Redescription Mining: Structure Theory and Algorithms

Laxmi Parida
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598, USA
parida@us.ibm.com

Naren Ramakrishnan
Department of Computer Science
Virginia Tech, VA 24061, USA
naren@cs.vt.edu

Abstract

We introduce a new data mining problem—redescription mining—that unifies considerations of conceptual clustering, constructive induction, and logical formula discovery. Redescription mining begins with a collection of sets, views it as a propositional vocabulary, and identifies clusters of data that can be defined in at least two ways using this vocabulary. The primary contributions of this paper are conceptual and theoretical: (i) we formally study the space of redescriptions underlying a dataset and characterize their intrinsic structure, (ii) we identify impossibility as well as strong possibility results about when mining redescriptions is feasible, (iii) we present several scenarios of how we can custom-build redescription mining solutions for various biases, and (iv) we outline how many problems studied in the larger machine learning community are really special cases of redescription mining. By highlighting its broad scope and relevance, we aim to establish the importance of redescription mining and make the case for a thrust in this new line of research.

Introduction

The central goal of this paper is to introduce a new data mining task—redescription mining—that is potentially of interest to a broad AAAI audience working in machine learning, knowledge representation, and scientific discovery. As the name indicates, to redescribe something is to describe anew or to express the same concept in a different vocabulary. The input to redescription mining is a collection of sets such as shown in Fig. 1. Each bubble in this diagram denotes a meaningful grouping of objects (in this case, countries) according to some intensional definition. For instance, the colors green, red, cyan, and yellow (from right, counterclockwise) refer to the sets 'permanent members of the UN security council,' 'countries with a history of communism,' 'countries with land area $> 3,000,000$ square miles,' and 'popular tourist destinations in the Americas (North and South).' We will refer to such sets as *descriptors*. An example redescription for this dataset is then: 'Countries with land area $> 3,000,000$ square miles outside of the Americas' are the same as 'Permanent members of the UN security council who have a history of communism.' This redescription defines the set $\{$Russia, China$\}$, once by a set intersection, and again by a set difference. The goal of redesription mining is to find which subsets afford multiple definitions and to find these definitions. The underlying premise is that sets that can indeed be defined in (at least) two ways are likely to exhibit concerted behavior and are, hence, interesting.

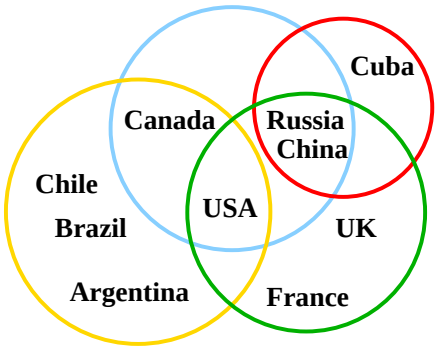

Figure 1: Example input to redescription mining.

As is clear, redescription mining exhibits traits of conceptual clustering, constructive induction, and logical formula discovery. It is a form of conceptual clustering (Fisher 1987; Michalski 1980) because the mined clusters are required to have not just one meaningful description, but two. It is a form of constructive induction since the features important for learning must be automatically constructed from the given vocabulary of sets. And finally, finding redescriptions can be viewed as learning equivalence relationships between boolean formulas.

Why is this problem important?

We posit that today's data-driven sciences, such as bioinformatics, have greatly increased the ease with which descriptors can be defined over a universal set of objects. There is an urgent need to integrate multiple forms of characterizing datasets, situate the knowledge gained from one dataset in the context of others, and harness high-level abstractions for uncovering cryptic and subtle features of data. This is important in order to relate results from different studies, or reconcile the analyses of a common set of biological entities (e.g., stress-responsive genes) from diverse computational techniques and, in this way, expose novel patterns underly-

ing data. Furthermore, redescription mining can be viewed as a generalization of many problems studied in the larger machine learning community:

Profiling Classes: Inducing understandable definitions of classes using a set of features (Valdes-Perez, Pericliev, & Pereira 2000) is a goal common to all descriptive classification applications. Redescription mining loses the distinction between classes and features and provides necessary as well as sufficient descriptors for covering other descriptors. An additional requirement in class profiling is to choose features that absolutely or partially contrast one class from another and, typically, to choose a minimal subset of features that can contrast *all* pairs of classes. Redescriptions impose an equivalence relation over the expression space that can be efficiently harnessed to answer these queries.

Niche Finding: Niche finding is a special case of profiling classes where the classes are singleton sets. Redescriptions hence cover single instances, such as this example from the domain of universities (Valdes-Perez 1999): 'Wake Forest University is the only suburban Baptist university.' Finding niches for individuals and objects has important applications in product placement, targeting recommendations, and personalization.

Analogical Reasoning: Learning (functional) determinations is an important aid in analogical reasoning (Russell 1989) and categorical analysis. A determination captures functional dependency but not the exact form of the dependency, e.g., 'if two instances agree on attribute X, then they also agree on attribute Y.' When the attributes are sets and feature values indicate set membership, a determination can only exist as a redescription, since the domains of all attributes is boolean. Hence redescriptions enjoy the same inductive applications that determinations have, such as allowing us to posit properties for instances based on their membership in certain sets.

Story Telling: While traditional redescription mining is focused on finding object sets that are similar, story telling aims to explicitly relate object sets that are disjoint (and hence, dissimilar). The goal of this application (Ramakrishnan *et al.* 2004) is to find a path or connection between two disjoint sets through a sequence of intermediaries, each of which is an approximate redescription of its neighbor(s). A simple example is the word game where we are given two words, e.g. PURE and WOOL, and we must morph one into the other by changing only one letter at a time (meaningfully). Here we can think of a word as a set of (letter,position) pairs so that all meaningful English words constitute the descriptors. One solution is: PURE $\rightarrow$ PORE $\rightarrow$ POLE $\rightarrow$ POLL $\rightarrow$ POOL $\rightarrow$ WOOL. Each step of this story is an approximate redescription between two sets, having three elements in common. On a more serious note, story telling finds applications in bioinformatics, for instance, where the biologist is trying to relate a set of genes expressed in one experiment to another set, implicated in a different pathway.

	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
o_1	0	0	0	1	1	0	0	1
o_2	1	0	1	0	1	1	0	1
o_3	1	1	0	0	0	1	1	0
o_4	0	1	1	0	0	1	0	0
o_5	0	0	0	1	0	0	1	1

Figure 2: Example dataset matrix D.

Schema Matching: Learning semantic mappings between diverse schema (Doan, Domingos, & Halevy 2003) is important for data reconciliation and information integration applications. If we view the schema as sets, computing one-to-one schema matchings is a problem of redescription mining. We are thus extending the realm of descriptors from propositional to predicate variables. This also allows us to explore redescription in the context of inductive logic programming, e.g., for inferring relational definitions that mutually reinforce each other.

We hence argue that redescription mining is an especially fruitful line of inquiry, understanding of whose issues will shed light on many related areas of data mining. The primary contributions of this paper are conceptual and theoretical —insights into the structure of redescription spaces, results about when mining redescriptions is feasible, and scenarios of how we can custom-build redescription mining solutions for various biases.

Formalisms

Formally, the inputs to redescription mining are the universal set of objects $O = \{o_1, o_2, \ldots, o_n\}$, and a set (the vocabulary) $F = \{F_1, F_2, \ldots, F_m\}$ of proper subsets of O. The elements of F (called *features*) are assumed to form a covering of O ($\bigcup_i F_i = O$), but not necessarily a partition. For notational convenience, this information can be summarized in the $n \times m$ binary *dataset matrix* D (see Fig. 2) whose rows represent objects, columns represents the features, and the entry D_{ij} is 1 if object o_i is a member of feature F_j, and 0 otherwise. The reader will notice the immediate parallels between D and the traditional item-transaction modeling in association rule mining.

Definition 1. *(descriptor e, features $F(e)$, objects $O(e)$) A descriptor is a boolean expression on a set of features $V \subseteq F$. Given a descriptor e, we will denote the set of features involved in e by $F(e)$ and the set of objects it represents (for a presumed D) by $O(e)$.*

For ease of interpretability, notice that we have overloaded notation: F denotes the entire set of features, whereas $F(e)$ denotes the subset of F that participates in e (similarly for O). Also, in writing boolean expressions, we will use boolean connectives ($\wedge$, $\vee$, $\neg$) as well as set constructors ($\cap$, $\cup$, $-$) interchangeably, but never together in the same expression. Example descriptors are F_3, $F_1 \cap F_4$, $\neg F_2 \vee F_3$, and $F_1 - (F_1 - F_4)$.

Two descriptors e_1 and e_2 defined over (resp.) V_1 and V_2 are distinct (denoted as $e_1 \neq e_2$), if one of the following holds: (1) $V_1 \neq V_2$, or (2) there exists some D for which $O(e_1) \neq O(e_2)$. Notice that this condition rules out tautolo-

gies. For example the descriptors $F_1 \cap F_4$ and $F_1 - (F_1 - F_4)$ are not distinct.

Definition 2. *(redescriptions $R(e)$, $O(R(e))$ e' is a* redescription *of e, if and only if $O(e) = O(e')$ holds for the given D. $R(e)$ is the set of all distinct redescriptions of e. $O(R(e))$ is defined to be $O(e)$.*

In the example dataset matrix D of Fig. 2, $(F_3 \cap F_1) \cup (F_4 - F_3)$ is a redescription of $(F_7 - F_6) \cup (F_5 - F_7)$, since they both induce the same set of objects: $\{o_1, o_2, o_5\}$. Furthermore, these expressions are also redescriptions of F_8. We will also adopt the convention that $e \in R(e)$ so that we can partition descriptor space into non-empty and non-overlapping sets. (We still require that all elements of $R(e)$ be distinct from each other.) The following two lemmas follow readily:

Lemma 1. *Given D, if $e_1, e_2 \in R(e)$, $e_1 \neq e_2$, then $(e_1 \wedge e_2), (e_1 \vee e_2) \in R(e)$.*

Lemma 2. *Redescription is reflexive, symmetric, and transitive: it induces a partition on a collection of descriptors on D.*

Clearly, the set of redescriptions of e as defined by $R(e)$ contains redundant elements; we will soon see how to define $R(e)$ succinctly without enumerating all its elements. As a first attempt at arriving at a minimal set of redescriptions for a descriptor e, we can reason whether this set would be pairwise disjoint in its use of features, i.e., whether the following holds.

Conjecture 1. *Fixing a set of features can endow a unique (upto tautology) description of a set of objects.*

We answer in the negative using a counterxample. Given the D of Fig. 2, there are at least two distinct redescriptions $(e_1 \neq e_2)$ such that $F(e_1) = F(e_2) = \{F_3, F_4\}$ and $O(e_1) = O(e_2) = \{o_1, o_2, o_4, o_5\}$: (1) $e_1 = F_3 \vee F_4$, and, (2) $e_2 = F_3 \oplus F_4 = (\neg F_3 \wedge F_4) \vee (F_3 \wedge \neg F_4)$. Redescription relationships are hence heavily data dependent. Conversely, note that if the values of both F_3 and F_4 are flipped for o_3 in D, e_1 and e_2 are no longer redescriptions of each other. While e_2 would continue to denote the set of objects $\{o_1, o_2, o_4, o_5\}$, the definition of e_1 would get expanded.

Definition 3. *(relaxation $X(e)$ of e, $e' \leq e$) Given descriptors e and e', defined on the features V and V' respectively, e' is a relaxation of e, denoted as $(e' \leq e)$, if $e \Rightarrow e'$ is a tautology. The collection of all the relaxations of e is denoted by $X(e)$.*

Unlike redescription, note that relaxation is a dataset-agnostic concept and is tantamount to formulating a version space (Mitchell 1982) over descriptors. For example, descriptor F_1 is a relaxation of $F_1 \wedge F_2$, $F_1 \vee F_2$ is a relaxation of $F_1 \vee (F_2 \wedge F_3)$, and $F_1 \vee F_2$ is also a relaxation of F_1. It is easy to see the following:

Lemma 3. *Relaxation is reflexive, anti-symmetric, and transitive: it induces a partial order on a collection of descriptors on D.*

Lemma 4. *For each $e_2 \in X(e_1)$, $O(e_2) \supseteq O(e_1)$.*

Given a dataset D, note that a relaxation of e is not necessarily a redescription of e. Consider our running example: $F_1 \in X(F_1 \wedge F_2)$ but $F_1 \notin R(F_1 \wedge F_2)$ since $O(F_1) = \{o_2, o_3\} \supset \{o_3\} = O(F_1 \wedge F_2)$. On the other hand, $F_4 \in X(F_4 \wedge F_8)$ and $O(F_4) = O(F_4 \wedge F_8) = \{o_1, o_5\}$, hence $F_4 \in R(F_4 \wedge F_8)$.

Irredundant Representation of R(e)

We next address the question of describing $R(e)$ in the most concise manner, without any loss of information.

Definition 4. *($Frontier(R(e))$) $Fr(R(e)) \subseteq R(e)$ is defined as follows: (1) for each $e'' \in R(e)$, there is an $e' \in Fr(R(e))$ such that $e'' \in X(e')$ and (2) there is no $e'' \in R(e)$ such that some $e' \in Fr(R(e))$ is a relaxation of e''.*

Theorem 1. *$Fr(R(e))$ is a singleton set.*

Proof. Since $R(e)$ is non-empty, it is clear that $Fr(R(e))$ must contain at least one element. We hence concentrate on proving that $Fr(R(e))$ contains at most one element. Assume this is not true and that there exist distinct $p > 1$ expressions $e_1, e_2, \ldots, e_p \in Fr(R(e))$. Then by Lemma 1, $e_1 \wedge e_2 \wedge \ldots \wedge e_p \in R(e)$, and by the first part of Definition 4, $e_1 \wedge e_2 \wedge \ldots \wedge e_p \in Fr(R(e))$. Then, by the second part of Definition 4, $e_1, e_2, \ldots, e_p \notin Fr(R(e))$. Hence the assumption is wrong and $p = 1$. □

Definition 5. *(relaxation distance p, $X_p(e)$) e' is a relaxation of e at distance p denoted as $(e' \leq_p e)$, if there exists distinct expressions $(e' = e_0), e_1, e_2, \ldots, (e_p = e)$ such that $e_0 \leq e_1 \leq e_2 \ldots \leq e_p$. $X_p(e)$ is the collection of all relaxations of e at distance p.*

We will assume that a descriptor is at a relaxation distance of zero from itself. Note that if $e' \in X_p(e)$ so that $O(e) \subseteq O(e')$, then $O(e') - O(e) \leq p$, for any given D. We give the following lemma that shows the redescription terrain to be 'continuous' in the space of relaxations of a given descriptor (see Fig. 3).

Lemma 5. *Denote $Fr(R(e))$ by e_f. Then if $(e' \in R(e)) \in X_p(e_f)$ for some $p > 0$, then there must exist some $(e'' \in X_q(e_f)) \in R(e)$ for some $0 \leq q < p$ such that $e' \in X(e'')$.*

Proof. Assume the contrary, i.e., there exists a e'' at a relaxation distance q from e_f and for which $e' \in X(e'')$ but $e'' \notin R(e)$. Since $e' \leq e'' \leq e_f$, $O(e_f) \subseteq O(e'') \subseteq O(e')$. Since $e', e_f \in R(e)$, $O(e_f) = O(e')$. Then $O(e'') = O(e_f)$. Hence $e'' \in R(e)$ must hold. □

We now show how to determine the membership of a non-frontier descriptor e (i.e., which redescription terrain it belongs to) for a given dataset D.

Lemma 6. *Given D, let e_1 be a frontier and $e \in X_p(e_1), R(e_1)$. Then if $e \in X_q(e_2)$ for some frontier $e_2 \neq e_1$, then $q > p$.*

Proof. $O(e) = O(e_1)$ since $e \in R(e_1)$. $O(e) \supseteq O(e_2)$ since $e \in X(e_2)$. Thus $O(e_1) \supseteq O(e_2)$, hence $e_1 \in X(e_2)$. Let $e_1 \in X_r(e_2)$ for some $r > 0$ (note that r cannot be

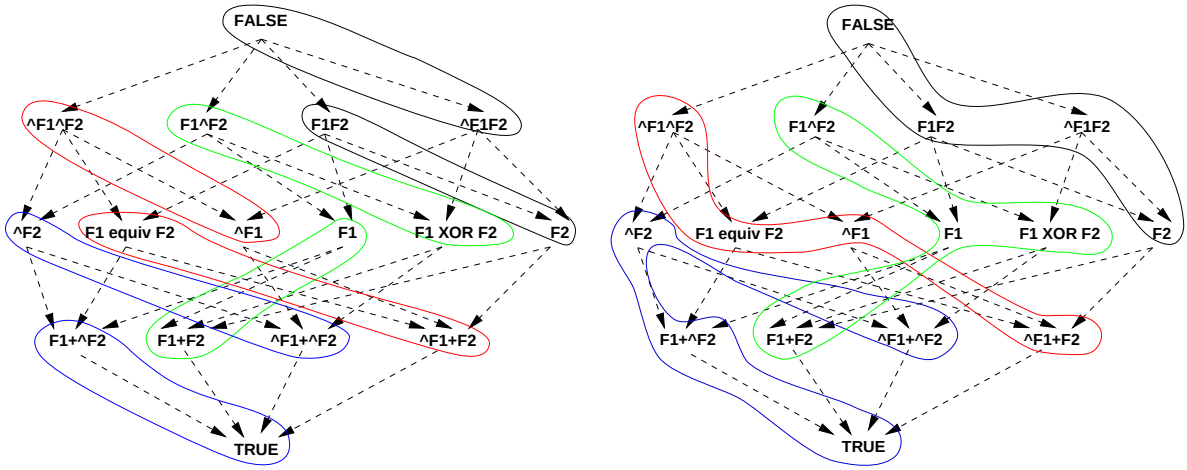

Figure 3: A relaxation lattice over two boolean variables (involving $2^{2^2} = 16$ possible descriptors). Two examples over this lattice are given depicting redescription 'terrains' (in closed curves). The underlying data for the examples comes from: (left) columns F_1, F_2 and rows o_1, o_2, o_3 of Fig. 2. (right) columns F_1, F_2 and rows o_1, o_2 of Fig. 2.

```
Initialize() {
  if e ∈ S
    then E[e].front ← e; E[e].len ← 0
    else E[e].len ← ∞

  (initialize the topmost e for BFS)
  if e = FALSE
    then E[e].depth ← 0
    else E[e].depth ← −1
}

CompRedscrp() {
  for d = 0 . . . D
    foreach e with (E[e].depth = d)
      foreach e′ ∈ X_1(e)
        E[e′].depth ← (d + 1)
        if E[e].len < E[e′].len
         then E[e′].len ← E[e].len + 1;
              E[e′].front ← E[e].front;
}
```

Figure 4: Algorithm for computing redescriptions.

zero since then the premise $e_2 \neq e_1$ will not hold). Then $e \in X_{r+p}(e_2)$. Thus $q = r + p$ and the result follows. □

Lemma 6 essentially states that each non-frontier descriptor e has a unique frontier e_f at the shortest possible relaxation distance p, i.e., $e \in X_p(e_f)$. This suggests a very concise representation for $R(e)$ as $Fr(R(e))$ with the following algorithm to compute all redescriptions given S, the set of all frontiers for the dataset D: traverse the relaxation lattice once using a breadth first search (BFS), while computing the frontier that is the shortest relaxation distance away from each non-frontier descriptor. Let E be an array of three-tuples indexed by the expression e. $E[e].front$ stores the frontier e_f such that $e \in R(e_f)$; $E[e].len$ stores the relaxation distance of e from e_f; and $E[e].depth$ is the depth of e in the relaxation lattice (to aid in the breadth first traversal). E is initialized as shown by *Initialize* in Fig. 4. Thus $E[e_f].len = 0$ for each frontier e_f. Then, *CompRedescrp* proceeds down the lattice beginning from 'FALSE' and assigns every descriptor to its closest equivalence class (here the depth of $(e = TRUE)$ in the relaxation lattice is D). Note that during the execution of this routine, assignment and re-assignment of descriptors to frontiers happens continuously, as long as we find a frontier closer than the current best known frontier. At the termination of the routine, the redescription sets are defined as $R(e_f) = \{e \mid E[e].front = e_f\}$.

Understanding Redescriptions

As Fig. 3 reveals, even the deletion of a single row (in this case o_3) causes noticeable changes in the landscape of redescription terrains. In this section, we present further insights into the relationship between dataset characteristics and redescription space.

Impossibility Results

We begin by making some impossibility statements in the context of mining redescriptions. The first statement asserts an impossibility about finding even a single descriptor for certain object sets, and the second statement asserts an impossibility about finding any redescriptions at all in a dataset.

Lemma 7. *If two rows (o_i and o_j) of D are identical, there can be no descriptor involving o_i but not o_j.*

This is easy to see since no boolean expression can be constructed over D's columns that can discriminate between the two objects. The second impossibility result holds when D has at least as many rows as a truth table:

Theorem 2. *Given a ($n \times m$) dataset D such that every possible m-tuple binary vector (there are 2^m of them) is a row in D, let e be a descriptor defined on D's columns. Then*

$R(e) = \{e\}$, *i.e., no descriptor defined over the columns of* D *has a distinct redescription.*

Proof. Assume the contrary, i.e., there exists some $e' \neq e$ such that $O(e) = O(e')$. Then $Fr(R(e)) = Fr(R(e'))$. Let $V = F(Fr(R(e))) = F(Fr(R(e')))$. Next, let D' be the dataset restricted to the columns in V; D' then has all possible $|V|$-tuples. But now $O(e) \neq O(e')$ since $e' \neq e$. Thus e' must be the same as e and subsequently $R(e) = \{e\}$. □

Strong Possibility Result

We next show that if even one or few rows are absent in the dataset D, each descriptor e has a non-trivial redescription.

Theorem 3. *Given a* $(n \times m)$ *dataset* D *such that at least one of the* m*-tuple binary vectors is absent in D. Then for each descriptor* e *defined on* D*'s columns,* $|R(e)| > 1$*, i.e., every descriptor* e *defined over the columns of* D *has a redescription* $e' \neq e$*.*

Proof. Consider some expression e with the support rows as $O(e)$. Let the absent m-tuples be denoted as A; these correspond to the missing collection of rows. Consider the expression e' with $O(e')$ = $O(e) \cup A$. Since $A \cap O(e) = \phi$ and $A \neq \phi$, then e' must be distinct from e, but $O(e) = O(e')$ given D, hence e' is a redescription of e. Thus any e defi ned on D has a distinct redescription e' constructed as above. Hence the result. □

Corollary 1. *Given a* $(n \times m)$ *dataset* D *such that* p m*-tuple binary vectors are absent in D. Then for each descriptor* e *defined on* D*'s columns,* $|R(e)| = 2^p$*, i.e., every descriptor* e *defined over the columns of* D *has* $(2^p - 1)$ *distinct redescriptions.*

Proof. Let O^m = $\{o_1^m, o_2^m, \ldots, o_p^m\}$ be the missing vectors in the data. Consider $O(e)$, then there exists e' such that $O(e')$ = $O(e) \cup (O^{m'} \subseteq O^m)$. Then $e' \in X(e)$. Also there exists no e'' with $O(e'') \cap O^m = \phi$ such that $O(e) \subset O(e'') \subset O(e')$, hence e' is a redescription of e. There exist 2^p such $O^{m'}$ sets. Thus e can have $(2^p - 1)$ distinct redescriptions. □

Theorem 4. *(Dichotomy Law) Given a dataset* D *either no expression* e *has a distinct redescription or all expressions* e *on* D *have distinct redescriptions.*

Proof. Let D be an $n \times m$ matrix of elements. If $n = 2^m$ and all the rows are distinct, then by Theorem 2, no expression e has a redescription. Otherwise, by Theorem 3 each expression e has a distinct redescription. Hence the result. □

Forms of expression e

In light of the dichotomy law, to ensure that the problem of mining redescriptions is well-posed, we focus our attention on various biases on the form of e. Notice that if the bias is too general, then the bi-state phenomenon will continue to hold, e.g.:

	F_1	F_2	F_3
o_1	0	0	0
o_2	1	0	1
o_3	1	1	0
o_4	0	1	1
o_5	0	0	0

Figure 5: An example dataset matrix D to show that the dichotomy law does not hold for specifi c forms of expressions.

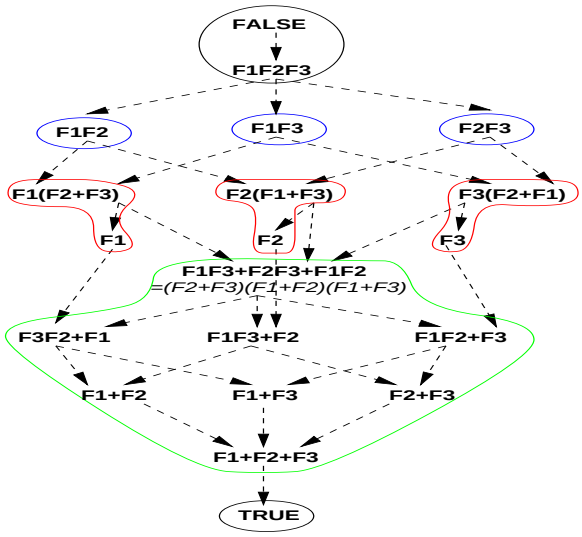

Figure 6: A relaxation lattice over monotone expressions of three boolean variables. The redescription terrains for the data of Fig. 5 are shown encircled in closed curves. Notice that for this class of expressions, the dichotomy law (Theorem 4) does not hold.

Corollary 2. *If the expressions are in CNF or DNF form, then the dichotomy law holds for the collection of descriptors.*

The bias we study in this paper pertains to monotone forms (Bshouty 1995) (with the expressions either in CNF or DNF). An alternative (not studied in detail here) is to choose a general expression with a small number of variables. We make the following observation.

Observation 1. *If the expressions are in (1) monotone form or (2) use only some* $p < m$ *variables, then the dichotomy law does not hold.*

Proof. (Case 1) Consider the dataset shown in Fig. 5. Let $e_1 = F_1F_2$, then $O(e_1) = \{e_3\}$ and has no distinct redescriptions in monotone form. However $e_2 = F_1(F_2 + F_3)$ has a redescription F_1 with $O(e_2) = \{2, 3\}$ (see Fig. 6). (Case 2) Let us restrict our attention to descriptors that use at most $p = 2$ variables. Let e_1 = $F_1 + F_2$ with $O(e_1)$ = $\{o_2, o_3, o_4\}$. Then $F_2 + F_3$ and $F_1 + F_3$ are both redescriptions of e_1. But $e_2 = F_1$ has no redescriptions (except itself). □

Mining Exact Redescriptions

We now present a general framework that, given a $n \times m$ dataset D and support $k \leq n$, identifies all descriptors e and their redescriptions $R(e)$ such that $|O(e)| \geq k$. We focus on mining exact redescriptions here; the next section deals with approximate redescriptions. We also begin by focusing only on conjunctive forms and later show how we can extend such an algorithm to mine more general forms, such as CNF or DNF. One advantage of the conjunctions bias is that when e' is a relaxation of e, we have $F(e') \subseteq F(e)$.

Our basic mining approach has two steps:

1. Compute the $O(R(e))$'s for the e's in the predefined form and extract $Fr(R(e))$ for each.
2. Use *CompRedscrp* to compute redescriptions of e from $Fr(R(e))$.

Computing O(R(e)), Fr(R(e))

We claim that, restricted to conjunctions, $Fr(R(e))$ is an expression that involves *all* the features that play a role in $R(e)$.

Lemma 8. $F(Fr(R(e))) = \cup_{e' \in R(e)} F(e')$

Proof. Let there exist $e' \in R(e)$ such that

$$F(e') \setminus F(Fr(R(e))) \neq \phi$$

Then clearly $e' \notin X(Fr(R(e)))$ which is a contradiction, hence the assumption must be wrong. Thus for each $e' \in R(e)$, $F(e') \subseteq F(Fr(R(e)))$. Further, $Fr(R(e)) \in R(e)$, hence the result. □

This result shows that if there is a mechanism for computing $O(e)$ where e involves as many variables as possible, it can be used for computing all the descriptors (and subsequently redescriptions). Therefore our algorithmic strategy is to compute biclusters in the dataset matrix D; the rows in the bicluster correspond to $O(R(e))$ and when the bicluster is maximal (see below), $F(Fr(R(e)))$ can be derived from the columns.

From Biclusters to Redescriptions

Given D, a bicluster is a non-empty collection of rows O and a non-empty collection of columns F such that for a fixed $j \in F$, $D_{ij} = c_j$ for a constant c_j, for each $i \in O$ and; there does not exist $i' \notin O$ s.t. $D_{i'j} = c_j$ for each $j \in F$.

The bicluster is maximal (also called a closed 'itemset' in the association rule mining literature (Zaki 2004)) if there does not exist $j' \notin F$ with $D_{ij'} = c_{j'}$ for each $i \in O$ and some fixed $c_{j'}$. These conditions define the 'constant columns' type of biclusters (see (Madeira & Oliveira 2004) for different flavors of biclusters used in the bioinformatics community). The bicluster is minimal if, for each $j \in F$, the collection of rows O and the collection of columns $F \setminus \{j\}$ is no longer a bicluster.

We have implemented a generic biclustering algorithm ($BiCluster$) that can find either maximal or minimal biclusters. Intuitively, the algorithm searches levelwise in the space of features, evaluating each in turn for inclusion in the biclusters. In transitioning from one level to another, there are three choices: include the new feature, include the negation of the new feature, or ignore it. Maintaining tuples $\mathcal{T}$ of (row set, column set) pairs the algorithm recursively computes the biclusters and, at the same time, computes the partial order of connectivity necessary to obtain redescriptions. $BiCluster$ has parameters to control the depth of expressions constructed, which branches of the search tree are explored (negated, non-negated, or both), and other user-specified constraints. The output of $BiCluster$ is utilized in the redescription mining routines below. Before we proceed to describe the specific routines, the following lemma is straightforward to verify.

Lemma 9. *Let $\overline{\mathcal{T}}$ be defined as follows: (1) $\overline{\mathcal{T}}[1] = O - \mathcal{T}[1]$ and (2) $\overline{\mathcal{T}}[2] = \{\overline{f} \mid f \in \mathcal{T}[2]\}$. $\overline{\mathcal{T}}$ is a minimal disjunction form if and only if $\mathcal{T}$ is a minimal conjunction form.*

Let C be all the biclusters computed. In the case of maximal biclusters, the time taken by BiCluster is $O(L \log n + mn)$ where $L = \sum_{c \in C} |c|$. In the case of minimal biclusters, the time taken by BiCluster is $O(|C| \log n + mn)$. Also, in each of the cases that follow, $BiCluster$ is invoked only a constant number (≤ 2) of times.

Mining redescriptions in monotone CNF

Since the forms are monotone, no negation of a variable (column) is permitted. We stage the mining of monotone CNF expressions into:

1. Find all minimal monotone disjunctions in D, by performing the following two substeps:
   (a) Let $\overline{D}$ be the negation of D defined as follows:
   $$\overline{D}_{ij} = \begin{cases} 1 & \text{if } D_{ij} = 0 \\ 0 & \text{otherwise} \end{cases}$$
   Find all minimal conjunctions in $\overline{D}$ using $BiCluster$ in the minimal mode;
   (b) Extract all minimal monotone disjunctions by negating each of these computed minimal conjunctions (see Lemma 9). Let the number of disjunctions computed be d in number.
2. Augment matrix D with the results of the last step. For each minimal disjunction form $\overline{\mathcal{T}}$, introduce a new column c in D with
   $$D_{ic} = \begin{cases} 1 & \text{if } i \in \overline{\mathcal{T}} \\ 0 & \text{otherwise} \end{cases}$$
   The augmented matrix, D', is then of size $n \times (m + d)$. Next, find all monotone conjunctions as maximal biclusters in D'.

Mining redescriptions in monotone DNF

There are two ways to computing the DNF form. The first is to compute the CNF forms and then derive the DNF forms for each CNF form. The other approach is to switch the order of calls to the routine $BiCluster$: first compute maximal conjunctions and next compute the minimal disjunctions. Due to space constraints, the details will be presented in the full version of the paper.

Mining Approximate Redescriptions

The Jaccard's coefficient $\mathcal{J}$ between two object sets O_1 and O_2 is given by:

$$\mathcal{J}(O_1, O_2) = \frac{|O_1 \cap O_2|}{|O_1 \cup O_2|}$$

$\mathcal{J} = 1.0$ iff $O_1 = O_2$. In practice, it is useful to consider approximate redescriptions with $0 < \mathcal{J} < 1.0$. Let $R_\theta(e)$ be the set of all approximate redescriptions of e with $\mathcal{J} \geq \theta$. This set can be computed as follows:

```
Let R be the set of all R_1's
  CompApproxRedscrp(e, θ)
  {
    O(R_θ(e)) ← O(R_1(e))
    For each R_1(e') ∈ R
      If (J(O(R(e)), O(R_1(e'))) ≥ θ) then
        O(R_θ(e)) ← O(R_θ(e)) ∪ O(R_1(e'))
  }
```

Notice that this algorithm is complete, i.e., *every* eligible approximate redescription is extracted. In practice however, this algorithm can be made very efficient using the partial order connectivity of the exact approximations.

Application Preview

Space considerations preclude us from desribing the many practical applications of redescription mining. We present a brief summary of a study conducted using bioinformatics datasets. Here, the objects are genes and the descriptors define membership in various categories of the public domain Gene Ontology (GO). GO actually comprises three parallel taxonomies, detailing cellular components (GO CEL), molecular function (GO MOL), and biological process (GO BIO) assignments. We utilized data from six different organisms (Baker's yeast, the model plant Arabidopsis, Worm, Fly, Mouse, and Human) and obtained descriptor definitions for these organisms from the GO database. Learning relationships between these descriptors, even within an organism, sheds valuable insight into what types of biological processes are localized in which parts of the cell, and handled by what types of molecular machinery. These relationships also help in mundane tasks such as functional assignment for uncategorized genes. Finally, tracking such redescriptions across the six organisms helps us understand how Eukaryotic organisms have specialized constructs or evolved new distinctions for performing similar (or related) roles. For ease of interpretation, we restricted our attention to only exact redescriptions and where the expressions on either side involve only at most two factors. Fig 7 (a) shows a simple redescription between a GO BIO (51013) and GO CEL (8352) categories over the Fly genome, involving 4 genes. This shows that genes localized under 'katanin' are involved in microtubule severing. We will return to Fig. 7 (b) and (c) shortly. Fig. 7 (d), (e), and (f) show redescriptions that hold in multiple organisms. Fig. 7 (d), in particular, shows a redescription straddling all three GO taxonomies, relating a set intersection to a single descriptor, and which holds over two genomes – Yeast and Worm. This redescription involves 12 genes in the Yeast genome and 4 genes in the Worm genome. Fig. 7 (f) gives a redescription over three genomes (Human, Mouse, Worm). This redescription involves 45 genes for Human, 30 genes for Mouse, and 16 genes for Worm. The potential for scientific discovery can be seen in these examples and, more clearly, in Fig. 7 (b & c). These two scenarios attempt to redescribe the same GO MOL descriptor in two different genomes; in the case of the Worm genome, 'translation release factor activity' is redescribed into the intersection of of 'translational termination' and 'cytoplasm.' Whereas in the case of the Arabidopsis genome, the cellular component is classified as 'unknown.' If sufficient homology exists between the genes involved in these redescriptions, we can transfer functional classifications from the Worm to the Arabidopsis genome; this showcases the ability of redescription mining to yield valuable insights into set overlaps and dissimilarities. This application is currently being extended into a general framework for reasoning about semantic similarity in biological taxonomies.

Related Work

Redescriptions were first studied in (Ramakrishnan *et al.* 2004) which also proposes an approach (CARTwheels) to mining redescriptions by exploiting two important properties of binary decision trees. First, if the nodes in such a tree correspond to boolean membership variables of the given descriptors, then we can interpret paths to represent set intersections, differences, or complements; unions of paths would correspond to disjunctions. Second, a partition of paths in the tree corresponds to a partition of objects. These two properties are employed in CARTwheels which grows two trees in opposite directions so that they are joined at the leaves. Essentially, one tree exposes a partition of objects via its choice of subsets and the other tree tries to grow to match this partition using a different choice of subsets. If partition correspondence is established, then paths that join can be read off as redescriptions. CARTwheels explores the space of possible tree matchings via an alternation process wherebytrees are repeatedly re-grown to match the partitions exposed by the other tree.By suitably configuring this alternation, we can guarantee, with non-zero probability, that any redescription existing in the dataset would be found. However, CARTwheels has a tendency to re-find already mined redescriptions as it searches for potentially unexplored regions of the search space.

This paper has proposed a new class of theoretically well-founded algorithms that do not suffer from the above drawback. Like (De Raedt & Kramer 2001), our approach can be construed as an integration of a version space approach (the systematic use of relaxation to structure the space of descriptors) with the emphasis of modern data mining software (to find *all* patterns satisfying a certain constraint). However, the focus on redescriptions has brought out interesting insights about structure theory (such as how redescriptions always exist for any descriptor in a complete bias) as well as improved understanding about algorithms (such as how finding redescriptions can exploit biclustering at its core).

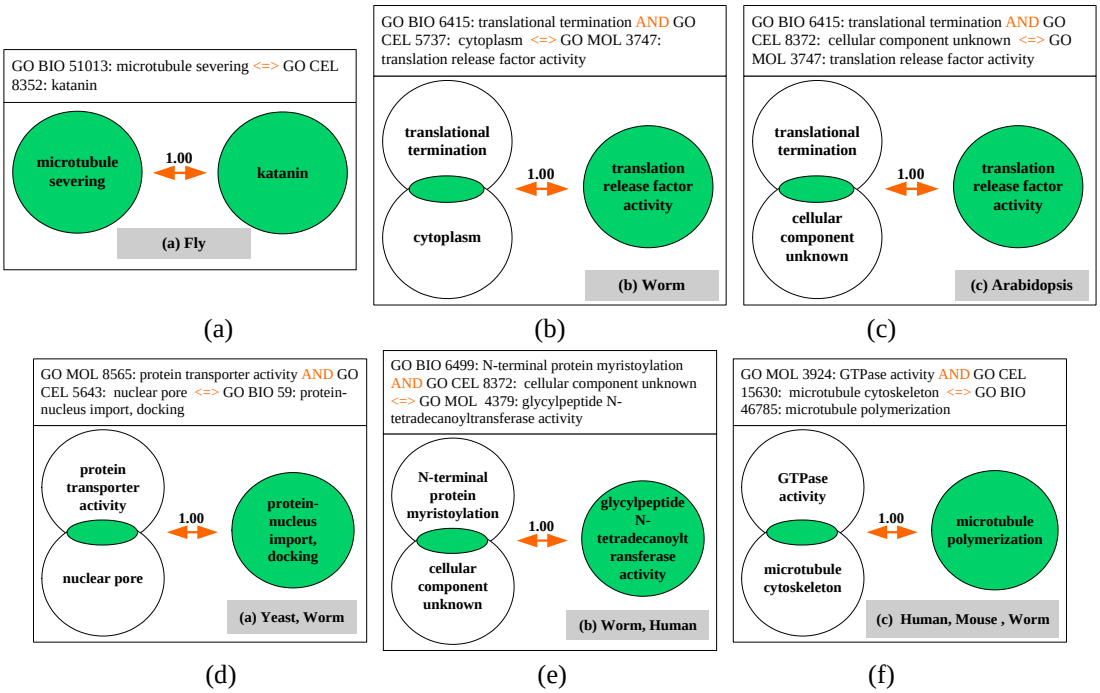

Figure 7: Six redescriptions mined between different taxonomies of the GO ontology. (Top) Redescriptions that hold in a particular organism. (Bottom) Redescriptions that hold in more than one organism.

Discussion

Our goal in this paper has been to introduce the redescription mining problem along with illustrations of what makes the problem interesting and important. Being able to reason about any arbitrary collection of sets in an expressive way will empower domain specialists to relate their disparate vocabularies and will further knowledge discovery. Our future work involves a detailed experimental study in a large domain and exploring applications in schema matching & relational mining using the ideas presented here as building blocks.

Acknowledgements

Deept Kumar helped us gather the results presented in this paper. This work is supported in part by US NSF grants ITR-0428344, EIA-0103660, IBN-0219332, NIH grant N01-A1-40035, and DoD MURI grant N00014-01-1-0852 and sabbatical support (from NYU and IISc, Bangalore) to the second author.

References

Bshouty, N. 1995. Exact Learning Boolean Functions via the Monotone Theory. *Information and Computation* Vol. 123(1):146–153.

De Raedt, L., and Kramer, S. 2001. The Levelwise Version Space Algorithm and its Application to Molecular Fragment Finding. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, 853–862.

Doan, A.; Domingos, P.; and Halevy, A. 2003. Learning to Match the Schemas of Data Sources: A Multistrategy Approach. *Machine Learning* Vol. 50(3):279–301.

Fisher, D. 1987. Knowledge Acquisition via Incremental Conceptual Clustering. *Machine Learning* Vol. 2(2):139–172.

Madeira, S., and Oliveira, A. 2004. Biclustering Algorithms for Biological Data Analysis: A Survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* Vol. 1(1):24–45.

Michalski, R. 1980. Knowledge Acquisition through Conceptual Clustering: A Theoretical Framework and Algorithm for Partitioning Data into Conjunctive Concepts. *International Journal of Policy Analysis and Information Systems* Vol. 4:219–243.

Mitchell, T. 1982. Generalization as Search. *Artificial Intelligence* Vol. 18(2):203–226.

Ramakrishnan, N.; Kumar, D.; Mishra, B.; Potts, M.; and Helm, R. 2004. Turning CARTwheels: An Alternating Algorithm for Mining Redescriptions. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*, 266–275.

Russell, S. 1989. *The Use of Knowledge in Analogy and Induction*. London: Pitman.

Valdes-Perez, R.; Pericliev, V.; and Pereira, F. 2000. Concise, Intelligible, and Approximate Profi ling of Multiple Classes. *International Journal of Human-Computer Studies* Vol. 53(3):411–436.

Valdes-Perez, R. 1999. PickNiche Software. http://www.cs.cmu.edu/∼sci-disc/pickniche.html.

Zaki, M. 2004. Mining Non-Redundant Association Rules. *Data Mining and Knowledge Discovery* Vol. 9(3):223–248.