan, and Sihem Amer-Yahia. 2009. It takes variety to make a world: diversification in recommender systems. In *EDBT*. 368–378.

[62] Meike Zehlike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, and Ricardo Baeza-Yates. 2017. Fa* ir: A fair top-k ranking algorithm. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*.

[63] Xiong Zhang, Jonathan Engel, Sara Evensen, Yuliang Li, Çağatay Demiralp, and Wang-Chiew Tan. 2020. Teddy: A System for Interactive Review Analysis. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[64] Mengyu Zhou, Wang Tao, Ji Pengxin, Han Shi, and Zhang Dongmei. 2020. Table2Analysis: Modeling and Recommendation of Common Analysis Patterns for Multi-Dimensional Data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 320–328.

[65] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*. 22–32.