Proceedings of the International MultiConference of Engineers and Computer Scientists 2008 Vol I
IMECS 2008, 19-21 March, 2008, Hong Kong

Confidence-based Concept Discovery in Multi-Relational Data Mining

Yusuf Kavurucu* Pinar Senkul† Ismail Hakki Toroslu‡

Abstract—**Multi-relational data mining has become popular due to the limitations of propositional problem definition in structured domains and the tendency of storing data in relational databases. Several relational knowledge discovery systems have been developed employing various search strategies, heuristics, language pattern limitations and hypothesis evaluation criteria, in order to cope with intractably large search space and to be able to generate high-quality patterns. In this work, a new ILP-based concept discovery method is described in which user-defined specifications are relaxed. Moreover, this new method directly works on relational databases. In addition to this, a new confidence-based pruning is used in this technique. A set of experiments are conducted to test the performance of the new method.**

Keywords: Multi-relational Data Mining, Concept Discovery, ILP

1 Introduction

Due to the impracticality of single-table data representation, multi-relational databases are needed to store complex data for real life, data intensive applications. This has led to the development of multi-relational learning systems that are directly applied to relational data [4]. Most relational upgrades of data mining and concept learning systems employ first-order predicate logic as representation language for background knowledge and data structures/patterns. The learning systems, which induce logical patterns or programs valid for given background knowledge, have been gathered under a research area, called Inductive Logic Programming (ILP) [8].

In this work, we propose a predictive [1] concept learning ILP system, namely Confidence-based Concept Discovery (C^2D), which employs relational association rule mining concepts and techniques. It utilizes absorption operator of inverse resolution for generalization of concept instances in the presence of background knowledge and refines these general patterns into frequent and strong concept definitions with an Apriori-based specialization operator based on confidence. The major contributions of this work can be listed as follows:

1. The main difficulty in relational ILP systems is searching in intractably large hypothesis spaces. In order to cope with this problem, relational ILP systems put strong declarative biases on the semantics of hypotheses. In this work, we aimed to relax the declarative biases in such a way that body clauses may have variables which do not exist in the head predicate. In order to reduce the search space, a confidence-based pruning mechanism is used.

2. Many multi-relational rule induction systems require the user to determine the input-output modes of predicate arguments. Since mode declarations require a high level Prolog and domain knowledge, it is not meaningful to expect such a declaration from a normal user. Instead of this, we use the information about relationships between entities in the database if given.

3. The expected error of an hypothesis according to positive versus all (positive and negative) examples do not have much difference if the number of examples is large enough [7]. In other words, logic programs are learnable with arbitrarily low expected error from only positive examples. As relational databases contain only positive information, a pure multi-relational data mining system based on logic programming could be developed which relies on only positive instances stored as relations. Therefore, the proposed system directly works on relational database, without any requirement of negative instances.

4. The definition of confidence is modified to apply Closed World Assumption (CWA) in relational databases. We introduce type relations to the body of the clauses in order to express CWA.

This paper is organized as follows: Section 2 presents the related work. Section 3 explains the proposed method. Section 4 discusses the experimental results of the proposed method on real-world problems. Finally, Section 5 includes concluding remarks.

*Contact author (Ph.D. Student). Middle East Technical University Computer Eng. Dept. Email: yusuf.kavurucu@ceng.metu.edu.tr

†Asst.Prof. Middle East Technical University Computer Engineering Department 06531 Ankara, Turkey. Email: karagoz@ceng.metu.edu.tr

‡Prof. Middle East Technical University Computer Eng. Dept. Email: toroslu@ceng.metu.edu.tr

[1] In predictive ILP systems, there is a specific target concept to be learned in the light of past experiences

2 Related Work

In this section, we describe some of the well-known ILP systems related to our system.

PROGOL [6] is a top-down relational ILP system, which is based on inverse entailment. It performs a search through the refinement graph. It reduces the hypothesis space by using a set of mode declarations given by the user, and a most specific clause (also called bottom clause) as the greatest lower bound of the refinement graph. A bottom clause is a maximally specific clause, which covers a positive example and is derived using inverse entailment. PROGOL starts the search with empty body, and goes through in the refinement lattice, which has literals that are elements of the bottom clause. PROGOL chooses the clause having maximum f value [6]. PROGOL applies the covering approach and supports learning from positive data.

A Learning Engine for Proposing Hypotheses (ALEPH) [10] is a top-down relational ILP system based on inverse entailment similar to PROGOL. The basic algorithm is the same as PROGOL whereas it is possible to apply different search strategies, evaluation functions and refinement operators. It is also possible to define more settings in ALEPH such as minimum confidence and support.

WARMR [1] is a descriptive ILP system that employs Apriori rule as search heuristics. Therefore, it does not search for clauses in order to model a target relation. Instead, it finds frequent queries having the target relation by using support criteria. Then, it is possible to extract association rules having target relation in the head according to confidence criteria.

The proposed work is similar to ALEPH as both systems produce concept definition from the given target. WARMR is another similar work in a sense that, both systems employ Apriori-based searching methods. In contrast to ALEPH and WARMR, our system does not need input/output mode declarations. It only requires type specifications of the arguments, which already exist together with relational tables corresponding to predicates. Some of the ILP-based systems require negative information, whereas our system directly works on databases which have only positive data. In our algorithm, negative information is implicitly described in the data sets according to CWA. Finally, it presents a new confidence-based hypothesis evaluation criterion and search space pruning method.

3 C^{2}D: Confidence-based Concept Discovery Method

C^{2}D is a concept discovery system that uses first-order logic as the concept definition language and generates a set of definite clauses having the target concept in the head. However, C^{2}D only allows unification of predicate arguments having same types. It is developed on the basis of the systems described in [12, 13]. In C^{2}D, two mechanisms are effective for pruning the search space. The first one is a generality ordering on the concept clauses based on θ-subsumption and is defined as follows: A definite clause C θ-subsumes a definite clause C$'$, i.e. at least as general as C$'$, if and only if $\exists\theta$ such that: $head(C) = head(C')$ and $body(C') \supseteq body(C)\theta$. The second one, which is new in C^{2}D, is the *confidence* which is utilized as follows: If the confidence value of a clause is not higher than the confidence values of the two parent clauses in the Apriori search lattice, then it is pruned. By this way, in the solution path, each specialized clause has higher confidence value than its parents. A similar aproach is used in the Dense-Miner system [9] for traditional association rule mining.

Another new feature of C^{2}D is its parametric structure for *support, confidence, recursion* and *f-metric* definitions. The user can set support threshold and she/he can allow or disallow the use of support as a part of the pruning mechanism. It is possible to set confidence threshold for selecting the best clause, so that the best clause will have an acceptable confidence value. Similarly, by changing the value of the recursion parameter, it is possible to allow generating recursive or only linearly recursive hypothesis, or totally disallow recursive concept definitions. Another parametric declaration is for the *f-metric* (adapted from f-score formula [5]), whose definition is as follows:

$$f\text{-}metric = ((B^2 + 1) \times confidence \times support) \,/\, ((B \times confidence) + support))$$

The user can emphasize the effect of support or confidence by changing the value of B.

The database given in Figure 1, is used as a running example in this paper. In this example, *daughter (d)* is the concept to be learned, and two concept instances are given. Background facts of two relations, namely *parent (p)* and *female (f)* are provided. Finally, types of the attributes of relations are listed.

Concept Inst.	**Backgr. Facts**	**Type Dec.**
d(mary, ann).	p(ann, mary).	d(person, person).
d(eve, tom).	p(ann, tom).	p(person, person).
	p(tom, eve).	f(person).
	f(ann).	
	f(mary).	
	f(eve).	

Figure 1: The database of the daughter example with type declarations

3.1 Improved Confidence Definition

Two criteria are important in the evaluation of a candidate concept rule: *support* and *confidence*. The *support* value of a definite clause C is defined as the number of

different bindings for the variables in the head relation that satisfies the clause, divided by the the number of different bindings for the variables in the head relation. The *confidence* of a definite clause C is defined as the number of different bindings for the variables in the head relation that satisfies the clause, divided by the number of different bindings for the variables in the head relation that satisfies the body literals. As the head relation is the key relation in our method, the support and confidence calculation of a clause is same as calculation in query extensions. These values can be obtained with SQL queries given in [2].

In this work, the application of the confidence is modified since the current definition has a problem. In order to illustrate the problem with the classical definition of confidence, consider the following example clauses:

d(X, Y) :- p(Y, tom). (s=0.5, c=1.0)

d(X, Y) :- f(X). (s=1.0, c=0.67)

Confidence is the ratio of number of positive instances deducible from the clause over number of examples deducible from the clause. In other words, it shows how strong the clause is. For the first clause, the confidence value shows that it is very strong. However, out of the following four deducible facts *d(ann, ann), d(mary, ann), d(tom, ann)* and *d(eve, ann)*, only one of them (only d(mary, ann) is positive) exists in the database. As a result, the first clause covers some negative instances. Similarly, for the second clause, the confidence value is also high. The facts *d(ann, ann), d(ann, mary), d(ann, tom), d(ann, eve), d(mary, ann), d(mary, mary), d(mary, tom), d(mary, eve), d(eve, ann), d(eve, mary), d(eve, tom)* and *d(eve, eve)* are deducible from the second clause, but only 2 of them (only d(mary, ann) and d(eve, tom) are positive) exist in the database. Out of 12 possible ground instances, only 2 of them are concept instances. The confidence of this rule must be very low (such as $2/12 = 0.17$).

In order to solve this problem, we add type relations to the body of the clause corresponding to the arguments of the head predicate whose variable does not appear in the body predicates. The type tables for the arguments of the target relation are created in the database (if they do not exist). For the *daughter* example, *person* table is the type table. The type table contains all possible values of the corresponding argument of the target relation in the database. For the *daughter* example, *person* table contains 4 records which are *ann, mary, tom* and *eve*. Each new literal has a relation name as the corresponding head predicate argument type and has one argument that is the same as the corresponding head predicate argument. The rules obtained by adding type literals are used only to compute the confidence values, and for the rest of the computation, original rules without type literals are used.

The addition of type relations models the positive instances better and reflects the confidence value correctly. By this way, negative instances can be deduced as in CWA. Besides this, since the type relation is always true for the instance, this modification does not affect the semantics of the clause. In addition, the definition of the confidence query remains intact.

According to these modifications, the new clauses' support and confidence values for the daughter example are as follows:

d(X, Y) :- p(Y, tom), person(X). (s=0.5, c=0.25)

d(X, Y) :- f(X), person(Y). (s=1.0, c=0.17)

3.2 The Algorithm

The algorithm of C^{2}D, given in Figure 2, starts with selecting a positive concept instance. The most general clauses with two literals, one in the head and one in the body, that entail the positive example are generated and then the concept rule space is searched with an Apriori-based specialization operator. In the refinement graph, if support parameter is on and the frequency of a clause is below the support threshold, it is pruned as an infrequent clause. In addition to this, clauses whose confidence values are not higher than their parents' confidence values are also eliminated. When the maximum depth reached or no more candidate clause can be found, if confidence parameter is on, then the clauses that have less confidence value than the confidence threshold are eliminated for the solution set. Among the produced strong and frequent rules, according to the given hypothesis evaluation criteria, the best clause is selected and the rule search is repeated for the remaining concept instances that are not in the coverage of the selected hypothesis clauses. If there is no possible best clause found for the selected positive concept instance, then the algorithm will select another positive concept instance. In the rest of this section, the main steps of the algorithm are described.

Generalization: The generalization step is similar to the approach in [13]. After picking the first uncovered positive example, C^{2}D searches facts related to selected concept instance in the database, including the related facts that belong to the target concept in order for the system to induce recursive rules. Two facts are related if they share the same constant in the predicate argument positions of the same type. Then, the system generalizes the concept instance with all related facts. If a primary-foreign key relation exist between the head and body relations, the foreign key argument of the body relation can only have the same variable name as the primary key argument of the head predicate in this step. In this way, the key relation exists in the following specialization steps.

- **Input:** I: concept inst. set, BF: back. facts (from DB)
- **Output:** H: hypothesis set (initially H=$\emptyset$)
- **Parameters:** *min-sup*, *min-conf*, B (used in f-metric)
  md: maximum depth (body literal count)
- **Local Variables:** C_i: candidate clauses set at level i
  d: level, G: generalization clauses set
  FSC: frequent and strong clauses set
- Repeat until I is covered by H (until $I = \emptyset$) or
  No more possible G can be found according to given params:
  1. Select p from I.
  2. GENERALIZATION: Generate G of p by using BF
  3. Initialize C_1:=G, FSC:=$\emptyset$, d:=1
  4. REFINEMENT OF GENERALIZATION:
  While $C_d \neq \emptyset$ and $d \leq md$
     a. $FSC := FSC \cup$ FREQ_STR_CLS(C_d, *min-sup*)
     b. C_{d+1}= CAND_GEN(FSC, *min-sup*)
        i. UNION: For each pair of the clauses in level d compute each possible union clause.
        ii. For each union clause satisfying *min-sup*
        -SPECIALIZATION: Generate clauses by unifying existential variables
        -FILTERING: Discard non-frequent and less-confident clauses
     c. d:=d+1.
  5. EVALUATION:
     Eliminate clauses from FSC according to *min-conf*
     Select c_{best} from FSC
  6. COVERING: Compute $I_c \subseteq I$ covered by c_{best}
  7. $H := H \cup c_{best}$
  8. $I := I$ - I_c
-Return H.

Figure 2: C^{2}D algorithm

In the *daughter* example, C^{2}D selects the first concept instance *daughter(mary, ann)* and then finds the related fact set of the current concept instance {*parent(ann, mary), parent(ann, tom), female(ann), female(mary)*}. The system generalizes *daughter(mary, ann)* with each related fact. For instance, by applying absorption operator to the *daughter(mary, ann)* and *parent(ann,mary)*, the concept descriptions of the form

$\{daughter(mary, ann) \ :- \ parent(ann, mary)\}\theta_2^{-1}$ are derived.

Refinement of Generalization: C^{2}D refines the two literal concept descriptions with an Apriori-based specialization operator that searches the definite clause space in a top-down manner, from general to specific. As in Apriori, the search proceeds level-wise in the hypothesis space and it is mainly composed of two steps: frequent clause set selection from candidate clauses and candidate clause set generation as refinements of the frequent clauses in the previous level. The standard Apriori search lattice is extended in order to capture first-order logical clauses and the candidate generation and frequent pattern selection tasks are customized for first-order logical clauses.

In the *daughter* example, for the concept instance *daughter(mary, ann)*, two literal generalizations are generated in the presence of related facts and the first level of the search lattice is populated with these generalizations. With the support threshold value 0.8, the system eliminates the infrequent clauses. Notice that among 18 clauses generated for *daughter(mary, ann)* and *parent(ann, mary)*, only 6 of them satisfy the threshold (bold ones below). Other clauses below are generated from other generalizations.

d(X, Y) :- p(Z, mary).	**d(X, Y) :- p(Y, X).**
d(X, Y) :- p(Z, X).	**d(X, Y) :- p(ann, Z).**
d(X, Y) :- p(Y, Z).	**d(X, Y) :- p(Z, U).**
d(X, Y) :- p(Z, tom).	d(X, Y) :- f(X).
d(X, Y) :- f(Z).	

In the candidate clause generation step, candidate clauses for the next level of the search space are generated. It is composed of three important steps:

1. Frequent clauses of the previous level are joined to generate the candidate clauses via union operator. In order to apply the union operator to two frequent definite clauses, these clauses must have the same head literal, and bodies must have all but one literal in common. Since only clauses that have the same head literal are combined, the search space is partitioned into disjoint Apriori sublattices according to the head literal.

2. For each frequent union clause, a further specialization step is employed that unifies the existential variables of the same type in the body of the clause. By this way, clauses with relations indirectly bound to the head predicate can be captured.

3. Except for the first level, the candidate clauses that have confidence value not higher than parent's confidence values are eliminated.

Evaluation: For the first instance of the target concept, which has not been covered by the hypothesis yet, the system constructs the search tree consisting of the frequent and confident candidate clauses that induce the current concept instance. Then it eliminates the clauses having less confidence value than the confidence threshold. Finally, the system decides on which clause in the search tree represents a better concept description than other candidates according to f-metric definition.

Covering: After the best clause is selected, concept instances covered by this clause are determined and removed from the concept instances set. The main iteration continues until all concept instances are covered or no more possible candidate clause can be found for the uncovered concept instances.

In the *daughter* example, the search tree constructed for the instance *daughter(mary, ann)* is traversed for the best clause. The clause *d(X, Y) :- p(Y, X), f(X)* with support value of 1.0 and the confidence value of 1.0 (f-metric=1.0) is selected and added to the hypothesis. Since all the concept instances are covered by this rule, the algorithm

terminates and outputs the following hypothesis:

$daughter(X, Y)$:- $parent(Y, X), female(X)$.

4 Experimental Results

One of the interesting test cases that we have used is a complex family relation, "same-generation" learning problem. In the data set, 344 pairs of actual family members are given as positive examples of *same-generation (sg)* relation. Additionally, 64 background facts are provided to describe the *parental (p)* relationships in the family. As there are 47 different person in the examples, the *person* table (type table) has 47 records. In this experiment, only linear recursion is allowed and B value is set to be 1. We set the support threshold as 0.3, confidence threshold as 0.6 and maximum depth as 3.

C^{2}D finds the following clauses for this data set:

sg(X, Y) :- sg(Z, U), p(Z, X), p(U, Y).

sg(X, Y) :- sg(Z, U), p(Z, Y), p(U, X).

sg(X, Y) :- p(Z, X), p(Z, Y).

For this data set, ALEPH and PROGOL cannot find a solution under default settings. Under strong mode declarations and constraints, ALEPH finds the following hypothesis:

sg(X, Y) :- p(Z, X), p(Z, Y).

sg(X, Y) :- sg(X, Z), sg(Z, Y).

sg(X, Y) :- p(Z, X), sg(Z, U), p(U, Y).

However, PROGOL can only find "sg(X, Y) :- sg(Y, Z), sg(Z ,X)." as a solution.

The experiment shows that, C^{2}D can find the correct hypothesis set for the same generation problem whereas ALEPH and PROGOL cannot.

Among the experiments we conducted, another test case is a well-known benchmark problem called *Finite Element Mesh Design* [3]. The task is to learn the rules to determine the number of elements for a given edge in the presence of the background knowledge such as type of edges, boundary conditions and geometric positions.

There are 223 positive training examples and 1474 background facts in the data set. The target relation *mesh* has two arguments having *element* and *integer* type. The primary key for the target relation is *element* and it exists in all background relations as a foreign key. The type tables *element* and *integer* are created having 278 and 13 records. The test relation has 55 examples.

For Mesh Design data set, recursion is disallowed, support threshold is set as 0.1, B is set as 1 and maximum depth is set as 3. We test the data set on several confidence thresholds (0.1 through 0.5). C^{2}D is run on this data set by deriving a set of definite clauses for the four structures *(b, c, d, e)* and these clauses are tested on the remaining structure *(a)*. The details of the results and coverage of previous systems are shown in Figure 3.

System		Cov. (over 55 records)
FOIL		17
GOLEM		17
PROGOL		17
MFOIL		19
ALEPH (strict decl.)		26
C^{2}D (with min-conf)	0.1	31
	0.2	25
	0.3	15
	0.4	19
	0.5	17

Figure 3: Test results for the mesh-design data set

Selection of parameters is important to induce meaningful results for sparse data sets such as Mesh Design data set. For example, we get different results according to different minimum confidence threshold values. For some confidence thresholds, C^{2}D finds better results according to previous systems.

The third experiment is conducted on the first PTE challenge data set. In this data set, compounds are classified as carcinogenic or non-carcinogenic in the presence of background knowledge such as atom-bond structures, mutagenicity and structural groups. There are 298 compounds in the training set, 39 compounds exist in the test set. The background knowledge has rougly 25,500 facts [11]. The target relation *pte-active* has two arguments having *drug* and *bool* type. The primary key for the target relation is *drug* and it exists in all background relations as a foreign key. The type tables *drug* and *bool* are created having 340 and 2 (T/F) records.

For this experiment, recursion is disallowed, support threshold is set as 0.05, confidence threshold as 0.7, B is set as 1 and maximum depth is set as 3. The predictive accuracy of the hypothesis set is computed by the proportion of the sum of the carcinogenic concept instances classified as positive and non-carcinogenic instances classified as negative to the total number of concept instances that the hypothesis set classifies. The predictive accuracies of the state-of-art methods and C^{2}D for PTE-1 data set are listed in Figure 4. You may refer to [11] for more information on these systems.

Ashby and RASH are special systems which are developed for this kind of problem sets so that their predictive accuracies are higher than our method. The hypothesis set of C^{2}D does not include any rule about the molecular substructures composing of atom and bond relations. Also, C^{2}D cannot handle continuous data (such as atom charge) with comparison operators. Actually, PRO-

Method	Type	Pred. Acc.
Ashby	Chemist	0.77
PROGOL	ILP	0.72
RASH	Biological Potency Analysis	0.72
C^{2}D	**ILP + DM**	**0.69**
TIPT	Propositional ML	0.67
Bakale	Chemical Reactivity Analysis	0.63
Benigni	Expert-guided Regression	0.62
DEREK	Expert System	0.57
TOPCAT	Statistical Discrimination	0.54
COMPACT	Molecular Modeling	0.54

Figure 4: Predictive accuracies for the PTE-1 data set

GOL's hypothesis is composed of nine first-order clauses and five of them are related to the atom-bond relations in the molecule. We can conclude that the predictive accuracy of C^{2}D will increase if carcinogenesis data is normalized in a preprocessing step and also continuous data is handled.

5 Conclusion and Future Work

This work presents a concept discovery system, C^{2}D, with a new improved confidence-based hypothesis evaluation criterion and confidence-based search space pruning mechanism. For the improved hypothesis evaluation criterion, conventional confidence definition is modified by adding type predicates for the arguments of the head predicate that do not appear in the body. By this way, the need for the inclusion of negative examples in the confidence value of the clause is removed, without changing the semantics of the clauses.

Confidence-based pruning is used in the candidate filtering phase. If the confidence value of the generated clause is not higher than confidence values of its parents, it means that the specifications through it will not improve the hypothesis to be more confident. By this way, such clauses are directly eliminated at early steps.

In addition to these features, as in the systems in [12, 13], C^{2}D combines rule extraction methods in ILP and Apriori-based specialization operator. The main benefits of this approach are relaxing the strong declarative biases and applying the method on relational databases. In addition to these, this system does not require user specification of input/output modes of arguments of predicates and negative concept instances. Instead of this, it uses the information provided by the relationships between entities in the database if given. Thus, it provides a suitable data mining framework for non-expert users who are not expected to know much about the semantic details of large relations they would like to mine.

The proposed system is tested on several benchmark problems including the same-generation, mesh design and PTE-1 challenge. The experiments reveal promising test results that are comparable with the performance of current state-of-the-art knowledge discovery systems. Also, it may find better rules if one-to-many relationships and continuous data can be handled in the algorithm.

References

[1] L. Dehaspe and L. D. Raedt. Mining association rules in multiple relations. In *ILP'97: Proceedings of the 7th International Workshop on Inductive Logic Programming*, pages 125–132, London, UK, 1997. Springer-Verlag.

[2] L. Dehaspe and H. Toivonen. Discovery of relational association rules. In S. Džeroski and N. Lavrač, editors, *Relational Data Mining*, pages 189–212. Springer-Verlag, September 2001.

[3] B. Dolšak and S. Muggleton. The application of inductive logic programming to finite-element mesh design. In S. Muggleton, editor, *Inductive Logic Programming*, pages 453–472. Academic Press, 1992.

[4] S. Džeroski. Multi-relational data mining: an introduction. *SIGKDD Explorations*, 5(1):1–16, 2003.

[5] C. Goutte and E. Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *European Colloquium on IR Research (ECIR'05)*, pages 345–359, Xerox Research Centre Europe 6, chemin de Maupertuis F-38240 Meylan, France, 2005. Springer.

[6] S. Muggleton. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995.

[7] S. Muggleton. Learning from positive data. In *Proceedings of the 6th International Workshop on Inductive Logic Programming*, volume 1314 of *Lecture Notes in Artificial Intelligence*, pages 358–376. Springer-Verlag, 1996.

[8] S. Muggleton. Inductive Logic Programming. In *The MIT Encyclopedia of the Cognitive Sciences (MITECS)*. MIT Press, 1999.

[9] J. Roberto J. Bayardo, R. Agrawal, and D. Gunopulos. Constraint-based rule mining in large, dense databases. *Data Mining and Knowledge Discovery*, 4(2-3):217–240, 2000.

[10] A. Srinivasan. The aleph manual, 1999.

[11] A. Srinivasan, R. D. King, S. Muggleton, and M. J. E. Sternberg. Carcinogenesis predictions using ILP. In *Proceedings of the 7th International Workshop on Inductive Logic Programming*, volume 1297, pages 273–287. Springer-Verlag, 1997.

[12] S. D. Toprak. A new hybrid multi-relational data mining technique. Master's thesis, Middle East Technical University, Computer Engineering Department, Ankara, Turkey, May 2005.

[13] S. D. Toprak, P. Senkul, Y. Kavurucu, and I. H. Toroslu. A new ILP-based concept discovery method for business intelligence. In *ICDE Workshop on Data Mining and Business Intelligence*, April 2007.