Redescription Mining: Structure Theory and Algorithms

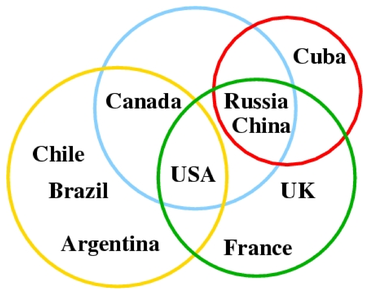
Laxmi Parida* and Naren Ramakrishnan†

* IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA

† Department of Computer Science, Virginia Tech, VA 24061, USA

Redescriptions

Redescriptions are shifts-of-vocabulary, patterns that define a selected subset of objects in at least two ways. The input to redescription mining is a collection of sets (*descriptors*) such as the following vocabulary over countries.



An example redescription is: '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' that redefines the set {Russia, China}. The goal of redescription mining is to find those subsets that afford multiple definitions along with these definitions.

Redescription mining generalizes multiple machine learning paradigms

Profiling Classes: Redescriptions lose the distinction between classes and features and provide necessary as well as sufficient descriptors to cover other descriptors.

Niche Finding: Specialization of class profiling to singleton sets. Example: 'Wake Forest University is the only suburban Baptist University' is a redescription that identifies a niche for Wake Forest University.

Analogical Reasoning: Functional determinations are key to analogical reasoning. Redescriptions are specialization of determinations to binary attributed datasets.

Story Telling: A sequence of approximate redescriptions is a story between two (potentially) disjoint sets. For instance, if we think of words as sets of (letter, position) pairs, an example of a story is: PURE → PORE → POLE → POLL → POOL → WOOL.

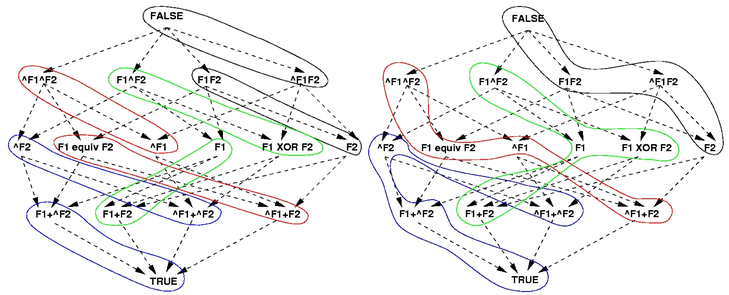
Schema Matching: Computing redescriptions can be viewed as inferring one-to-one mappings between propositional schema represented as set systems. Suggests natural extension of redescription to predicate vocabularies, akin to inductive logic programming.

Mining Redescriptions

Given a space of descriptors and possible descriptor expressions, redescriptions are 'terrains' over the lattice of relaxations among the expressions. Consider the example dataset D of three objects $\times$ two descriptors:

	F_1	F_2
o_1	0	0
o_2	1	0
o_3	1	1

The left figure below depicts the redescription terrains for all three rows. The right figure depicts the change in terrains if the last row (o_3) is deleted. Each terrain has a unique frontier and each non-frontier descriptor has a unique frontier at the shortest possible relaxation distance. This means that we can compute just the frontiers to mine all redescriptors.



Theoretical Results

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

Theorem: 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. Then no descriptor defined over the columns of D has a distinct redescription.

Theorem: Given a $(n \times m)$ dataset D such that at least one of the m-tuple binary vectors is absent in D. Then every descriptor e defined over the columns of D has a redescription $e' \neq e$.

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

Making Redescription Mining Well Posed

A bias on the form of descriptor expressions helps violate the dichotomy law and ensure well posedness of redescription mining. For instance, if the expressions are either in (i) monotone form or (ii) use only some $p < m$ variables, then the dichotomy law does not hold.

Composing Redescription Mining Solutions

A redescription mining algorithm can be compositionally constructed from a (constant column) biclustering algorithm. Example for monotone CNF: (i) First negate given dataset D into $\overline{D}$; mine minimal biclusters in $\overline{D}$. (ii) Second, negate each of the mined conjunctions (giving disjunctions) and augment dataset D to yield D'. (iii) Finally, mine maximal conjunctions in D'.

Redescribing GO Descriptors