MRI: Meaningful Interpretations of Collaborative Ratings[*]

Mahashweta Das[‡], Sihem Amer-Yahia[†], Gautam Das[‡], Cong Yu[††]
[‡]University of Texas at Arlington; [†]Qatar Computing Research Institute; [††]Google Research
[‡]{mahashweta.das@mavs, gdas@cse}.uta.edu,
[†]sihemameryahia@acm.org, [††]congyu@google.com

ABSTRACT

Collaborative rating sites have become essential resources that many users consult to make purchasing decisions on various items. Ideally, a user wants to quickly decide whether an item is desirable, especially when many choices are available. In practice, however, a user either spends a lot of time examining reviews before making an informed decision, or simply trusts overall rating aggregations associated with an item. In this paper, we argue that neither option is satisfactory and propose a novel and powerful third option, *Meaningful Ratings Interpretation (MRI)*, that automatically provides a meaningful interpretation of ratings associated with the input items. As a simple example, given the movie "Usual Suspects," instead of simply showing the average rating of 8.7 from all reviewers, MRI produces a set of *meaningful* factoids such as "male reviewers under 30 from NYC love this movie". We define the notion of meaningful interpretation based on the idea of data cube, and formalize two important sub-problems, *meaningful description mining* and *meaningful difference mining*. We show that these problems are NP-hard and design randomized hill exploration algorithms to solve them efficiently. We conduct user studies to show that MRI provides more helpful information to users than simple average ratings. Performance evaluation over real data shows that our algorithms perform much faster and generate equally good interpretations as brute-force algorithms.

1. INTRODUCTION

Collaborative rating sites drive a large number of decisions today. For example, online shoppers rely on ratings on `Amazon` to purchase a variety of goods such as books and electronics, and movie-goers use `IMDb` to find out about a movie before renting it. Typically, the number of ratings associated with an item (or a set of items) can easily reach hundreds or thousands, thus making reaching a decision cumbersome. For example, on the review site `Yelp`, a not-so-popular restaurant *Joe's Shanghai* received nearly a thousand ratings, and more popular restaurants routinely exceed that number by many multipliers. Similarly, the movie *The Social Network* has received $42,000+$ ratings on `IMDb` after being released for just two months!

To cope with the overwhelming amount of information, a user can either spend a lot of time examining ratings and reviews before making an informed decision (*maximalist option*), or simply go with overall rating aggregations, such as average, associated with an item (*minimalist option*). Not surprisingly, most users choose the latter due to lack of time and forgo the rich information embedded in ratings and in reviewers' profiles. Typically, average ratings are generated for a few pre-defined populations of reviewers (e.g., average among movie critics). In addition, aggregated ratings are only available for one item at a time, and therefore a user cannot obtain an understanding of a set of items of interest, such as all movies by a given director.

In this paper, we aim to help users make better decisions by providing *meaningful interpretations* of ratings of items of interest, by leveraging metadata associated with items and reviewers in online collaborative rating sites. We call this problem *meaningful rating interpretation* (MRI), and define two sub-problems: *meaningful description mining* (DEM) and *meaningful difference mining* (DIM).

Given a set of items, the first problem, *meaningful description mining*, aims to identify groups of reviewers who share similar ratings on the items, with the added constraint that each group consists of reviewers who are describable with a subset of their attributes (i.e., gender, age, etc.). The description thus returned to the user contains a small list of *meaningfully labelled* groups of reviewers and their ratings about the item, instead of a single monolithic average rating. This added information can help users judge items better by surfacing inherent reviewers' biases for the items. For example, the movie *Titanic* may have a very high overall average rating, but it is really the group of *female reviewers under the age of* 20 who give it very high ratings and raise the average. A user can then make informed decisions about items based on whether she tends to agree with that group.

The second problem, *meaningful difference mining*, aims to help users better understand controversial items by identifying groups of reviewers who consistently disagree on those items, again with the added constraint that each group is

[*]The work of Mahashweta Das and Gautam Das is partially supported by NSF grants 0812601, 0915834, 1018865, a NHARP grant from the Texas Higher Education Coordinating Board, and grants from Microsoft Research and Nokia Research.

Proceedings of the VLDB Endowment, Vol. 4, No. 11
Copyright 2011 VLDB Endowment 2150-8097/11/08... $ 10.00.

described with a meaningful label. For the movie *Titanic*, we can see that two groups of reviewers, *females under 20* and *males between 30 and 45*, are in consistent disagreement about it: the former group loves it while the latter does not.

We emphasize that while the examples above all involve a single item, both description mining and difference mining can be applied to a set of items with a common feature. For example, we can apply them to all movies directed by *Woody Allen* and help users learn some meaningful trends about *Woody Allen* as a director. The algorithms we describe in this paper apply equally whether we are analyzing the ratings of a single item or a set of items.

2. PRELIMINARIES

We model a collaborative rating site $\mathcal{D}$ as a triple $\langle \mathcal{I}, \mathcal{U}, \mathcal{R} \rangle$, representing the sets of items, reviewers and ratings respectively. Each rating $r \in \mathcal{R}$ is itself a triple $\langle i, u, s \rangle$, where $i \in \mathcal{I}$, $u \in \mathcal{U}$, and $s \in [1, 5]$ is the integer rating that reviewer u has assigned to item i[1]. Furthermore, $\mathcal{I}$ is associated with a set of attributes, denoted $\mathcal{I}_A = \{ia_1, ia_2, \ldots\}$, and each item $i \in \mathcal{I}$ is a tuple with $\mathcal{I}_A$ as its schema. In another word, $i = \langle iv_1, iv_2, \ldots \rangle$, where each iv_j is a value for attribute ia_j. 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. As a result, each rating, $r = \langle i, u, s \rangle$, is a tuple, $\langle iv_1, iv_2, \ldots, uv_1, uv_2, \ldots, s \rangle$, that concatenates the tuple for i, the tuple for u, and the numerical rating score s. The set of all attributes (including both item and reviewer attributes) is denoted as $A = \{a_1, a_2, \ldots\}$.

Item attributes are typically provided by the rating site. For example, restaurants on `Yelp` are described with attributes such as *Cuisine* (e.g., Thai, Sushi), *Attire* (e.g., Formal, Casual). Movies on `IMDb` are described with *Title*, *Genre* (e.g., Drama, Animation), *Actors*, *Directors*. An item attribute can be multi-valued (e.g., a movie can have many actors). Reviewer attributes include mostly demographics such as *Age*, *Gender*, *ZipCode* and *Occupation*. Such attributes can either be provided to the site by the reviewer directly as in `MovieLens`, or obtained from social networking sites such as `Facebook` as their integration into content sites becomes increasingly common. In this paper, we focus on item ratings describable by reviewer attributes. Our ideas can be easily extended to explain reviewer ratings by item attributes.

We model the notion of *group* based on data cube [6]. Intuitively, a group is a set of ratings described by a set of attribute value pairs shared among the reviewers and the items of those ratings. A group can also be interpreted as a selection query condition. More formally, a group description is defined as $c = \{\langle a_1, v_1 \rangle, \langle a_2, v_2 \rangle, \ldots\}$, where each $a_i \in A$ (where A is the set of all attributes as introduced earlier) and each v_i is a value for a_i. For example, $\{\langle \texttt{genre}, \texttt{war} \rangle, \langle \texttt{location}, \texttt{nyc} \rangle\}$ describes a group representing all ratings of "war" movies by reviewers in "nyc." The total number of groups that can exist is given by $n = \prod_{i=1}^{|A|}(|\langle a_i, v_j \rangle| + 1)$, where $|A|$ is the cardinality of the set of attributes and $|\langle a_i, v_j \rangle|$ is the number of distinct v_j values each attribute a_i can take. When the ratings are viewed as tuples in a data warehouse, this notion of group coincides with the definition of cuboids in the data cube literature. Here, we take the view that, unlike unsupervised clustering of ratings, ratings grouped this way are much more meaningful to users, and form the foundation for meaningful rating interpretations. We now define three essential characteristics of the group.

[1]For simplicity, we convert ratings at different scales into the range $[1, 5]$.

First, **coverage**: Given a rating tuple $r = \langle v_1, v_2, \ldots, v_k, s \rangle$, where each v_i is a value for its corresponding attribute in the schema A, and a group $c = \{\langle a_1, v_1 \rangle, \langle a_2, v_2 \rangle, \ldots, \langle a_n, v_n \rangle\}, n \leq k$, we say c covers r, denoted as $r \lessdot c$, iff $\forall i \in [1, n]$, $\exists r.v_j$ such that v_j is a value for attribute $c.a_i$ and $r.v_j = c.v_i$. For example, the rating $\langle \texttt{female}, \texttt{nyc}, \texttt{cameron}, \texttt{winslet}, 4.0 \rangle$ is covered by the group $\{\langle \texttt{gender}, \texttt{female} \rangle, \langle \texttt{location}, \texttt{nyc} \rangle, \langle \texttt{actor}, \texttt{winslet} \rangle\}$.

Second, **relationship between groups**: A group c_1 is considered an ancestor of another group c_2, denoted $c_1 \supset c_2$, iff $\forall j$ where $\langle a_j, v_j \rangle \in c_2$, $\exists \langle a_j, v'_j \rangle \in c_1$, such that $v_j = v'_j$, or v'_j semantically contains v_j according to the domain hierarchy. For example, the group of ratings g_1 by reviewers who live in Michigan is a parent of the group of ratings g_2 by reviewers who live in Detroit, since Detroit is located in Michigan according to the location hierarchy[2].

Third, **recursive coverage**: Given a rating tuple r and a group c, we say c recursively covers r iff $\exists c'$, such that, $c \supset c', r \lessdot c'$. For example, $\langle \texttt{female}, \texttt{nyc}, \texttt{cameron}, \texttt{winslet}, 4.0 \rangle$ is recursively covered by $\{\langle \texttt{gender}, \texttt{female} \rangle, \langle \texttt{location}, \texttt{USA} \rangle, \langle \texttt{actor}, \texttt{winslet} \rangle\}$. For the rest of the paper, we use the term coverage to mean recursive coverage for simplicity, unless otherwise noted.

2.1 Meaningful Rating Interpretation

When the user is exploring an item (or a set of items) I, our goal is to meaningfully interpret the set of ratings for I, denoted R_I. Given a group c, the set of ratings in R_I that are covered by c are denoted as $c_{R_I} = \{r \mid r \in R_I \wedge r \lessdot c\}$. Similar to data cubes, the set of all possible groups form a lattice of n nodes, where the nodes correspond to groups and the edges correspond to parent/child relationships. Note that, for a given I, there are many groups not covering any rating from R_I. Let n' denote the total number of groups covering at least one rating. Solving the MRI problem is therefore to *quickly identify "good" groups that can help users understand ratings more effectively.*

Before introducing the problem formally, we first present a running example, shown in Figure 1, which will be used throughout the rest of the paper.

Example 1. Consider the use case where we would like to explain all ratings of the movie (item) *Toy Story*, by identifying describable groups of reviewers sharing common rating behaviors. As in data cube analysis, we adopt a lattice structure to group all ratings, where each node in the lattice corresponds to a group containing rating tuples sharing the set of common attribute value pairs, and each edge between two nodes corresponds to the parent/ child relationship. Figure 1 illustrates a partial lattice for *Toy Story*, where we have four reviewer attributes to analyze[3]: *gender* (G), *age* (A), *location* (L) and *occupation* (O). For simplicity, exactly one distinct value per attribute is shown in the

[2]Those domain hierarchies are essentially dimension tables and we assume they are given in our study.

[3]Since there is only one movie in this example, item attributes do not apply here.

example: $\langle \texttt{gender}, \texttt{male} \rangle$, $\langle \texttt{age}, \texttt{young} \rangle$, $\langle \texttt{location}, \texttt{CA} \rangle$ and $\langle \texttt{occupation}, \texttt{student} \rangle$. As a result, the total number of groups in the lattice is 16. Each group (i.e., node in the lattice) maps to a set of rating tuples that are satisfied by the selection condition corresponding to the group label, and the numeric values within each group denotes the total number of ratings and the average rating within the group. For example, the base (bottom) group corresponds to all 452 ratings of *Toy Story*, with an average rating of 3.88, while the double circled group in the center of the lattice corresponds to the 75 ratings provided by 'male & student' reviewers, who collectively gave it an average rating of 3.76. □

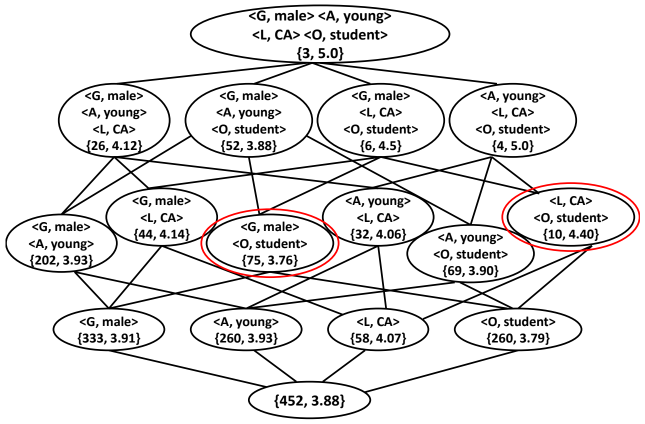

<G, male> <A, young> <L, CA> <O, student> {3, 5.0}
<G, male> <A, young> <L, CA> {26, 4.12}
<G, male> <A, young> <O, student> {52, 3.88}
<G, male> <L, CA> <O, student> {6, 4.5}
<A, young> <L, CA> <O, student> {4, 5.0}
<G, male> <A, young> {202, 3.93}
<G, male> <L, CA> {44, 4.14}
<G, male> <O, student> {75, 3.76}
<A, young> <L, CA> {32, 4.06}
<A, young> <O, student> {69, 3.90}
<L, CA> <O, student> {10, 4.40}
<G, male> {333, 3.91}
<A, young> {260, 3.93}
<L, CA> {58, 4.07}
<O, student> {260, 3.79}
{452, 3.88}

Figure 1: Partial rating lattice for movie *Toy Story* with one distinct value for each attribute; the full lattice contains more nodes with multiple distinct values for each attribute.

Even when there is only a single item, the number of groups associated with its ratings can be too large for a user to browse. The challenge is therefore to identify "good" groups to be highlighted to the user. We define desiderata that such "good" groups should follow:

Desiderata 1: Each group should be easily understandable by the user. While this desiderata is often hard to satisfy through unsupervised clustering of ratings, it is easily enforced in our approach since each group is structurally meaningful and has an associated description that the user can understand.

Desiderata 2: Together, the groups should cover enough ratings in R_I. While ideally we would like all ratings in R_I to be covered, it is often infeasible given the constraint on the number of groups that a user can reasonably go through.

Desiderata 3: Ratings within each group should be as consistent as possible, i.e., should reflect users with similar opinions toward the input item(s). Note that we are referring to opinions within a group instead of opinions across groups. In fact, difference in opinion across groups is the key differentiator between the two sub-problems of MRI, which we will formally define in the next section.

3. PROBLEM DEFINITIONS

We now formally define the two sub-problems of meaningful rating interpretation: *meaningful description mining* (DEM) and *meaningful difference mining* (DIM).

3.1 Meaningful Description Mining

Our first goal is to give a meaningful description of all the ratings over an item set I. We propose to present to the users a small set of meaningfully labelled rating groups (i.e., cuboids), each with their own average ratings. Specifically, we consider three main factors. First, the *number of cuboids*, k, to be presented to the user must be limited, so that users are not overwhelmed with too many cuboids. Second, all cuboids presented to the user must collectively *cover a large enough portion of ratings* for items in I. Third, the returned cuboids must collectively have the *minimum aggregate error*, which we will define next.

Consider a set of ratings R_I over input items in I. For each cuboid c, let $\texttt{avg}(c) = avg_{r_i \lessdot c}(r_i.s)$ (where $r_i.s$ is the score of the i^{th} tuple) be the average numerical score of ratings covered by c. Given a set of cuboids, C, to be returned to the user. We define two formal notions:

Description coverage: Let $C_{R_I} = \{r \mid r \in R_I, \exists c \in C, s.t.\ r \lessdot c\}$, $\texttt{coverage}(C, R_I) = \frac{|C_{R_I}|}{|R_I|}$.

Description error: Let $E_r = \texttt{avg}(|r.s - \texttt{avg}_{c \in C \wedge r \lessdot c}(c)|)$, $\texttt{error}(C, R_I) = \Sigma_{r \in R_I}(E_r)$.

Intuitively, *description coverage* measures the percentage of ratings covered by at least one of the returned cuboids, while *description error* measures how well the group average approximates the numerical score of each individual rating. (When a rating is covered by more than one returned cuboids, we average the errors over all the cuboids that cover the rating.)

Problem 1. *The problem of meaningful description mining (DEM) for a given set of items I and their ratings R_I, identify a set of cuboids C, such that:*

- $\texttt{error}(C, R_I)$ *is minimized, subject to:*
    - $|C| \leq k$;
    - $\texttt{coverage}(C, R_I) \geq \alpha$.

Theorem 1. *The decision version of the problem of meaningful description mining (DEM) is NP-Complete even for boolean databases, where each attribute ia_j in $\mathcal{I}_A$ and each attribute ua_j in $\mathcal{U}_A$ takes either 0 or 1.*

Proof: Please refer to Appendix A.1.1.

3.2 Meaningful Difference Mining

Another important goal of rating interpretation is to identify meaningful groups of ratings where reviewers' opinions on the item(s) are divergent. To accomplish this goal, we start by dividing R_I into two sets $R_I^+ = \{r \mid r \in R_I \wedge r.s \geq \theta^+\}$ and $R_I^- = \{r \mid r \in R_I \wedge r.s \leq \theta^-\}$, where θ^+ and θ^- are thresholds that define whether a rating should be considered positive or negative respectively. Intuitively, θ^+ and θ^- can either be decided statically or dynamically according to the mean and variances of R_I. While setting the thresholds statically is easier computationally, it is not always clear what the thresholds should be. As a result, we follow the dynamic approach and set θ^+ and θ^- to be one standard deviation above and below the mean of R_I respectively.

Given R_I^+, R_I^- and a set of cuboid groups, C, we can now formalize the notion of *balance* as follows:

Balance: Let indicator $I_{(r_1, r_2)} = 1$ if and only if $\exists c \in C$ s.t. $r_1 \lessdot c \wedge r_2 \lessdot c$ (i.e., there is at least one cuboid in C that covers both r_1 and r_2.). We then have $\texttt{balance}(C, R_I^+, R_I^-) = m \times \Sigma_{r_1 \in R_I^+, r_2 \in R_I^-} I_{(r_1,r_2)}$, where $m = \frac{1}{|R_I^+| \times |R_I^-|}$ is the normalization factor that normalizes all balance values into $[0, 1]$.

Intuitively, the notion of *balance* captures whether the positive and negative ratings are "mingled together" (high balance) or "separated apart" (low balance).

PROBLEM 2. *The problem of meaningful difference mining (DIM) for a given set of items I and their ratings R_I (split into R_I^+, R_I^-), identify a set of cuboids C, such that:*

- $\texttt{balance}(C, R_I^+, R_I^-)$ *is minimized, subject to:*
  - $|C| \leq k$;
  - $\texttt{coverage}(C, R_I^+) \geq \alpha \wedge \texttt{coverage}(C, R_I^-) \geq \alpha$.

THEOREM 2. *The decision version of the problem of meaningful difference mining (DIM) is NP-Complete even for boolean databases.*

Proof: Please refer to Appendix A.1.2.

4. ALGORITHMS

In this section, we propose efficient algorithms for both description mining and difference mining tasks.

4.1 Description Mining Algorithms

Given a set I of items and the set R_I of all ratings over I, the description mining task (Section 3.1) aims to identify a set C of cuboids over R_I, such that the aggregate error, $\texttt{error}(C, R_I)$, is minimized, and the size and coverage constraints are satisfied. The baseline approach is to enumerate all possible combinations of cuboids over R_I. We introduce this *Exact Algorithm* first, and later propose a more efficient heuristic algorithm based on *randomized hill exploration.*

Exact Algorithm (E-DEM): This algorithm uses brute-force to enumerate all possible combinations of cuboids to return the *exact* (i.e., optimal) set of cuboids as the rating descriptions. Algorithm 1 illustrates its high level pseudo code. The algorithm consists of two stages. During the first stage, it maps the rating lattice to the ratings of the given item set I. In particular, lattice nodes that do not cover any rating in R_I are not materialized and the average ratings of the remaining lattice nodes are computed. In the second stage, the algorithm looks up all $\binom{n}{k}$ possible sets of cuboids C, where n is the number of lattice nodes remaining after the first stage. The set that minimizes $\texttt{error}(C, R_I)$ such that $|C| \leq k$ and $\texttt{coverage}(C, R_I) \geq \alpha$. Clearly, Algorithm E-DEM is exponential in the number of cuboids and can be prohibitively expensive.

Randomized Hill Exploration Algorithm (RHE-DEM): A common heuristic technique for solving optimization problems similar to our description mining problem is *random restart hill climbing* [12]. A straightforward adoption of this technique involves the following. We first randomly select a set of k cuboids as the starting point. The process then continues by replacing one cuboid in the current set with one of its lattice neighbors[4] not in the set as long as the substitution reduces the aggregate error. The algorithm

Algorithm 1 – E-DEM Algorithm (R_I, k, α) : C

```
- Build the rating lattice of $n'$ cuboids (out of $n$), each of
  which covers at least one tuple from $R_I$.
1: while true do
2:     build set $C \leftarrow$ combinatorics-getnext($n'$, $k$)
3:     if $\texttt{coverage}(C, R_I) \geq \alpha$ then
4:         build list $L \leftarrow$ ( $C$, $\texttt{error}(C, R_I)$ )
5:     end if
6: end while
7: $C \leftarrow$ min $\texttt{error}(C, R_I)$ in $L$
8: return $C$
```

stops when no improvements can be made indicating a *local minima* has been reached. The process is repeated with multiple diverse sets of cuboids to increase the probability of finding the *global minima* that satisfies the constraints.

However, this simple application of hill climbing fails for our description mining task because of the critically important coverage constraint, $\texttt{coverage}(C, R_I) \geq \alpha$. For any given set of cuboids randomly chosen as the starting point, the probability of it satisfying the coverage constraint is fairly small since most cuboids in the lattice cover a small number of ratings. Since the simple hill climbing algorithm cannot optimize for both coverage and aggregate error at the same time, the results produced by the simple hill climbing algorithm often fail the coverage constraint. Hence, a large number of restarts is required before a solution can be found, negating the performance benefits.

To address this challenge, we propose the *Randomized Hill Exploration Algorithm* (RHE-DEM) which first initializes a randomly selected set of k cuboids as the starting point. However, instead of immediately starting to improve the aggregate error, it *explores* the hill to identify nearby cuboid sets that satisfy the coverage constraint. Specifically, RHE-DEM performs iterative improvements on the coverage that lead to a different set of cuboids where the coverage constraint is satisfied. This new cuboid set is then adopted as the starting point for the error optimization with the added condition that an improvement is valid only when the coverage constraint is satisfied. Furthermore, this exploration can advance in multiple directions, producing multiple cuboid sets as new start points based on the single initial cuboid set. Since we found single direction exploration works well in practice, we have not pursued this technique.

The details of the algorithm are shown in Algorithm 2. Intuitively, we begin with the rating lattice constructed on R_I. The algorithm starts by picking k random cuboids to form the initial set C. For each cuboid c_i in C, we swap c_i with each of its neighbors c_j in the lattice, while the other cuboids in C remain fixed, to generate a new combination (i.e., cuboid set). The exploration phase computes $\texttt{coverage}(C, R_I)$ for each obtainable combination of k cuboids, until it finds one that satisfies $\texttt{coverage}(C, R_I) \geq \alpha$. The resulting set then acts as the initial condition for the second phase of the optimization to minimize the aggregate error $\texttt{error}(C, R_I)$. The configuration that satisfies $\texttt{coverage}(C, R_I) \geq \alpha$ and incurs minimum error $\texttt{error}(C, R_I)$ is the *best* rating explanation for item set I.

EXAMPLE 2. Consider the example rating lattice introduced in Example 1 and suppose k=2, α=80%. The complete rating lattice will have many more cuboids than

[4]Two cuboids are neighbors if they are directly connected in the lattice.

Algorithm 2 – RHE-DEM Algorithm (R_I, k, α) : C

```
- Build the rating lattice of $n'$ cuboids (out of $n$), each of
  which covers at least one tuple from $R_I$.
1: $C \leftarrow$ randomly select $k$ of $n'$ cuboids
2: if coverage($C, R_I$) $\geq \alpha$ then
3:    $C \leftarrow$ satisfy-coverage($C, R_I$))
4: end if
5: $C \leftarrow$ minimize-error($C, R_I$))
6: $C' \leftarrow$ best $C$ so far
7: return $C'$

  // method satisfy-coverage ($C, R_I$): $C$
1: while true do
2:    val $\leftarrow$ coverage($C, R_I$)
3:    for each cuboid $c_i$ in $C$, each neighbor $c_j$ of $c_i$ do
4:       $C' \leftarrow C - c_i + c_j$
5:       $val' \leftarrow$ coverage($C', R_I$)
6:       if $val' \geq \alpha$ then
7:          return $C'$
8:       end if
9:    end for
10: end while

  // method minimize-error ($C, R_I$): $C$
1: while true do
2:    val $\leftarrow$ error($C, R_I$)
3:    $\mathcal{C} = \emptyset$
4:    for each cuboid $c_i$ in $C$, each neighbor $c_j$ of $c_i$ do
5:       $C' \leftarrow C - c_i + c_j$
6:       if coverage($C', R_I$) $\geq \alpha$ then
7:          add ($C'$, error($C', R_I$)) to $\mathcal{C}$
8:       end if
9:    end for
10:   let $(C'_m, val'_m) \in \mathcal{C}$ be the pair with minimum error
11:   if $val'_m \geq val$ then
12:      return $C$ // we have found the local minima
13:   end if
14:   $C \leftarrow C'_m$
15: end while
```

what is shown in Figure 1, since there are several other attribute-value pairs such as ⟨gender, female⟩, ⟨age, old⟩, ⟨location, NY⟩, etc. Precisely, the total number of cuboids in the rating lattice for *Toy Story* is n = 17490, of which n' = 1846 have $c_{\mathcal{R}_I} \neq 0$. However, we focus on the example rating lattice having 16 groups to illustrate our description mining algorithms. The exact algorithm will investigate all $\binom{16}{2}$ (or, $\binom{1846}{2}$ for complete rating lattice) possible combinations to retrieve the *best* rating descriptions. On the other hand, the randomized hill exploration algorithm begins by randomly selecting a set of k=2 cuboids, say c_1 = {⟨G, male⟩, ⟨O, student⟩} and c_2 = {⟨L, CA⟩, ⟨O, student⟩} (marked in double circle in Figure 1). Here C_{R_I} = 79, which does not satisfy the constraint coverage(C, R_I) ≥ 80%. Keeping c_2 fixed, the obtainable combinations by swapping c_1 with its parent/child are: {c'_1, c_2}, {c''_1, c_2}, {c'''_1, c_2} and {c''''_1, c_2}, where c'_1 = {⟨G, male⟩}, c''_1 = {⟨O, student⟩}, c'''_1 = {⟨G, male⟩, ⟨O, student⟩, ⟨A, young⟩} and c''''_1 = {⟨G, male⟩, ⟨O, student⟩, ⟨L, CA⟩}. We see c'_1 = {⟨G, male⟩}, c_2 ={⟨L, CA⟩, ⟨O, student⟩} satisfy the coverage constraint. The set {c'_1, c_2} is then used as the initial condition to explore the connected lattice and minimize the description error. RHE-DEM on this partial rating lattice eventually returns the cuboids {⟨G, male⟩} and {⟨O, student⟩} as the rating interpretations who share similar ratings on *Toy Story*. □

4.2 Difference Mining Algorithms

Similar to the description mining task, the task of difference mining (Section 3.2) poses an optimization problem with the goal of, given an item set I, identifying a set C of cuboids with the most divergent opinions regarding the ratings R_I over I (i.e., minimizing the aggregate balance, balance(C, R_I^+, R_I^-)) and satisfying the size and coverage constraints. The difference mining task is even more challenging because computing the optimization objective, balance, is very expensive. We describe this challenge and propose a similar heuristic hill exploration algorithm that leverages the concept of *fundamental region*.

Exact Algorithm (E-DIM): Similar to Algorithm E-DEM, this algorithm uses brute-force to enumerate all possible combinations of cuboids.

Randomized Hill Exploration Algorithm (RHE-DIM): The difference mining problem shares many similar characteristics with the description mining problem. In particular, the measure, aggregate balance, needs to be minimized, while at the same time, a non-trivial constraint, coverage above a threshold, must be satisfied. This makes the direct application of prior heuristic techniques such as hill climbing difficult. As a result, we leverage the same randomized hill exploration technique as introduced in Section 4.1 and propose Algorithm RHE-DIM.

Similar to RHE-DEM, RHE-DIM first initializes a randomly selected set of k cuboids. It explores the search space, in the first phase, to find a new set of cuboids such that the coverage constraint is satisfied. During the second phase, the algorithm iteratively improves the aggregate balance while ensuring that the coverage constraint remains satisfied, until a *local minima* is identified.

Unlike the description mining problem, however, *computing the optimization measure* balance(C, R_I^+, R_I^-) *for the difference mining problem can be very expensive.* When done naively, it involves a quadratic computation that scans all possible pairings of positive and negative ratings, for each set of k cuboids we encounter during the second phase. To address this computational challenge, we introduce the concept of *fundamental region* (FR), which defines core rating sets induced by a set of k cuboids, to aid the computation of balance(C, R_I^+, R_I^-). The idea is inspired by the notion of finest partitioning [1], with the key difference here being the need for keeping track of both positive and negative ratings.

Definition 1. *Given R_I and the set C of k cuboids in the rating lattice, we can construct a k-bit vector signature for each tuple in R_I, where a bit is set to true if the tuple is covered by the corresponding cuboid. A* fundamental region (denoted by F) *is thus defined as the set of ratings that share the same signature. The number of fundamental regions is bounded by $2^k - 1$, but is often substantially smaller.*

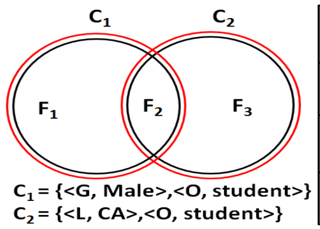
Given the set of fundamental regions, we can compute balance(C, R_I^+, R_I^-) by iterating over all pairs of fundamental regions instead of all pairs of tuples, thus having a significant performance advantage. Specifically, for a self-pair involving a single region F_i, we have balance($C, R_{I\,i}^+, R_{I\,i}^-$) = $F_i(R_I^+) \times F_i(R_I^-)$; for a pair of distinct regions F_i

and F_j sharing at least a common cuboid, we have $\texttt{balance}(C, R^+_{I\,ij}, R^-_{I\,ij}) = F_i(R^+_I) \times F_j(R^-_I) + F_j(R^+_I) \times F_i(R^-_I)$. Finally, we have:

$$\texttt{balance}(C, R^+_I, R^-_I) = m \times (\sum_i \texttt{balance}(C, R^+_{I\,i}, R^-_{I\,i}) + \sum_{ij} \texttt{balance}(C, R^+_{I\,ij}, R^-_{I\,ij})) \quad (1)$$

where m is the normalization factor described in Section 3.2.

EXAMPLE 3. Consider a set $C = \{c_1, c_2\}$ of $k = 2$ cuboids, where $c_1 = \{\langle \texttt{G}, \texttt{male}\rangle, \langle \texttt{O}, \texttt{student}\rangle\}$ and $c_2 = \{\langle \texttt{L}, \texttt{CA}\rangle, \langle \texttt{O}, \texttt{student}\rangle\}$ (marked in double circle in Figure 1). The two cuboids partition the set of 79 ratings (covered by C) into 3 fundamental regions F_1, F_2 and F_3 each having a distinct signature, as shown in Figure 2. The positive and negative rating tuple counts, $F(R^+_I)$ and $F(R^-_I)$ respectively in each region are also presented in Figure 2. By Equation 1, $\texttt{balance}(C, R^+_I, R^-_I)$, can be computed as: $\frac{1}{46\times 33}\times$ (40×29 + 4×2 + 2×2 + (40×2 + 4×29) + (4×2 + 2×2)), based on counts in F_1, F_2, F_3, (F_1, F_2) and (F_2, F_3) respectively. □



F	C_1C_2	Count $F(R^+_I), F(R^-_I)$
F_1	1 0	40, 29
F_2	1 1	4, 2
F_3	0 1	2, 2

Figure 2: Computing $\texttt{balance}(C, R^+_I, R^-_I)$ using Fundamental Regions.

THEOREM 3. *Given R_I and C, $\texttt{balance}(C, R^+_{I\,i}, R^-_{I\,i})$ computed using Equation 1 is equivalent to the one computed using the formula in Section 3.2.*

Proof: Please refer to Appendix A.1.3.

This fundamental region based balance computation involves populating a $min(n_{fr}, |R^+_I|) \times min(n_{fr}, |R^-_I|)$ matrix, where n_{fr} is the number of fundamental regions induced by the set C of k cuboids ($n_{fr} \leq 2^k - 1$), each cell stores the balance between a pair of FRs (or a self pair), and summing over all cells to compute the overall balance. The details are presented in Algorithm 3. Finally, Algorithm RHE-DIM works the same way as RHE-DEM presented in Algorithm 2, with all $\texttt{error}(C, R_I)$ computation being replaced with compute-balance(C, R^+_I, R^-_I) of Algorithm 3.

4.3 Algorithm Discussion

The computational complexity of the description mining and difference mining problems can be viewed as depending on the parameters: R_I (R^+_I, R^-_I), the set of ratings over item set I; n', the number of cuboids in rating lattice covering at least one rating from R_I; and k, the number of cuboids to be presented to user. The exact algorithms E-DEM and E-DIM are exponential in n'. The heuristic algorithms RHE-DEM and RHE-DIM work well in practice (as shown in Section 5); but of course they do not guarantee any sort of worst case behavior, either in running time or in result quality. We note that the performance of RHE-DIM for difference mining is dependent on the computation of the optimization measure, $\texttt{balance}(C, R^+_I, R^-_I)$.

Algorithm 3 − compute-balance(C, R^+_I, R^-_I): v

```
1: for $i'$=1 to $n_{fr}$ do
2:   $F_{i'}(R^+_I, R^-_I) \leftarrow \{count(F_{i'}, R^+_I), count(F_{i'}, R^-_I)\}$
3: end for
4: for i=1 to $n_{fr}$, j=1 to $n_{fr}$ do
5:   pairing-matrix-fr(i, j) $\leftarrow 0$
6: end for
7: for i=1 to $2^k - 1$, j=1 to $2^k - 1$ do
8:   if i = j and pairing-matrix-fr(i, j) = 0 then
9:     pairing-matrix-fr(i, j) $\leftarrow F_i(R^+_I) \times F_i(R^-_I)$
10:  else if i $\neq$ j and
       pairing-matrix-fr(i, j) = 0 and
       $F_i$, $F_j$ belongs to same cuboid in $C$ then
11:    pairing-matrix-fr(i, j) $\leftarrow F_i(R^+_I) \times F_j(R^-_I)$
12:    pairing-matrix-fr(j, i) $\leftarrow F_j(R^+_I) \times F_i(R^-_I)$
13:  end if
14: end for
15: $v \leftarrow$ sum of all non-zero products in pairing-matrix-fr
16: return $v$
```

The naive way of computing the aggregate balance involves a quadratic computation that scans all possible pairings of positive and negative ratings, for each set C of k cuboids, during the second phase. The runtime complexity for balance computation this way is O($k \times |R^+_I| \times |R^-_I|$). The alternate way of using the fundamental regions reduces the complexity since it concerns pairings of positive and negative rating regions, instead of pairings of positive and negative rating tuples. The number of fundamental regions for a set of k cuboids is 2^k-1. Therefore, the reduced running time is given by O($k \times min(2^k - 1, |R^+_I|) \times min(2^k - 1, |R^-_I|)$).

Finally, the notion of partitioning ratings into fundamental regions also motivates us to design *incremental techniques* to speed up the execution time of our difference mining algorithms. Please refer to Appendix A.2 for our incremental algorithms. The implementation of the incremental algorithms are left as part of our future work.

5. EXPERIMENTS

We conduct a set of comprehensive experiments to demonstrate the quality and efficiency of our proposed MRI algorithms. First, we show that our randomized hill exploration algorithms are scalable and achieve much better response time than the exact algorithms while maintaining similar result quality (Section 5.1). Second, through a set of Amazon Mechanical Turk studies, we demonstrate that interpretations generated by our approaches are superior to the simple aggregate ratings returned by current systems (Section 5.2).

Data Set: We use the MovieLens [4] 100K ratings dataset for our evaluation purposes because the two alternative MovieLens datasets with more ratings (1M and 10M ratings datasets) do not contain user details that are required for our study. The dataset has 100,000 ratings for 1682 movies by 943 users. Four user attributes (gender, age, occupation, location) are used for cuboid description, with the number of distinct values ranging from 2 (gender) to 52 (location).

The number of ratings for each movie can vary significantly, which can have a significant impact on the performance of the algorithms. As a result, we organize the movies into 6 bins of equal sizes, in the order of increasing number of ratings. In particular, *Bin 1* contains movies with the

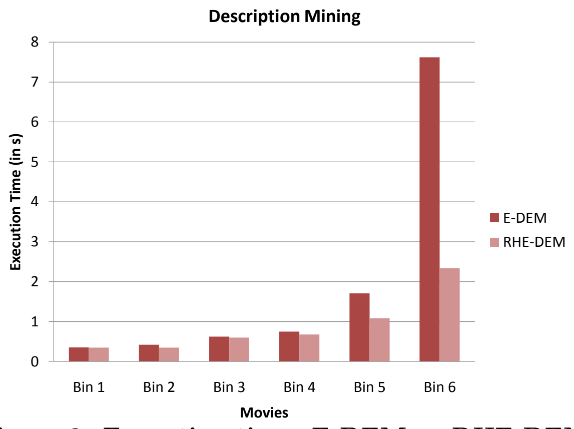


Figure 3: Execution time: E-DEM vs RHE-DEM.

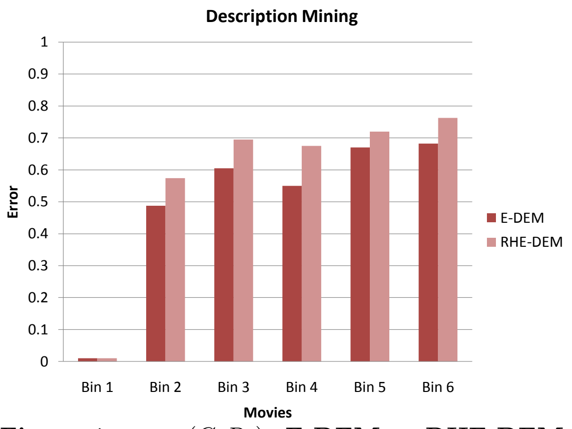


Figure 4: $\mathtt{error}(C, R_I)$: E-DEM vs RHE-DEM.

fewest number of ratings (on average 2) and *Bin 6* contains movies with the highest number of ratings (on average 212). Additional details about the dataset are in Appendix A.3.1.

System configuration: Our prototype system is implemented in Java with JDK 5.0. All experiments were conducted on an Windows XP machine with 3.0Ghz Intel Xeon processor and 2GB RAM. The JVM size is set to 512MB. All numbers are obtained as the average over three runs.

5.1 Performance Evaluation

We compare the execution time for computing interpretations using the exact and the randomized hill exploration algorithms. For all our experiments, we fix the number of groups to be returned at $k = 2$, since the brute-force algorithms are not scalable for larger k.

Figure 3 and 4 compares the average execution time and average description error respectively of E-DEM and RHE-DEM. As expected, while the execution time difference is small for movies with small number of ratings, RHE-DEM computes the descriptions much faster than E-DEM for movies with a large number of ratings (i.e., Bins 5, 6). Moreover, it reduces the execution time from over 7 seconds to about 2 seconds on average for movies in *Bin 6*, which is significant because we can effectively adopt description mining in an interactive real-time setting with our RHE algorithm. Despite signicant reduction in the execution time, our heuristic algorithm does not compromise too much in terms of quality. In fact, as Figure 4 illustrates, the average description error is only slightly larger for RHE.

Similar results are found with the comparison of E-DIM and RHE-DIM algorithms, described in details in Appendix A.3.2. Furthermore, both RHE-DEM and RHE-DIM algorithms scale well with the number of cuboids (i.e., k), the details of which is shown in Appendix A.3.3.

5.2 User Study

We now evaluate the benefits of rating interpretations in an extensive user study conducted through Amazon Mechanical Turk (AMT)[5]. In particular, we aim to analyze whether users prefer our sophisticated rating interpretations against the simple rating aggregation currently adopted by all online rating sites. We conduct two sets of user studies, one for description mining and one for difference mining. Each set involves 4 randomly chosen movies and 30 independent single-user tasks. For each movie in the task, we ask the user to select the most preferred rating interpretations among the three alternatives: simple aggregate ratings, rating interpretation by E-DEM (or E-DIM) algorithms, and those by RHE-DEM (or RHE-DIM) algorithms. The details of the user study setup can be found in Appendix A.3.4.

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

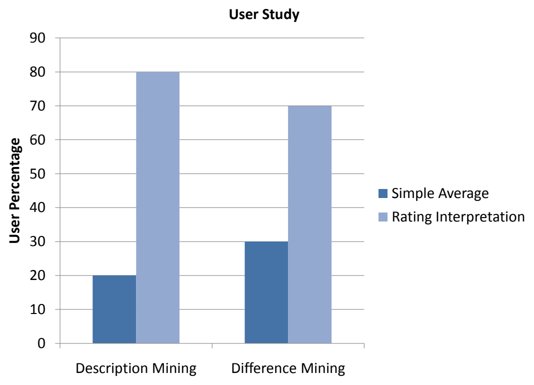


Figure 5: Users Prefer Rating Interpretations.

Figure 5 compares the simple overall average rating approach against our approach returning movie rating interpretations to the user. The simple average represents the percentage of users choosing average ratings, whereas the latter is computed as an addition of percentage of users preferring rating interpretations produced by either exact or RHE algorithms. From the results, it is clear that users overwhelmingly prefer the more informative explanations to the overall average rating, thus confirming our motivation. We also observe that when an user is unfamiliar with a movie in the study, she is particularly inclined to meaningful rating explanations over average rating.

To verify that the quality of results produced by our RHE algorithms are on par with the exact algorithms, we leverage the same user study facility to compare the interpretations produced by both. As shown in Figure 6, from the user's perspective and for both description mining and difference mining, results produced by exact and RHE algorithms are statistically similar. In fact, the users even slightly prefer results from the heuristic algorithm for the difference mining. This validates that our heuristic algorithms are viable, cheaper, alternatives to the brute-force algorithms.

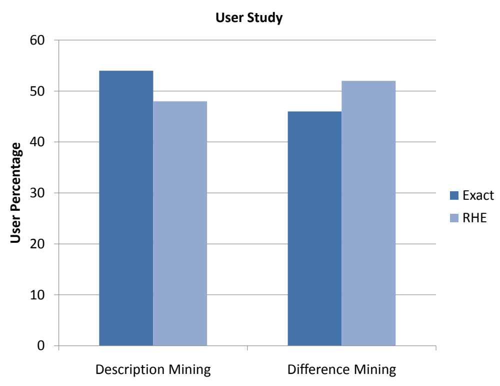


Figure 6: Exact and RHE Algorithms Produce Similar Results.

6. RELATED WORK

Data Cubes: Our idea of using structurally meaningful cuboids as the basis for rating interpretation is inspired by studies in data cube mining, first proposed in Gray et. al [6] and Ramakrishnan et. al [10]. Among those studies, Quotient Cube [9], KDAP [13], Intelligent Roll-ups [11] and Promotion Analysis [14] investigate the problem of ranking and summarizing cuboids, which is similar to our goal here. However, none of these adopts formal objective measures based on user ratings like in our work. To the best of our knowledge, our work is the first to leverage structurally meaningful descriptions for collaborative rating analysis.

Dimensionality Reduction: Several dimensionality reduction techniques, such as Subspace Clustering and PCA, were developed in order to describe a large structured dataset as labeled clusters. While Subspace Clustering [2] may be extended to handle our description mining task, it needs to be modified for scalability. Adapting subspace clustering to difference mining is a non-obvious algorithmic task though. On the other hand, PCA relies on pre-determining the set of attributes to use to describe clusters instead of discovering them on the fly, as in our work.

Recommendation Explanation: Due to the popular adoption of recommendation systems by online sites such as Amazon and Netflix, explaining recommendations has also received significant attention. Herlocker et. al [7] provides a systematic study of explanations for recommendation systems. Yu et. al [15] describes how explanations can be leveraged for recommendation diversification. Bilgic and Mooney [3] convincingly argues that the goal of a good explanation is not necessarily promotion, but to enable users to make well-informed decisions. Our study of rating interpretation is one step toward this ultimate goal of providing users with useful explanations to make informed decisions.

7. CONCLUSION

In this paper, we have introduced the novel problem of *meaningful rating interpretation* (MRI) in the context of collaborative rating sites that exploits the rich structure in metadata describing users to discover meaningful reviewer sub-populations to be presented to the user. Unlike unsupervised clustering approaches, groups returned by MRI are meaningful due to the common structural attribute value pairs shared by all reviewers in each group. Our experiments validate the need for rating interpretation, and demonstrate that our proposed heuristic algorithms generate equally good groups as exact brute-force algorithms with much less execution time. We intend to investigate alternate heuristic search techniques with smarter starting points, besides conducting experiments on larger datasets.

Finally, our work is a preliminary look at a very novel area of research and there appear to be many exciting directions of future research. For example, the problem can be extended to provide meaningful interpretations of ratings by reviewers of interest. Furthermore, additional constraints can be introduced such as diversity of rating explanations.

8. REFERENCES

[1] S. Acharya, P. B. Gibbons and V. Poosala. Congressional samples for approximate answering of group-by queries. In *SIGMOD*, pages 487–498, 2000.

[2] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD*, pages 94–105, 1998.

[3] M. Bilgic and R. J. Mooney. Explaining recommendations: Satisfaction vs. promotion. In *Beyond Personalization (workshop at IUI)*, pages 13–18, 2005.

[4] Y. Chen, F. M. Harper, J. A. Konstan and S. X. Li. Social comparisons and contributions to online communities: A field experiment on movielens. In *Computational Social Systems and the Internet*, 2007.

[5] N. M. David, D. W. Cheung and W. Lian. Similarity search in sets and categorical data using the signature tree. In *ICDE*, pages 75–86, 2003.

[6] J. Gray, A. Bosworth, A. Layman, D. Reichart and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. In *ICDE*, pages 152–159, 1996.

[7] J. L. Herlocker, J. A. Konstan and J. Riedl. Explaining collaborative filtering recommendations. In *CSCW*, pages 241–250, 2000.

[8] L. Hyafil and R. L. Rives. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5:15–17, 1976.

[9] L. V. S. Lakshmanan, J. Pei and J. Han. Quotient cube: how to summarize the semantics of a data cube. In *VLDB*, pages 778–789, 2002.

[10] R. Ramakrishnan and B.-C. Chen. Exploratory mining in cube space. *Data Mining and Knowledge Discovery*, 15(1):29–54, 2007.

[11] G. Sathe and S. Sarawagi. Intelligent rollups in multidimensional olap data. In *VLDB*, pages 531–540, 2001.

[12] D. E. Vaughan, S. H. Jacobson and H. Kaul. Analyzing the performance of simultaneous generalized hill climbing algorithms. *Computational Optimization and Applications*, 37:103–119, 2007.

[13] P. Wu, Y. Sismanis and B. Reinwald. Towards keyword-driven analytical processing. In *SIGMOD*, pages 617–628, 2007.

[14] T. Wu, D. Xin, Q. Mei and J. Han. Promotion analysis in multi-dimensional space. *PVLDB*, 2(1):109–120, 2009.

[15] C. Yu, L. V. S. Lakshmanan and S. Amer-Yahia. It takes variety to make a world: diversification in recommender systems. In *EDBT*, pages 368–378, 2009.

APPENDIX

A. APPENDIX

A.1 Proofs

In this section, we provide detailed proofs of various theorems in the main paper.

A.1.1 Proof of Theorem 1

THEOREM 1. *The decision version of the problem of meaningful description mining is NP-Complete even for boolean databases, where each attribute ia_j in $\mathcal{I}_A$ and each attribute ua_j in $\mathcal{U}_A$ takes a boolean value of either 0 or 1.*

Proof: The decision version of the problem of meaningful description mining (DEM) is as follows: For a given set of items and their ratings R_I, is there a set of cuboids C, such that $\texttt{error}(C, R_I) \leq \beta$, subject to $|C| \leq k$ and $\texttt{coverage}(C, R_I) \geq \alpha$. The membership of the decision version of the description mining problem in NP is obvious.

To verify NP-completeness, we reduce the Exact 3-Set Cover problem (EC3) to the decision version of our problem. EC3 is the problem of finding an exact cover for a finite set U, where each of the subsets available for use contain exactly 3 elements. The EC3 problem is proved to be NP-Complete by a reduction from the Three Dimensional Matching problem in computational complexity theory [8].

Let an instance of EC3 (U, S) consist of a finite set $U = \{x_1, x_2, \ldots x_n\}$ and a family $S = \{S_1, S_2, \ldots S_m\}$ of subsets of U, such that $|S_i| = 3$, $\forall i$ $1 \leq i \leq n$. We are required to construct an instance of DEM (R_I, k, α, β) having $k = (\frac{n}{3}+1)$, $\alpha = 100$ (so that, $\texttt{coverage}(C, R_I) = 100\%$) and $\beta = 0$ (so that, $\texttt{error}(C, R_I) = 0$); such that there exists a cover $C \subseteq S$ of $\frac{n}{3}$ pairwise disjoint sets, covering all elements in U, *if and only if*, a solution to our instance of DEM exists.

We define $(m+1)$ Boolean attributes $A = \{A_1, A_2, \ldots A_{m+1}\}$ and $(n+1)$ tuples $T = \{t_1, t_2, \ldots t_{n+1}\}$, where each entry t_i has a corresponding Boolean rating. For each $S_i = \{x_i, x_j, x_k\}$, A_i has Boolean 1 for tuples $\{t_i, t_j, t_k\}$; while the remaining tuples are set to Boolean 0. For attribute A_{m+1}, tuples $\{t_1, t_2, \ldots t_{n+1}\}$ are all set to 0. The ratings corresponding to tuples $\{t_1, t_2, \ldots t_n\}$ are all 0, while tuple $\{t_{n+1}\}$ has a rating of 1. Figure 7 illustrates example instances of the EC3 problem and our DEM problem.

As defined in Section 2, cuboids (or, groups) are selection query conditions retrieving structurally meaningful groupings of the ratings. For Boolean attributes $\{A_1, A_2, \ldots A_{m+1}\}$, a query condition $Q \in \{0, 1, *\}^{m+1}$, where attribute A_i in Q is set to 0, 1 or *, $\forall i$ $1 \leq i \leq (m+1)$. The space of all possible cuboids (or, query conditions) is 3^{m+1}.

Now, the DEM instance has a solution if $\texttt{error}(C, R_I) = 0$ and $\texttt{coverage}(C, R_I) = 100\%$. Note that, each cuboid in the set C of k cuboids in the solution for DEM should choose tuples either from $T_1 = \{t_1, t_2, \ldots t_n\}$ or from $T_2 = \{t_{n+1}\}$ to achieve $\texttt{error}(C, R_I) = 0$. We need one cuboid 0^{m+1} to cover the single tuple in T_2. Now, let us focus on how to cover tuples in T_1 with $\frac{n}{3}$ more cuboids.

Lemma 1: A selection query $Q \in \{0, *\}^m\{0, 1, *\}$ cannot retrieve non-empty set of tuples only from T_1.

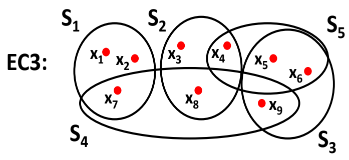
For a query $Q \in \{0, *\}^m\{0, 1, *\}$: if $Q \in \{0, *\}^m\{1\}$, no tuple is selected; if $Q \in \{0, *\}^m\{0, *\}$, non-empty set of tuples from both T_1 and T_2 are selected. Thus queries of the form $\{0, *\}^m\{0, 1, *\}$ cannot yield a solution for the DEM instance.

Lemma 2: A query $Q \notin \{0, *\}^{i-1}\{1\}\{0, *\}^{m-i}0, 1, *$, $\forall i$ $1 \leq i \leq m$ cannot have a solution for the DEM instance.

If a cuboid (or selection query) has 2 or more attributes A_i set to 1, the set of covered tuples is strictly smaller than 3. Thus cuboids that select exactly 3 elements have to have exactly one attribute A_i is set to 1, $\forall i$ $1 \leq i \leq m$,

From Lemmas 1 and 2 we conclude that a set C of $(\frac{n}{3}$ pairwise disjoint cuboids in which one cuboid covers exactly one tuple (defined by query $\{0\}^{m+1}$), and the remaining $\frac{n}{3}$ cuboids each cover exactly 3 tuples (each defined by a query of the form $\{0, *\}^{i-1}\{1\}\{0, *\}^{m-i}\{0, 1, *\}$, and satisfying $\texttt{error}(C, R_I) = 0$ and $\texttt{coverage}(C, R_I) = 100\%$ corresponds to the solution to EC3. The meaningful description mining problem is NP-Complete for Boolean databases. □

EC3:



$U = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9\}$

$S = \{S_1, S_2, S_3, S_4, S_5\}$

- $S_1 = \{x_1, x_2, x_7\}$
- $S_2 = \{x_3, x_4, x_8\}$
- $S_3 = \{x_5, x_6, x_9\}$
- $S_4 = \{x_7, x_8, x_9\}$
- $S_5 = \{x_4, x_5, x_6\}$

DEM:

R_I	A_1	A_2	A_3	A_4	A_5	A_6	Rating
t_1	1	0	0	0	0	0	0
t_2	1	0	0	0	0	0	0
t_3	0	1	0	0	0	0	0
t_4	0	1	0	0	1	0	0
t_5	0	0	1	0	1	0	0
t_6	0	0	1	0	1	0	0
t_7	1	0	0	1	0	0	0
t_8	0	1	0	1	0	0	0
t_9	0	0	1	1	0	0	0
t_{10}	0	0	0	0	0	0	1

Figure 7: Example instances of EC3 and DEM.

A.1.2 Proof of Theorem 2

THEOREM 2. *The decision version of the problem of meaningful difference mining is NP-Complete even for boolean databases.*

Proof: The decision version of the problem of meaningful difference mining (DIM) is as follows: For a given set of items and their ratings R_I, is there a set of cuboids C, such that $\texttt{balance}(C, R_I^+, R_I^-) \leq \beta$, subject to $|C| \leq k$ and $\texttt{coverage}(C, R_I^+) \geq \alpha \wedge \texttt{coverage}(C, R_I^-) \geq \alpha$. The membership of the decision version of the difference mining problem in NP is obvious. To verify its NP-completeness, we again reduce the Exact 3-Set Cover problem (EC3) to the decision version of DIM.

Similar to the proof in Theorem 1, we consider an instance of EC3 (U, S); we are required to construct an instance of DIM (R_I, k, α, β) having $k = (\frac{n}{3}+1)$, $\alpha = 100$ (so that, $\texttt{coverage}(C, R_I^+) = 100 \wedge \texttt{coverage}(C, R_I^-) = 100\%$) and $\beta = 0$ (so that, $\texttt{balance}(C, R_I^+, R_I^-) = 0$); such that there exists a cover $C \subseteq S$ of size $\frac{n}{3}$, covering all elements in

U, *if and only if*, a solution to our instance of DIM exists. The reduction follows the same steps as that in Theorem 1, except that the ratings corresponding to tuples $\{t_1, t_2, \ldots t_n\}$ are all 0 (indicating negative rating), while tuple $\{t_{n+1}\}$ has a rating of 1 (indicating positive rating). $\square$

A.1.3 Proof of Theorem 3

THEOREM 3. *Given R_I and C,* $\texttt{balance}(C, R^+_{I\,i}, R^-_{I\,i})$ *computed using Equation 1 is equivalent to the one computed using the formula in Section 3.2.*

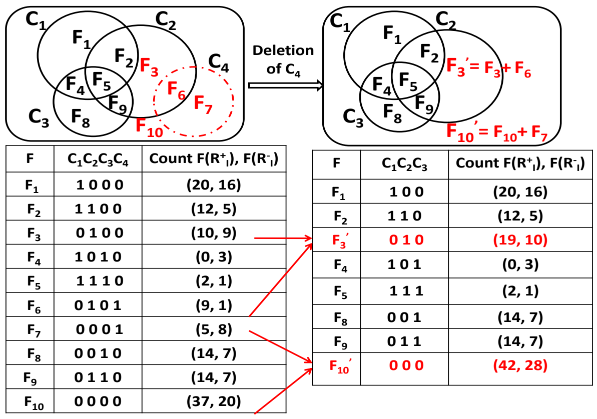
Proof: The standard computation of aggregate balance $\texttt{balance}(C, R^+_I, R^-_I)$ looks up all possible pairings of positive and negative ratings, for each set of k cuboids. The pseudo code of the standard technique is presented in Algorithm 3. It scans each of the k cuboids in C to identify possible positive and negative rating pairings. The method maintains a $|R^+_I| \times |R^-_I|$ matrix for book-keeping, all of whose elements are first initialized to zero and then set to one, whenever a particular element position (corresponding to postive-negative rating pairing) is encountered. The total number of one-s in the $|R^+_I| \times |R^-_I|$ matrix determines the measure $\texttt{balance}(C, R^+_I, R^-_I)$. $\square$

A.2 Incremental Balance Computation

The efficiency of our proposed algorithms can be improved by employing indexing on the data tables [5]. We can build index structure on our tables, so that selection queries are capable of retrieving rating tuples (and thereby compute description error or difference balance) without having to scan the entire database. The idea of partitioning ratings into fundamental regions, introduced in Section 3.2 can be further made use of in designing incremental techniques to speed up the execution time of our difference mining algorithm.

In our exact and RHE algorithms, investigation of a set C of k cuboids is followed by investigation of another set $C' = C - \{c_i\} + \{c_j\}$, where c_j is a neighbor of c_i. Since cuboids c_i and c_j are neighbors, they share a set of attribute value pairs. If c_j is a child of c_i in the connected lattice, C to C' brings about a reduction in the rating space; if c_j is a parent of c_i, C to C' results in an expansion in the set of ratings. In other words, the tuples that needs to be updated in the positive and negative rating tuple counts for the fundamental regions are those satisfied by the query $\{c_i\}$-$\{c_j\}$ (or, $\{c_j\}$-$\{c_i\}$). Therefore, one seemingly straightforward incremental technique will identify the tuples in $\{c_i\}$-$\{c_j\}$ (or, $\{c_j\}$-$\{c_i\}$) and update the positive and negative rating tuple aggregates of the fundamental regions getting affected due to c_j. However in such a framework, it will not be possible to identify the fundamental regions whose aggregates need to be updated and the method may eventually end up looking up all regions and rating tuples. Hence, we interpret the move from set C to C' as *deletion* of c_i followed by *insertion* of c_j.

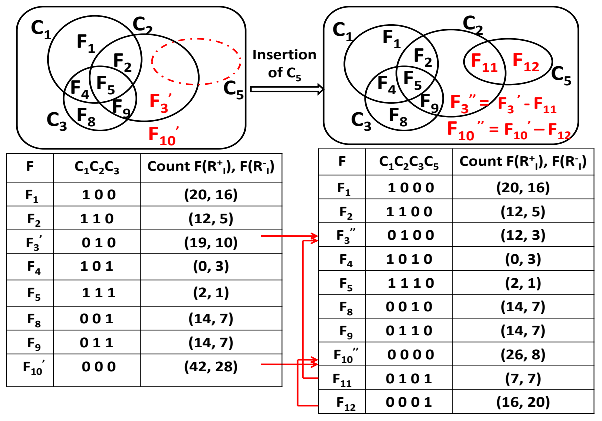
Deletion of c_i: Figure 8 explains our deletion operation through an example, where $C = \{c_1, c_2, c_3, c_4\}$ of $k = 4$ cuboids partitions 200 ratings in R_I into 10 fundamental regions F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_8, F_9 and F_{10}, each having a distinct 4-bit signature and a pair of positive and negative rating tuple aggregates $(F_i(R^+_I), F_i(R^-_I))$. Assume, we want to delete cuboid c_4 from set C of 4 cuboids (marked



F	$C_1C_2C_3C_4$	Count $F(R^+_I)$, $F(R^-_I)$
F_1	1 0 0 0	(20, 16)
F_2	1 1 0 0	(12, 5)
F_3	0 1 0 0	(10, 9)
F_4	1 0 1 0	(0, 3)
F_5	1 1 1 0	(2, 1)
F_6	0 1 0 1	(9, 1)
F_7	0 0 0 1	(5, 8)
F_8	0 0 1 0	(14, 7)
F_9	0 1 1 0	(14, 7)
F_{10}	0 0 0 0	(37, 20)

F	$C_1C_2C_3$	Count $F(R^+_I)$, $F(R^-_I)$
F_1	1 0 0	(20, 16)
F_2	1 1 0	(12, 5)
F_3'	0 1 0	(19, 10)
F_4	1 0 1	(0, 3)
F_5	1 1 1	(2, 1)
F_8	0 0 1	(14, 7)
F_9	0 1 1	(14, 7)
F_{10}'	0 0 0	(42, 28)

Figure 8: Deletion of cuboid C_4.



F	$C_1C_2C_3$	Count $F(R^+_I)$, $F(R^-_I)$
F_1	1 0 0	(20, 16)
F_2	1 1 0	(12, 5)
F_3'	0 1 0	(19, 10)
F_4	1 0 1	(0, 3)
F_5	1 1 1	(2, 1)
F_8	0 0 1	(14, 7)
F_9	0 1 1	(14, 7)
F_{10}'	0 0 0	(42, 28)

F	$C_1C_2C_3C_5$	Count $F(R^+_I)$, $F(R^-_I)$
F_1	1 0 0 0	(20, 16)
F_2	1 1 0 0	(12, 5)
F_3''	0 1 0 0	(12, 3)
F_4	1 0 1 0	(0, 3)
F_5	1 1 1 0	(2, 1)
F_8	0 0 1 0	(14, 7)
F_9	0 1 1 0	(14, 7)
F_{10}''	0 0 0 0	(26, 8)
F_{11}	0 1 0 1	(7, 7)
F_{12}	0 0 0 1	(16, 20)

Figure 9: Insertion of cuboid C_5.

in dotted circle in Figure 8). First, we identify the fundamental regions, whose aggregates will be affected ($F_i(R^+_I)$ and $F_i(R^-_I)$ will increase or decrease) due to the deletion of c_4. We compare the query c_4 with the 3 other queries c_1, c_2 and c_3. If an attribute-value pair in c_4 is identical to an attribute-value pair in at least one of c_1, c_2 and c_3, say c_2 in our example Figure 8, we determine the corresponding fundamental regions whose signature has 2nd bit set to 1 namely F_2, F_3, F_5, F_6, F_9, and the fundamental region whose all 4 bits are set to 0 namely F_{10} respectively. In this way, we reduce the set of fundamental regions that may be affected from 10 to 6. We further reduce this set of 6 fundamental regions to identify exactly those that will be affected due to deletion of c_4. The fundamental regions from F_2, F_3, F_5, F_6, F_9, F_{10} whose signature has 4th bit set to 1, namely F_6 and F_7 are marked - they will be deleted from the set of fundamental regions; the fundamental regions whose signatures are identical with that of F_6 and F_7 except the 4th bit, namely F_3 and F_{10} are also marked - the aggregates $(F_3(R^+_I), F_3(R^-_I))$ and $(F_{10}(R^+_I), F_{10}(R^-_I))$ will be updated. Therefore out of 10 fundamental regions, only 4 needs to be updated while the remaining 6 regions continue to be same. Let us denote the updated regions as F'_3 and F'_{10}. $F'_3(R^+_I) = F_3(R^+_I) + F_6(R^+_I)$; $F'_3(R^-_I) = F_3(R^-_I) + F_6(R^-_I)$ (marked in Figure 8) and $F'_{10}(R^+_I) = F_{10}(R^+_I) + F_7(R^+_I)$; $F'_{10}(R^-_I) = F_{10}(R^-_I) + F_7(R^-_I)$ (marked in Figure 8). Finally, we update the signatures of all the fundamental regions to

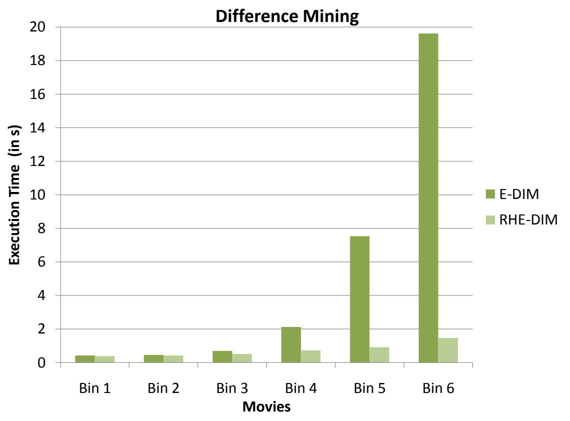


Figure 10: Execution time: E-DIM vs RHE-DIM.

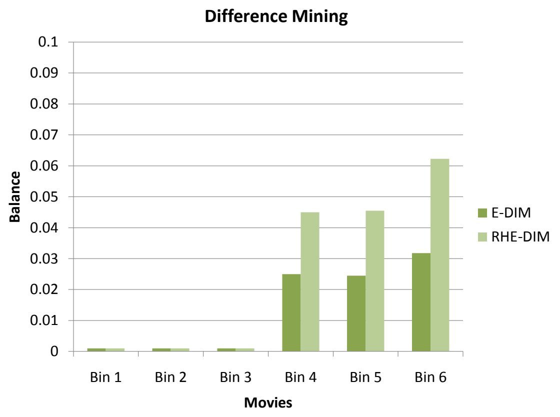


Figure 11: `balance`(C, R_I^+, R_I^-): E-DIM vs RHE-DIM.

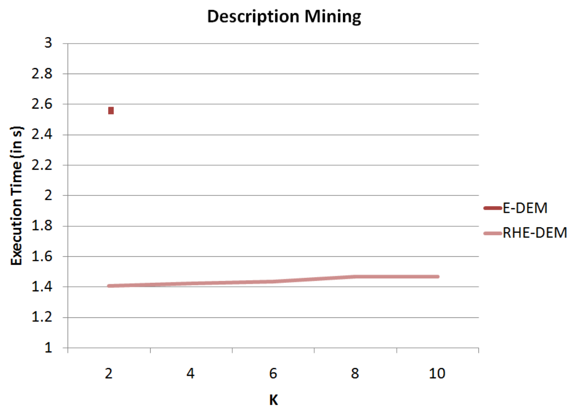


Figure 12: Execution time with increasing k.

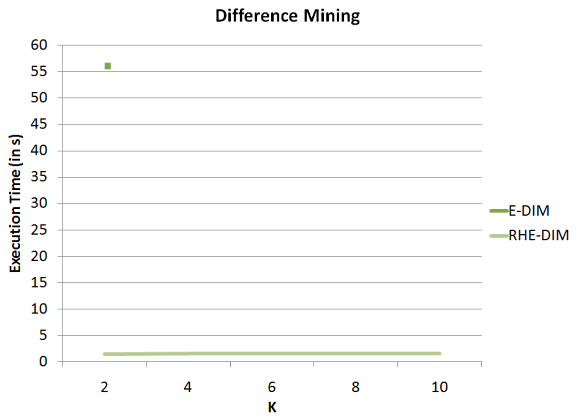


Figure 13: Execution time with increasing k.

a $(4-1) = 3$ bit vector by removing the 4^{th} bit. Note that deletion operation does not require us to visit the database at any point of time. We get $C'' = C$ - $\{c_4\} = \{c_1, c_2, c_3\}$ of $k = 3$ cuboids that partitions 200 ratings in R_I into 8 fundamental regions, F_1, F_2, F_3', F_4, F_5, F_8, F_9 and F_{10}'.

Insertion of c_j: Figure 9 explains our insertion operation that follows the deletion operation in Figure 8 to build $C' = C'' + \{c_5\} = C$ - $\{c_4\} + \{c_5\}$. Considering the same example, $C'' = \{c_1, c_2, c_3\}$ of $k = 3$ cuboids partitions 200 ratings in R_I into 8 fundamental regions F_1, F_2, F_3', F_4, F_5, F_8, F_9 and F_{10}', each having a distinct 3-bit signature and a pair of positive and negative rating tuple aggregates $(F_i(R_I^+), F_i(R_I^-))$. Assume, we want to insert cuboid c_5 into set C'' of 3 cuboids (marked in dotted circle in Figure 9). First, we identify the fundamental regions, whose aggregates will be affected ($F_i(R_I^+)$ and $F_i(R_I^-)$ will increase or decrease) due to the insertion of c_5. We compare the new query c_5 with the 3 existing queries c_1, c_2 and c_3. If an attribute-value pair in c_5 is identical to an attribute-value pair in at least one of c_1, c_2 and c_3, say c_2 in Figure 9, we determine the corresponding fundamental regions whose signature has 2^{nd} bit set to 1 namely F_2, F_3', F_5, F_9, and the fundamental region whose all 3 bits are set to 0 namely F_{10}' respectively. In this way, we reduce the set of fundamental regions that may be affected from 10 to 5. We further reduce this set of 5 fundamental regions to identify exactly those that will be affected due to insertion of c_5. The fundamental regions from F_2, F_3', F_5, F_9, F_{10}' whose signature has no other bit (other than the 2^{nd} bit) set to 1, namely F_3' and F_{10}' are marked - they will be partitioned into F_3'', F_{11} and F_{10}'', F_{12} respectively. Therefore out of 8 fundamental regions, only 2 needs to be partitioned, while the remaining 6 regions continue to be same. We update the signatures of all fundamental regions by setting the new bit (4^{th} bit now corresponds to c_5) to 0 in all except the two new regions F_{11} and F_{12}. Note, the first 3 bits in F_3'', F_{11} and F_{10}'', F_{12} are identical to that in F_3' and F_{10}' respectively. Next, we select the set of 50 tuples T' for the query condition in c_5 and assign a 4-bit vector signature to each tuple in T' based on its coverage by cuboids c_1, c_2, c_3 and c_5. The index structures on the data tables support a fast retrieval of T'. Once the signatures are built and the fundamental regions corresponding to T' are determined, we match the signatures from T' with that of fundamental regions F_3' and F_{10}', except the 4^{th} bit. If there exists a match with F_3' or F_{10}', we increment the counts of new regions $(F_{11}(R_I^+), F_{11}(R_I^-))$ or $(F_{12}(R_I^+), F_{12}(R_I^-))$ as well as update the counts of regions $(F_3''(R_I^+), F_3''(R_I^-))$ or $(F_{10}''(R_I^+), F_{10}''(R_I^-))$ respectively. $F_3''(R_I^+) = F_3'(R_I^+) - F_{11}(R_I^+)$; $F_3''(R_I^-) = F_3'(R_I^-) - F_{11}(R_I^-)$ (marked in Figure 9) and $F_{10}''(R_I^+) = F_{10}'(R_I^+) - F_{12}(R_I^+)$; $F_{10}''(R_I^-) = F_{10}'(R_I^-) - F_{12}(R_I^-)$ (marked in Figure 9). We get $C' = C'' + c_5 = \{c_1, c_2, c_3, c_5\}$ of $k = 4$ cuboids partitions 200 ratings in R_I into 10 fundamental regions, F_1, F_2, F_3'', F_4, F_5, F_8, F_9, F_{10}'', F_{11} and F_{12}.

A.3 Additional Experimental Evaluation

In this section, we provide additional details on our experimental evaluation that we are not able to describe in the main paper due to space limitation.

A.3.1 Additional Data Set Details

User attributes: There are four user attributes that we consider in the MovieLens dataset that we adopt, including gender, age, occupation and zipcode. The attribute *gender* takes two distinct values: male or female. We convert the numeric *age* into four categorical attribute values, namely teen-aged (under 18), young (18 to 35), middle-aged (35 to 55) and old (over 55). There are 21 different *occupations* listed by MovieLens, such as student, artist, doctor, lawyer, etc. Finally, we convert zipcodes to states in the USA (or foreign, if not in USA) by using the USPS zip code lookup (http://zip4.usps.com). This produces the user attribute, *location*, which takes 52 distinct values.

Binning the movies: As described in Section 5, one important factor when conducting the performance evaluation is the number of ratings that we are generating interpretations from. Intuitively, the more ratings we have to consider, the more costly the interpretation process is expected to be. Therefore, we order our set of 1682 movies according to the number of ratings each movie has, and then partition them into 6 bins of equal sizes, where Bin 1 contains movies with the fewest and Bin 6 contains movies with highest number of ratings. Table 1 shows the statistics of those bins. We randomly pick 100 movies from each bin and compare the execution time and the objective score (`error` for description mining and `balance` for difference mining) of both the exact algorithms and our heuristic algorithms.

	lowest #rtg	highest #rtg	avg #rtgs
Bin 1	1	4	2
Bin 2	4	11	7
Bin 3	11	27	18
Bin 4	27	59	41
Bin 5	59	121	84
Bin 6	121	583	212

Table 1: Bin Statistics.

A.3.2 Additional Performance Evaluation: Difference Mining Experiments

In Section 5.1, we compare the average execution time and the average error for description mining using the exact and randomized hill exploration algorithms. Here, Figures 10 and 11 report similar results comparing the average execution time and the average balance score respectively for the difference mining task. Again, we see that our heuristic algorithm performs much faster (reducing the execution time from over 20 second to less than 2 seconds) without compromising much on the overall balance score.

A.3.3 Additional Performance Evaluation: Scalability Experiments

Figures 12 and 13 illustrate the execution time of our RHE algorithms for description mining and difference mining respectively, over increasing number of cuboids in the results. A randomly chosen movie, *Gone With The Wind*, is used in this analysis. The results show that RHE algorithms are very scalable where the execution time remains reasonably small through the range of k values up to 10, which we believe to be the upper limit of how many explanations a user can consume for a single item. Note that the execution time of brute-force algorithms could not be reported beyond $k = 2$ because they failed to finish within a reasonable amount of time. High coverage of item ratings by few general groups such as $\{\langle \texttt{age}, \texttt{young} \rangle, \langle \texttt{occupation}, \texttt{student} \rangle\}$, etc. who frequenty particpate in collaborative rating sites and very low coverage by majority of the groups in the rating lattice such as $\{\langle \texttt{gender}, \texttt{female} \rangle, \langle \texttt{age}, \texttt{old} \rangle, \langle \texttt{occupation}, \texttt{librarian} \rangle\}$, etc. supports the exploration phase to reach a local minima quickly, thus making our RHE algorithms scalable.

A.3.4 User Study Details

Section 5.2 provides a high level overview of our Amazon Mechanical Turk user study. In this section, we dive into the details of how the user studies are conducted. There are two sets of user studies, one for description mining and one for difference mining. Each set involves 4 randomly chosen popular (with over 50 ratings) movies[6] and 30 independent single-user tasks. For description mining, the four movies chosen are *Toy Story*, *Titanic*, *Mission Impossible* and *Forrest Gump*. For difference mining, we bias toward more controversial movies and chose *Crash*, *101 Dalmatians*, *Space Jam* and *Ben Hur*.

Each task is conducted in two phases: *User Knowledge Phase* and *User Judgment Phase*. During the first phase, we estimate the users' seriousness about the task and familiarity about the movies in the task by asking them to complete a survey. The survey contains a few very simple questions about the movies that we use to prune out malicious users who simply try to complete the task by answering questions randomly. We also draw some interesting observations from the user study. In the second phase, for each movie in the task, we present to the user three alternative interpretations of the ratings about the movie for description mining and difference mining:

- Option **(a)** overall average rating (**simple**)
- Option **(b)** the interpretation produced by the exact algorithms (**E-DEM, E-DIM**)
- Option **(c)** the interpretation produced by our randomized hill exploration algorithms (**RHE-DEM, RHE-DIM**), where the number of explanations (i.e., the number of cuboid groups presented) for both exact and heuristic is limited at 3.

The user is then asked to judge which approach she prefers. The responses from all the users are then aggregated to provide an overall comparison between the three approaches.

[6]For movies that are less popular, uses can simply go over all the ratings one by one, therefore rating interpretation does not bring much benefit.